Patent Assignment Abstract of Title

Total Assignments: 4

Application #: 10972588 **Filing Dt:** 10/25/2004 **Patent #:** 7624302 **Issue Dt:** 11/24/2009

PCT #: NONE **Intl Reg #:** **Publication #:** US20060090064 **Pub Dt:** 04/27/2006

Inventors: Scott L. Michaelis, Anurupa Rajkumari, William B. McHardy

Title: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

Assignment: 1

Reel/Frame: 015932 / 0117 **Received:** 11/01/2004 **Recorded:** 10/25/2004 **Mailed:** 04/22/2005 **Pages:** 3

Conveyance: ASSIGNMENT OF ASSIGNORS INTEREST (SEE DOCUMENT FOR DETAILS).

Assignors: MICHAELIS, SCOTT L. **Exec Dt:** 10/20/2004

RAJKUMARI, ANURUPA **Exec Dt:** 10/20/2004

MCHARDY, WILLIAM B. **Exec Dt:** 10/19/2004

Assignee: HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P.
20555 SH 249
HOUSTON, TEXAS 77070

Correspondent: HEWLETT-PACKARD COMPANY
INTELLECTUAL PROPERTY ADMINISTRATION
RECORDS MANAGER
P.O. BOX 272400
FORT COLLINS, COLORADO 80527-2400

Assignment: 2

Reel/Frame: 037079 / 0001 **Received:** 11/09/2015 **Recorded:** 11/09/2015 **Mailed:** 11/20/2015 **Pages:** 1112

Conveyance: ASSIGNMENT OF ASSIGNORS INTEREST (SEE DOCUMENT FOR DETAILS).

Assignor: HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P. **Exec Dt:** 10/27/2015

Assignee: HEWLETT PACKARD ENTERPRISE DEVELOPMENT LP
11445 COMPAQ CENTER DRIVE WEST
HOUSTON, TEXAS 77070

Correspondent: HEWLETT PACKARD ENTERPRISE
3404 E. HARMONY ROAD MS 79
FORT COLLINS, CO 80528

Assignment: 3

Reel/Frame: 052567 / 0734 **Received:** 05/05/2020 **Recorded:** 05/05/2020 **Mailed:** 05/06/2020 **Pages:** 6

Conveyance: ASSIGNMENT OF ASSIGNORS INTEREST (SEE DOCUMENT FOR DETAILS).

Assignors: HEWLETT PACKARD ENTERPRISE DEVELOPMENT LP **Exec Dt:** 04/23/2020

HEWLETT PACKARD ENTERPRISE COMPANY **Exec Dt:** 04/23/2020

Assignee: SONRAÍ MEMORY, LTD.
SUITE 23, THE HYDE BUILDING, THE PARK, CARRICKMINE
DUBLIN, IRELAND DUBLIN 18

Correspondent: ADAM MULLOOLY
SUITE 23, THE HYDE BUILDING, THE PARK, CARRICKMINE
DUBLIN, DUBLIN 18 IRELAND

Assignment: 4

Reel/Frame: 058992 / 0571 **Received:** 02/11/2022 **Recorded:** 02/11/2022 **Mailed:** 02/22/2022 **Pages:** 5

Conveyance: ASSIGNMENT OF ASSIGNORS INTEREST (SEE DOCUMENT FOR DETAILS).

Assignor: SONRAÍ MEMORY, LTD. **Exec Dt:** 01/19/2022

Assignee: FORAS TECHNOLOGIES LTD.
THE HYDE BUILDING, SUITE 23,
THE PARK, CARRICKMINES,
DUBLIN, IRELAND 18

Correspondent: VORYS, SATER, SEYMOUR AND PEASE LLP
1909 K STREET NW
SUITE 900
WASHINGTON, DC 20006-1152

Search Results as of: 09/01/2023 09:29 PM

If you have any comments or questions concerning the data displayed, contact PRD / Assignments at 571-272-3350. v.2.6
Web interface last modified: Jun 26, 2017 v.2.6

Litigation Search Report CRU 3999

Reexamination Control No. 90/019,243

To: Examiner	From: Renee Preston
Location: CRU	Location: CRU 3999
Art Unit: 3992	Phone: (571) 272-1607
Date: 09/01/2023	E-mail: renee.preston@uspto.gov

Search Notes

U.S. Patent No. 7,624,302

1. Performed a KeyCite Search in Westlaw, which retrieves all history on the patent including any litigation.
2. Performed a search on the patent in Lexis CourtLink for any open dockets or closed cases.
3. Performed a search in Lexis in the Federal Courts and Administrative Materials databases for any cases found.
4. Performed a search in Lexis in the IP Journal and Periodicals database for any articles on the patent.
5. Performed a search in Lexis in the news databases for any articles about the patent or any articles about litigation on this patent.

Litigation: Cases Found

Status	Docket	Description
Open	1:23cv640	Foras Technologies Ltd. V. Nissan Motor Company, Ltd. Et Al
Open	2:23cv150	Foras Technologies Ltd. V. Toyota Motor North America, Inc. Et Al

Citing References (13)

Treatment	Title	Date	Type	Depth	Headnote(s)
—	**1.** BOOT PROCESSOR ROLE SWITCHING METHOD FOR MULTIPROCESSOR SYSTEM, INVOLVES DETERMINING WHETHER PROCESSOR FOR WHICH LOSS OF LOCKSTEP DETECTED IS ASSIGNED ROLE OF BOOT PROCESSOR, AND SWITCHING ROLE OF BOOT TO SPARE PROCESSOR Out Of Plan DWPI 2006-328202	Oct. 25, 2004	DWPI	—	—
—	**2.** RF 058992/0571 Out Of Plan	Feb. 11, 2022	Assignments	—	—
—	**3.** RF 052567/0734 Out Of Plan	May 05, 2020	Assignments	—	—
—	**4.** RF 037079/0001 Out Of Plan	Nov. 09, 2015	Assignments	—	—
—	**6.** RF 015932/0117 Out Of Plan	Oct. 25, 2004	Assignments	—	—
—	**7.** PatStat 7624302	Mar. 09, 2010	Patent Status Files	—	—
—	**8.** INTEGRATED CIRCUIT CHIP WITH CORES ASYMMETRICALLY ORIENTED WITH RESPECT TO EACH OTHER Out Of Plan US PAT 11698841 , U.S. PTO Utility	July 11, 2023	Patents	—	—
—	**9.** INTEGRATED CIRCUIT CHIP WITH CORES ASYMMETRICALLY ORIENTED WITH RESPECT TO EACH OTHER Out Of Plan US PAT 11269742 , U.S. PTO Utility	Mar. 08, 2022	Patents	—	—
—	**10.** INTEGRATED CIRCUIT CHIP WITH CORES ASYMMETRICALLY ORIENTED WITH RESPECT TO EACH OTHER Out Of Plan US PAT 10649865 , U.S. PTO Utility	May 12, 2020	Patents	—	—
—	**11.** INTEGRATED CIRCUIT CHIP WITH CORES ASYMMETRICALLY ORIENTED WITH RESPECT TO EACH OTHER Out Of Plan US PAT 10002056 , U.S. PTO Utility	June 19, 2018	Patents	—	—
—	**12.** STORAGE SYSTEM AND CONTROL METHOD OF STORAGE SYSTEM Out Of Plan US PAT 8312314 , U.S. PTO Utility	Nov. 13, 2012	Patents	—	—
—	**13.** STORAGE SYSTEM AND CONTROL METHOD OF STORAGE SYSTEM Out Of Plan US PAT 7958391 , U.S. PTO Utility	June 07, 2011	Patents	—	—
—	**14.** STORAGE SYSTEM AND CONTROL METHOD OF STORAGE SYSTEM Out Of Plan US PAT 7809981 , U.S. PTO Utility	Oct. 05, 2010	Patents	—	—

WESTLAW © 2024 Thomson Reuters. No claim to original U.S. Government Works.

☐ US7624302B2 - System and method for switching the role of boot processor to a spare processor responsive to detection of loss of lockstep in a boot processor

Original

Bibliographic information & Images

Publication number and date
US7624302B2
2009-11-24
Granted

Application number and date
US10972588 {/api/control?client=&query=AN:{"US10972588"}&autosearch=true}
2004-10-25

Priority number and date
US10972588 {/api/control?client=&query=PRN:{"US10972588"}&autosearch=true}
2004-10-25

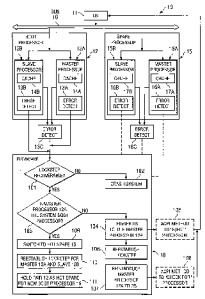
Ultimate owner
FORAS TECHNOLOGIES {/api/control?client=&query=uo:{"FORAS%20TECHNOLOGIES"}&autosearch=true}

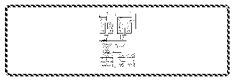
Original assignee
Scott L. Michaelis {/api/control?client=&query=pa:{"Scott%20L.%20Michaelis"}&autosearch=true}

Current assignee
SONRAÍ MEMORY, LTD. {/api/control?client=&query=pacu:{"SONRA%C3%8D%20MEMORY%2C%20LTD."}&autosearch=true}

Inventor
Scott L. Michaelis {/api/control?client=&query=in:{"Scott%20L.%20Michaelis"}&autosearch=true}

1/5 Clipped image













Abstract

According to one embodiment, a method comprises detecting loss of lockstep (LOL) for a processor in a multi-processor system. The method further comprises determining that the processor for which the LOL is detected is assigned the role of boot processor, and switching the role of boot processor to a spare processor without shutting down the system's operating system. In another embodiment, a method comprises system firmware determining that an LOL is detected for a lockstep pair of processors that are assigned the role of boot processor in a system. The method further comprises determining one of the lockstep pair of processors that is not the cause of the LOL, and copying the state of the determined one of the lockstep pair of processors that is not the cause of the LOL to a spare processor. The method further comprises switching the role of boot processor to the spare processor.

Original

Claims

1. A method comprising: detecting loss of lockstep (LOL) for a processor in a multi-processor system; determining that the processor for which said LOL is detected is assigned a role of boot processor; changing a system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected, to a different value, which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor; and changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to a spare processor without shutting down the system's operating system.

2. The method of claim 1 wherein the spare processor is a hot spare that is held in idle until switched the role of boot processor.

3. The method of claim 1 wherein the spare processor is an active spare.

4. The method of claim 3 wherein said switching further comprises: causing the system's operating system to idle the active spare.

5. The method of claim 1 wherein the spare processor is a processor module that includes a master and slave processors that operate in lockstep.

6. The method of claim 1 wherein said determining that the processor for which said LOL is detected is assigned the role of boot processor comprises: accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor.

7. The method of claim 1 further comprising: determining the processor that is assigned the role of spare processor.

8. The method of claim 7 wherein said determining the processor that is assigned the role of spare processor comprises: accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of spare processor.

9. The method of claim 1 wherein said determining step is performed by system firmware.

10. The method of claim 1 wherein said switching step is performed by system firmware.

11. The method of claim 1 wherein the processor for which said LOL is detected includes a lockstep pair of processors, and wherein said switching comprises: determining one of the lockstep pair of processors that is not the cause of said LOL; and copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to said spare processor.

12. The method of claim 11 wherein said spare processor includes a lockstep pair of processors, and wherein said copying comprises: copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to each of said lockstep pair of processors of said spare processor.

13. A method comprising: system firmware determining that loss of lockstep (LOL) is detected for a lockstep pair of processors that are assigned the role of boot processor in a system; determining one of said lockstep pair of processors that is not the cause of said LOL; copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to a spare processor; changing a system identifier, used by the system's operating system to recognize the boot processor, of said lockstep pair of processors that are assigned the role of boot processor, to a different value, which causes the s system's operating system to not recognize the lock step pair of processors for which said LOL is detected as the processor; and changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to said spare processor.

14. The method of claim 13 wherein the spare processor is a hot spare that is held in idle until switched the role of boot processor.

15. The method of claim 13 wherein the spare processor is an active spare.

16. The method of claim 15 further comprising: causing the system's operating system to idle the active spare.

17. The method of claim 13 wherein said switching causes the system's operating system to recognize said spare processor as said boot processor without shutting down said operating system.

18. The method of claim 13 wherein said determining step is performed by system firmware.

19. The method of claim 13 wherein said copying step is performed by system firmware.

20. The method of claim 13 wherein said switching step is performed by system firmware.

21. A system comprising: a plurality of processor modules that each include a master processor and a slave processor that operate in lockstep; an operating system; error detection logic operable to detect loss of lockstep (LOL) for at least one of said processor modules; and system firmware operable, responsive to the detection of LOL for a first of said processor modules, to determine whether said first processor module is assigned a role of system boot processor; wherein if determined that said first processor module is assigned the role of system boot processor, said system firmware is further operable to cause said operating system to recognize another processor module as system boot processor without shutting down said operating system.

22. The system of claim 21 wherein the another processor module is a processor module assigned a role of spare processor.

23. The system of claim 21 wherein the another processor module is held in idle until said system firmware causes said operating system to recognize said another processor module as said system boot processor.

24. The system of claim 21 wherein the system firmware causes the operating system to idle the another processor module before causing said operating system to recognize said another processor module as said system boot processor.

25. The system of claim 21 further comprising: non-volatile data storage storing information that assigns the role of system boot processor to one of said plurality of processor modules.

26. A system comprising: system firmware means for detecting loss of lockstep (LOL) for a processor in a multi-processor system; means for determining whether the processor for which said LOL is detected is assigned the role of boot processor; and system firmware means for switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which said LOL is detected is assigned the role of boot processor.

27. The system of claim 26 wherein said processor in said multi-processor system is an Itanium Processor Family (IPF) processor.

Original

Families

Display in a list (/api/control?client=&familiesSearch=_) INPADOC

This section contains all family relations of this particular document. For more information about the different families, refer to the User Manual (Glossary/Families).

Extended (2) Priority date/claims	Complete (2) Priority date/claims	Main (2) Priority date/claims	Domestic (2) Priority date/claims	Publication number	Application number	Title	Status
2004-10-25 US10972588	2004-10-25 US10972588	2004-10-25 US10972588	2004-10-25	US7624302B2	US10972588	System and method for switching the role of boot processor to a spare processor responsive to detection of loss of lockstep in a boot processor	Granted
				US20060090064A1	US10972588	System and method for switching the role of boot processor to a spare processor responsive to detection of loss of lockstep in a boot processor	Granted

Description

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] The present application is related to the following concurrently filed and commonly assigned U.S. Patent Applications: U.S. patent application Ser. No. 10/973,077 (U.S. Published Patent Application 20060107112) titled "SYSTEM AND METHOD FOR ESTABLISHING A SPARE PROCESSOR FOR RECOVERING FROM LOSS OF LOCKSTEP IN A BOOT PROCESSOR"; U.S. patent application Ser. No. 10/973,004 (U.S. Published Patent Application 20060107117, now U.S. Pat. No. 7,308,566) titled "SYSTEM AND METHOD FOR CONFIGURING LOCKSTEP MODE OF A PROCESSOR MODULE"; U.S. patent application Ser. No, 10/972,835 (U.S. Published Patent Application 20060107114) titled "SYSTEM AND METHOD FOR USING INFORMATION RELATING TO A DETECTED LOSS OF LOCKSTEP FOR DETERMINING A RESPONSIVE ACTION"; U.S. patent application Ser. No. 10/972,588 (U.S. Published Patent Application 20060090064) titled "SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR"; U.S. patent application Ser. No. 10/973,003 (U.S. Published Patent Application 20060107116) titled "SYSTEM AND METHOD FOR REESTABLISHING LOCKSTEP FOR A PROCESSOR MODULE FOR WHICH LOSS OF LOCKSThP IS DETECTED"; U.S. patent application Ser. No. 10/972,888 (U.S. Published Patent Application 20060107115) titled "SYSTEM AND METHOD FOR SYSTEM FIRMWARE CAUSING AN OPERATING SYSTEM TO IDLE A PROCESSOR"; U.S. patent application Ser. No. 10/973,075 (U.S. Published Patent Application 20060107111) titled "SYSTEM AND METHOD FOR REINTRODUCING A PROCESSOR MODULE TO AN OPERATING SYSTEM AFTER LOCKSTEP RECOVERY"; U.S. patent application Ser. No. 10/972,796 (U.S. Published Patent Application 20060107106) titled "SYSTEM AND METHOD FOR MAINTAINING IN A MULTI-PROCESSOR SYSTEM A SPARE PROCESSOR THAT IS IN LOCKSTEP FOR USE IN RECOVERING FROM LOSS OF LOCKSTEP FOR ANOTHER PROCESSOR", the disclosures of which arc hereby incorporated herein by reference.

DESCRIPTION OF RELATED ART

[0002] Silent Data Corruption ("SDC") is a difficult problem in the computing industry. In general, SDC refers to data that is corrupt, but which the system does not detect as being corrupt. SDCs primarily occur due to one of two factors: a) a broken hardware unit or b) a "cosmic" event that causes values to change somewhere in the system. Broken hardware means that a "trusted" piece of hardware is silently giving wrong answers. For example, the arithmetic unit in a processor is instructed to add 1+1 and it returns the incorrect answer 3 instead of the correct answer 2. An example of a cosmic event is when a charged particle (e.g., alpha particle or cosmic ray) strikes a region of a computing system and causes some bits to change value (e.g., from a 0 to a 1 or from a 1 to a 0).

[0003] Numerous techniques have been developed for detecting SDC to prevent the SDC from remaining "silent" or "undetected" within a system, as well as preventing such SDC from propagating through the system. Examples of these techniques include parity-based mechanisms and error correcting codes (ECCs) on buses and memory locations, as well as checksums and/or cyclic redundancy checks (CRC) over regions of memory. Parity-based mechanisms are often employed in processors, wherein a parity bit is associated with each block of data when it is stored. The parity bit is set to one or zero according to whether there is an odd or even number of ones in the data block. When the data block is read out of its storage location, the number of ones in the block is compared with the parity bit. A discrepancy between the values indicates that the data block has been corrupted. ECCs are parity-based mechanisms that track additional information for each data block. The additional information allows the corrupted bit(s) to be identified and corrected.

[0004] Parity/ECC mechanisms have been employed extensively for caches, memories, and similar data storage arrays. In the remaining circuitry on a processor, such as data paths, control logic, execution logic, and registers (the "execution core"), it is more difficult to apply parity/ECC mechanisms for SDC detection. Thus, there is typically some unprotected area on a processor in which data corruption may occur and the parity/ECC mechanisms do not prevent the corrupted data from actually making it out onto the system bus. One approach to SDC detection in an execution core (or other unprotected area of the processor chip) is to employ "lockstep processing." Generally, in lockstep processing two processors are paired together, and the two processors perform exactly the same operations and the results are compared (e.g., with an XOR gate). If there is ever a discrepancy between the results of the lockstep processors, an error is signaled. The odds of two processors experiencing the exact same error at the exact same moment (e.g., due to a cosmic event occurring in both processors at exactly the same time or due to a mechanical failure occurring in each processor at exactly the same time) is nearly zero.

[0005] A pair of lockstep processors may, from time to time, lose their lockstep. "Loss of lockstep" (or "LOL") is used broadly herein to refer to any error in the pair of lockstep processors. One example of LOL is detection of data corruption (e.g., data cache error) in one of the processors by a parity-based mechanism and/or ECC mechanism. Another example of LOL is detection of the output of the paired processors not matching, which is referred to herein as a "lockstep mismatch." It should be recognized that in some cases the data in the cache of a processor may become corrupt (e.g., due to a cosmic event), which once detected (e.g., by a parity-based mechanism or ECC mechanism of the processor) results in LOL. Of course, unless such corrupt data is acted upon by the processor, the output of that processor will not fail to match the output of its paired processor and thus a "lockstep mismatch" will not occur. For example, suppose that a value of "1" is stored to first location of cache in each of a pair of lockstep processors and a value of "1" is also stored to a second location of cache in each of the pair of lockstep processors. Further suppose that a cosmic event occurs for a first one of the processors, resulting in the first location of its cache being changed from "1" to "0", and thus corrupted. This data corruption in the first processor is a LOL for the pair. An error detection mechanism of this first processor may detect the data corruption, thus detecting the LOL. If the processors are instructed to act on the data of their first cache locations, then a lockstep mismatch will occur as the output of each of the processors will not match. For instance, if the processors each add the data stored to the first location of their respective cache with the data stored to the second location of their respective cache, the first processor (having the corrupt data) will output a result of "1" (0+1=1) while the second processor outputs a result of "2" (1+1=2), and thus their respective outputs will not match.

[0006] By employing such techniques as parity-based error detection mechanisms and output comparisons for lockstep paired processors, SDC detection can be enhanced such that practically no SDC occurring in a processor goes undetected (and thus such SDC does not remain "silent") but instead results in detection of LOL. However, the issue then becomes how best for the system to respond to detected LOL. The traditional response to detected LOL has been to crash the system to ensure that the detected error is not propagated through the system. That is, LOL in one pair of lockstep processors in a system halts processing of the system even if other processors that have not encountered an error are present in the system. However, with the increased desire for many systems to maintain high availability, crashing the system each time LOL is detected is not an attractive proposition. This is particularly unattractive for large systems having many processors because cosmic events typically occur more frequently as the processor count goes up, which would result in much more frequent system crashes in those large systems. High availability is a major desire for many customers having large, multi-processor systems, and thus having their system crash every few weeks is not an attractive option. Of course, permitting corrupt data to propagate through the system is also not a viable option.

[0007] Prior solutions attempting to resolve at least some detected SDCs without requiring the system to be crashed have been Operating System ("OS") centric. That is, in certain solutions the OS has been implemented in a manner to recover from a detected LOL without necessarily crashing the system. This OS-centric type of solution requires a lot of processor and platform specific knowledge to be embedded in the OS, and thus requires that the OS provider maintain the OS up-to-date as changes occur in later versions of the processors and platforms in which the OS is to be used. This is such a large burden that most commonly used OSs do not support lockstep recovery.

[0008] Certain solutions have attempted to recover from a LOL without involving the OS in such recovery procedure. For instance, in one technique upon LOL being detected, firmware is used to save the state of one of the processors in a lockstep pair (the processor that is considered "good") to memory, and then both processors of the pair are reset and reinitialized. Thereafter, the state is copied from the memory to each of the processors in the lockstep pair. This technique makes the processors unavailable for an amount of time without the OS having any knowledge regarding this unavailability, and if the amount of time required for recovery is too long, the system may crash. That is, typically, if a processor is unresponsive for X amount of time, the OS will assume that the processor is hung and will crashdump the system so that the problem can be diagnosed. Further, in the event that a processor in the pair cannot be reset and reinitialized (e.g., the processor has a physical problem and fails to pass its self-test), this technique results in crashing the system.

BRIEF SUMMARY OF THE INVENTION

[0009] According to one embodiment, a method comprises detecting loss of lockstep (LOL) for a processor in a multi-processor system. The method further comprises determining that the processor for which the LOL is detected is assigned the role of boot processor, and switching the role of boot processor to a spare processor without shutting down the system's operating system.

[0010] According to one embodiment, a method comprises system firmware determining that loss of lockstep (LOL) is detected for a lockstep pair of processors that are assigned the role of boot processor in a system. The method further comprises determining one of the lockstep pair of processors that is not the cause of the LOL, and copying the state of the determined one of the lockstep pair of processors that is not the cause of the LOL to a spare processor. The method further comprises switching the role of boot processor to the spare processor.

[0011] According to one embodiment, a system comprises a plurality of processor modules that each include a master processor and a slave processor that operate in lockstep. The system further comprises an operating system, and error detection logic operable to detect loss of lockstep (LOL) for at least one of the processor modules. The system further comprises system firmware operable, responsive to detection of LOL for a first of the processor modules, to determine whether the first processor module is assigned a role of system boot processor. If determined that the first processor module is assigned the role of system boot processor, the system

firmware is further operable to cause the operating system to recognize another processor module as system boot processor without shutting down the operating system.

[0012] According to one embodiment, a system comprises means for detecting loss of lockstep (LOL) for a processor in a multi-processor system. The system further comprises means for determining whether the processor for which the LOL is detected is assigned the role of boot processor, and means for switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which the LOL is detected is assigned the role of boot processor.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] FIG. 1 shows an example multi-processor system in which firmware utilizes a spare processor for recovering from LOL for the system's boot processor;

[0014] FIG. 2 shows a block diagram of one embodiment implemented for the IA-64 processor architecture;

[0015] FIG. 3 shows an example system having multi-processor cells in which an embodiment for switching the role of boot processor to a spare processor responsive to detected LOL in the boot processor may be employed;

[0016] FIG. 4 shows an exemplary operational flow diagram for switching the role of boot processor to a hot spare processor according to one embodiment;

[0017] FIG. 5 shows an exemplary operational flow diagram for switching the role of boot processor to an active spare processor according to one embodiment; and

[0018] FIG. 6 shows a more general operational flow diagram for certain embodiments.

DETAILED DESCRIPTION OF THE INVENTION

[0019] As described further herein and in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/973,076 (U.S. Published Patent Appplication 20060107107) titled "SYSTEM AND METHOD FOR PROVIDING FIRMWARE RECOVERABLE LOCKSTEP PROTECTION," the disclosure of which is incorporated herein by reference, certain techniques are provided for recovering from LOL detected for a boot processor in a multi-processor system, in which a spare processor is utilized for such recovery. For instance, upon LOL being detected for the system's boot processor, the spare processor assumes the role of boot processor, and then recovery of lockstep for the original boot processor is attempted. Accordingly, for such lockstep recovery technique, a spare processor is used for attempting to recover from LOL detected for the boot processor. Exemplary techniques are described in concurrently flied and commonly assigned U.S. patent application Ser. No. 10/973,077 (U.S. Published Patent Application 20060107112) titled "SYSTEM AND METHOD FOR ESTABLISHING A SPARE PROCESSOR FOR RECOVERING FROM LOSS OF LOCKSTEP IN A BOOT PROCESSOR," the disclosure of which is incorporated herein by reference, for establishing a spare processor for the boot processor. Such exemplary techniques include techniques for establishing a hot spare processor that is held in reserve until such time as it is used to assume the role of boot processor responsive to a detected LOL in the boot processor. The exemplary techniques further include techniques for establishing an active spare processor that is not held in reserve, but is instead made available to the OS for executing instructions during normal system operation and is dynamically transformed into a hot spare (i.e., it is idled) such that it can assume the role of the boot processor responsive to detection of LOL in the boot processor.

[0020] Embodiments are provided herein for switching the role of boot processor to an established spare processor responsive to detection of LOL in the boot processor. For instance, exemplary actions by system firmware for switching the role of boot processor to an established spare processor and thus causing the OS to recognize the established spare processor as the system's boot processor are provided in accordance with certain embodiments hereof.

[0021] As described further in concurrently filed and commonly assigned U.S. patent application Ser. No. U.S. patent application Ser. No. 10/973,077 (U.S. Published patent application 20060107112) titled "SYSTEM AND METHOD FOR ESTABLISHING A SPARE PROCESSOR FOR RECOVERING FROM LOSS OF LOCKSTEP IN A BOOT PROCESSOR," in certain embodiments, information (referred to herein as a "device tree") is stored to a multi-processor system, wherein such information identifies the processor in the system that is the boot processor and identifies the processor that is the spare for the boot processor. The information is stored such that it is accessible by the system's firmware, and the firmware, upon detecting LOL for the boot processor, determines the spare processor from the device tree and assigns the spare processor the role of boot processor. The firmware then attempts to recover lockstep for the processor for which LOL was detected. If the recovery of lockstep is successful, the processor for which lockstep was recovered (i.e., the former boot processor) may be assigned the role of spare for the current boot processor. Thus, the multi-processor system may recover from LOL detected for the boot processor without requiring a crash of the system (at least for certain types of "recoverable" LOLs).

[0022] In some implementations described herein, the spare processor for the boot processor is a "hot" spare, in which it is idled and not made available to the OS for processing instructions during normal system operation. That is, the hot spare processor is held in reserve, idling until such time as it assumes the role of boot processor responsive to detected LOL for the boot processor. In other implementations, the spare processor is not held in reserve, but is instead an "active" processor that is made available to the OS for executing instructions during normal system operation. Upon LOL being detected for the boot processor, the designated spare processor is dynamically transformed into a hot spare (i.e., it is idled) such that it can assume the role of the boot processor. For instance, upon the system's firmware detecting a LOL for the boot processor, the firmware accesses the device tree and determines the designated spare processor, and then the firmware instructs the OS to idle the designated spare processor (i.e., stop scheduling instructions for the processor) to make the "active" spare a "hot" spare, which then assumes the role of boot processor.

[0023] Concurrently filed and commonly assigned U.S. patent application Ser. No. 10/973,076 (U.S. Published Patent Application 20060107107) titled "SYSTEM AND METHOD FOR PROVIDING FIRMWARE RECOVERABLE LOCKSTEP PROTECTION," discloses a technique in which lockstep can be recovered for processors (particularly, non-boot processors) without requiring use of a spare processor. Rather, upon LOL being detected for a lockstep pair of processors, the firmware instructs the OS to idle (or "eject") the lockstep pair of processors for which LOL was detected and then the firmware attempts to reestablish lockstep (e.g., by resetting the processors). If successful in reestablishing lockstep, the firmware triggers the OS to recognize the lockstep pair of processors as again being available for servicing instructions.

[0024] However, for various reasons, in certain system architectures problems arise in attempting to idle (or eject) the boot processor from the system. Thus, in certain embodiments, a spare processor is used for recovering from LOL for the system's boot processor. Again, embodiments are provided herein for switching the role of boot processor to an established spare processor responsive to detection of LOL for the boot processor. In certain embodiments, the spare processor is used for recovering from LOL only in the case in which LOL is detected for the boot processor. For instances in which LOL is detected for non-boot processors, the lockstep may be recovered without the use of a spare by using the recovery technique described in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/973,076 (U.S. Published Patent Application 20060107107 titled "SYSTEM AND METHOD FOR PROVIDING FIRMWARE RECOVERABLE LOCKSTEP PROTECTION," for example.

[0025] Turning to FIG. 1, an example embodiment of a system **10** in which a spare processor is used in recovering from detected loss of lockstep (LOL) for a system boot processor is shown. System **10** includes OS **11**, as well as master processor **12**A and slave processor **12**B (collectively referred to as a lockstep processor pair **12**). In certain implementations the lockstep processor pair **12** may be implemented on a single silicon chip, which is referred to as a "dual core processor" in which master processor **12**A is a first core and slave processor **12**B is a second core. Further, lockstep processor pair **12** may be referred to as a processor or CPU "module" because it includes a plurality of processors (**12**A and **12**B) in such module. As described further herein, to certain portions of system **10**, such as its OS **11**, the processor module **12** may appear as a single processor because slave processor **12**B is merely used for redundant processing for checking the output of master processor **12**A. In this example, lockstep processor pair **12** is designated as the boot processor for system **10**. As described further herein, in certain embodiments lockstep processor pair **12** is designated as the system boot processor by storing information to the system that specifies such designation.

[0026] Master processor **12**A includes cache **14**A, and slave processor **12**B includes cache **14**B. OS **11** and lockstep processor pair **12** are communicatively coupled to bus **16**. Typically, master processor **12**A and slave processor **12**B are coupled to bus **16** via an interface that allows each of such processors to receive the same instructions to process, but such interface only communicates the output of master processor **12**A back onto bus **16**. The output of slave processor **12**B is used solely for checking the output of mater processor **12**A.

[0027] In this example, master processor **12**A includes error detect logic **13**A, and slave processor **12**B includes error detect logic **13**B. While shown as included in each of the processors **12**A and **12**B in this example, in certain embodiments the error detect logic **13**A and **13**B may be implemented external to processors **12**A and **12**B. Error detect logic **13**A and **13**B include logic for detecting errors, such as data cache errors, present in their respective processors **12**A and **12**B. Examples of error detect logic **13**A and **13**B include known parity-based mechanisms and ECC mechanisms. Error detect logic **13**C is also included, which may include an XOR (exclusive OR) gate, for detecting a lockstep mismatch between master processor **12**A and slave processor **12**B. As mentioned above, a lockstep mismatch refers to the output of master processor **12**A and slave processor **12**B failing to match. While shown as external to the lockstep processor pair **12** in this example, in certain embodiments error detect logic **13**C may be implemented on a common silicon chip with processors **12**A and **12**B.

[0028] Lockstep mismatch is one way of detecting a LOL between the master processor **12**A and slave processor **12**B. A detection of an error by either of error detect logic **13**A and **13**B also provides detection of LOL in the processors **12**A and **12**B. Because the detection of LOL by error detect logic **13**A and **13**B may occur before an actual lockstep mismatch occurs, the detection of LOL by error detect logic **13**A and **13**B may be referred to as a detection of a "precursor to lockstep mismatch". In other words, once an error (e.g., corrupt data) is detected by error detect logic **13**A or **13**B, such error may eventually propagate to a lockstep mismatch error that is detectable by error detect logic **13**C.

[0029] System **10** further includes a second lockstep processor pair **15** that is communicatively coupled to bus **16**. Lockstep processor pair **15** includes master processor **15**A and slave processor **15**B. Master processor **15**A includes cache **17**A, and slave processor **15**B includes cache **17**B. As with error detect logic **13**A-**13**C implemented for lockstep processor pair **12**, lockstep processor pair **15** has error detect logic **16**A-**16**C. While only two lockstep processor pairs **12** and **15** are shown for simplicity in the example of FIG. 1, system **10** may include any number of such lockstep processor pairs. As one specific example, in one implementation system **10** includes **64** lockstep processor pairs, wherein the master processors of the pairs perform parallel processing for the system. In the specific example of FIG. 1, lockstep processor pair **15** is designated as a spare for boot processor **12**, as described further herein.

[0030] In certain embodiments described herein, techniques are provided for switching the role of boot processor to the designated spare processor responsive to detection of LOL for the boot processor. Thus, the spare processor is used for recovery when LOL is detected for the system's boot processor. That is, the designated spare processor assumes the role of boot processor in the event that LOL is detected for the boot processor. In accordance with certain embodiments, upon LOL being detected for a boot processor, a hot spare processor (i.e., an idling processor that is available in the system) is transferred the role of boot processor, and then the old boot processor having LOL is reset to reestablish its lockstep. In other embodiments, a processor is designated as a spare processor for the boot processor, but such designated spare processor is not held in reserve (idled) during normal system operation. Rather, the designated spare is utilized as an active processing resource made available to OS **11** for processing instructions that OS **11** schedules for such "active" spare processor. Upon LOL being detected for the boot processor, the designated spare processor is idled (thereby, transforming such active processor into a hot spare), and is transferred the role of boot processor. Thereafter, the former boot processor for which the LOL was detected may be reset to reestablish its lockstep. Thus, in certain embodiments, a designated spare processor is not required to be held in reserve, but is instead an available resource for use within the system and is only transformed into an actual spare processor when needed for recovering from LOL for the boot processor.

[0031] Firmware **18** is also included in system **10**, which in this embodiment is invoked upon an error being detected by any of the error detect logics **13**A-**13**C and **16**A-**16**C. In certain embodiments, processors **12**A, **12**B, **15**A, and **15**B are processors from the Itanium Processor Family (IPF). IPF is a 64-bit processor architecture co-developed by Hewlett-Packard Company and Intel Corporation, which is based on Explicitly Parallel Instruction Computing (EPIC). IPF is a well-known family of processors. IPF includes processors such as those having the code names of MERCED, MCKINLEY, and MADISON. In addition to supporting a 64-bit processor bus and a set of 128 registers, the 64-bit design of IPF allows access to a very large memory (VLM) and exploits features in EPIC. While a specific example implementation of one embodiment is described below for the IPF architecture, embodiments described herein for establishing a spare processor for recovering from LOL detected for a boot processor are not limited in application to an IPF architecture, but may be applied as well to other architectures (e.g., 32-bit processor architectures, etc.).

[0032] Processor architecture generally comprises corresponding supporting firmware, such as firmware **18** of system **10**. For example, as described further below in conjunction with the specific example of FIG. 2, the IPF processor architecture comprises such supporting firmware as Processor Abstraction Layer (PAL), System Abstraction Layer (SAL), and Extended Firmware Interface (EFI). Such supporting firmware may enable, for example, the OS to access a particular function implemented for the processor. For instance, the OS may query the PAL as to the size of the cache implemented for the processor, etc. Other well-known functions provided by the supporting firmware (SAL, EFI) include, for example: (a) performing I/O configuration accesses to discover and program the I/O Hardware (SAL_PCI_CONFIG_READ and SAL_PCI_CONFIG_WRITE); (b) retrieving error log data from the platform following a Machine Check Abort (MCA) event (SAL_GET_STATE_INFO); (c) accessing persistent store configuration data stored in non-volatile memory (EFI variable services: GetNextVariableName, GetVariable and SetVariable); and accessing the battery-backed real-time clock/calendar (EFI GetTime and SetTime). Accordingly, the supporting firmware, such as the PAL, is implemented to provide an interface to the processor(s) for accessing the functionality provided by such processor(s). Each of those interfaces provide standard, published procedure calls that are supported.

[0033] In the example system **10** of FIG. 1, firmware **18** utilizes the above-mentioned hot spare technique for recovering from LOL for a boot processor. That is, processor **15** is designated as a hot spare for boot processor **12**, and such processor **15** is held as an idle, spare processor and used in the manner described further herein for assuming the role of boot processor in the event that a LOL is detected for boot processor **12**. Of course, as described further herein, the above-mentioned technique in which a designated spare is not held in reserve but is instead dynamically transformed into a spare processor may be implemented by firmware **18** in alternative embodiments. As described further herein, in certain embodiments, a device tree is stored to a data storage device (e.g., SRAM) such that it is accessible to the firmware **18**, wherein the device tree includes information identifying processor **12** as the system's boot processor and identifying processor **15** as the spare. The information specifying the boot processor and/or spare may be received as input from a user in certain embodiments. For instance, in certain embodiments a user interface is provided that enables a system administrator to designate the boot processor and/or spare processor in the multi-processor system. In other embodiments, the firmware **18** selects the boot processor and/or spare.

[0034] Suppose that during the system boot-up procedure, master processor **12**A assumes the role of boot processor, as shown in the example of FIG. 1. In this example embodiment, lockstep processor pair **15** is held as a hot spare for recovering from a LOL that may be detected for the boot processor **12**A. Of course, additional lockstep processor pairs may be included in system **10** (not specifically shown in the example of FIG. 1), and those additional lockstep processor pairs, which are not the system boot processor, may recover from LOL in a different manner that does not utilize a hot spare processor. Thus, in accordance with certain embodiments hereof, hot spare processor pair **15** is not needed for recovering from LOL detected for any non-boot processor, but may instead be used for recovery of LOL only if such LOL is detected for the boot processor (processor **12**A in this example).

[0035] Firmware **18** is included in this example, and upon detection of LOL by any of error detect logics **13**A-**13**C, firmware **18** determines, in operational block **101**, whether the detected LOL is a recoverable LOL. That is, firmware **18** determines in block **101** whether the detected LOL is of a type from which the firmware can recover lockstep for the lockstep processor pair **12** without crashing the system. As described further herein, lockstep is recoverable for certain detected LOLs (which may be referred to as "recoverable LOLs"), while lockstep is not recoverable for other detected LOLs (which may be referred to as "non-recoverable LOLs"). If the lockstep is not recoverable from the detected LOL, then in the example of FIG. 1 firmware **18** crashes the system in block **102**.

[0036] If the lockstep is recoverable, then operation advances to block **103** whereat the firmware determines whether the processor for which the LOL was detected is the system's boot processor. Because the LOL is detected for the lockstep processor pair **12** in this example, firmware **18** determines whether master processor **12**A is the system's boot processor. This can be determined, for example, by accessing the device tree **203** of FIG. 2 (described further below). The device tree may, in certain implementations, include a field for each processor indicating whether such processor is the system's boot processor. This field in the device tree may be set by the firmware during the boot-up process to identify the corresponding processor that is used as the system's boot processor or it may be pre-set by a system administrator (e.g., via a user interface), as examples. If determined in block **103** that the master processor **12**A is not the system's boot processor, then operation advances in this example to block **104**, and a lockstep recovery process is performed that does not utilize hot spare **15**.

[0037] In the specific example shown in FIG. 1, if the lockstep is determined to be recoverable in block **101** and the processor for which the LOL is detected is determined in block **103** to not be the system boot processor, firmware **18** cooperates with OS **11** via standard OS methods to recover the lockstep. For instance, in the example embodiment of FIG. 1, Advanced Configuration and Power Interface (ACPI) methods are used by firmware **18** to cooperate with OS **11**. Accordingly, no processor or platform specific knowledge is required to be embedded in OS **11**, but instead any ACPI-compatible OS may be used, including without limitation HP-UX and Open VMS operating systems. Example techniques that may be used for recovering from LOL for non-boot processors of a system are described further in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/973,076 titled "SYSTEM AND METHOD FOR PROVIDING FIRMWARE RECOVERABLE LOCKSTEP PROTECTION," the disclosure of which is incorporated herein by reference.

[0038] In the example embodiment of FIG. 1, if determined in block **103** that the processor for which the LOL is detected is not the system boot processor, firmware **18** triggers OS **11** to idle the master processor **12**A in operational block **104**. In this embodiment, firmware **18** utilizes an ACPI method **105** to "eject" master processor **12**A, thereby triggering OS **11** to idle the master processor **12**A (i.e., stop scheduling tasks for the processor). Of course, by idling master processor **12**A, slave processor **12**B will in turn be idled. Thus, idling master processor **12**A results in idling the lockstep processor pair **12**. In this example embodiment, OS **11** is not aware of the presence of slave processor **12**B, but is instead aware of master processor **12**A. The interface of lockstep processor pair **12** to bus **16** manages copying to slave processor **12**B the instructions that are directed by OS **11** to master processor **12**A. Thus, firmware **18** need not direct OS **11** to eject slave processor **12**B, as OS **11** is not aware of such slave processor **12**B in this example implementation. Again, by idling master processor **12**A, slave processor **12**B is also idled as it merely receives copies of the instructions directed to master processor **12**A. Of course, if in a given implementation OS **11** is aware of slave **12**B, firmware **18** may be implemented to also direct OS **11** to idle such slave processor **12**B in a manner similar to that described for idling master processor **12**A.

[0039] Firmware **18** then attempts to recover lockstep for the lockstep processor pair **12** in block **106**. For instance, firmware **18** resets the processor pair **12**. During such reset of processor pair **12**, system **10** can continue to operate on its remaining available processors. Once the processor pair **12** is reset and lockstep is recovered, firmware **18** reintroduces master processor **12**A to OS **11** in operational block **107**. In this embodiment, firmware **18** updates the ACPI device table information for master processor **12**A to indicate that such master processor **12**A is "present, functioning and enabled." As discussed in the ACPI 2.0 specification for the _STA status method of a device, the _STA (status) object returns the status of a device, which can be one of the following: enabled, disabled, or removed. In this respect, in the result code (bitmap) bit **0** is set if the device is present; bit **1** is set if the device is enabled and decoding its resources; bit **2** is set if the device should be shown in the UI; bit **3** is set if the device is functioning properly (cleared if the device failed its diagnostics); bit **4** is set if the battery is present; and bits **5**-**31** are reserved. A device can only decode its hardware resources if both bits **0** and **1** are set. If the device is not present (bit **0** cleared) or not enabled (bit **1** cleared), then the device must not decode its resources. Bits **0**, **1** and **3** are the "present, enabled and functioning" bits mentioned above. Firmware **18** utilizes an ACPI method **108** to trigger OS **11** to "check for" master processor **12**A, thereby reintroducing the master processor **12**A to OS **11**. As a result of checking for master processor **12**A, OS **11** will recognize that such master processor **12**A is again available and will thus begin scheduling tasks for master processor **12**A once again.

[0040] In view of the above, a recovery technique is provided that does not require utilizing a spare processor for recovering from LOL detected for non-boot processors. For various reasons, in certain system architectures problems arise in attempting to idle (or eject) the boot processor from the system. Thus, in certain embodiments, a hot spare processor is used for recovering from LOL for the system's boot processor. For instance, in the example of FIG. 1, if determined in block **103** that the master processor **12**A is the system's boot processor, operation advances to block **109** whereat the state of the "good" processor in the lockstep processor pair **12** is copied over to each of processors **15**A and **15**B in the spare lockstep processor pair **15**. Exemplary techniques that may be utilized by firmware **18** for switching the role of boot processor to the hot spare are described further herein below.

[0041] Thus, operational block **109** essentially makes the spare processor pair **15** the system's boot processor, and then in certain embodiments firmware **18** resets the lockstep processor pair **12** and reestablishes its lockstep in block **110**. Once lockstep is reestablished for lockstep processor pair **12**, that pair is held in operational block **111** as a hot spare for the new boot processor pair **15**. That is, firmware **18** updates the device tree **203** of FIG. 2 to reflect that lockstep processor pair **12** is a hot spare for the boot processor pair **15**. Thus, should a recoverable LOL be detected for the new boot processor **15**, the above process may be used by firmware **18** to make the spare lockstep processor pair **12** the boot processor and then recover lockstep for pair **15**. An example technique for resetting the lockstep processor pair **12** and reestablishing its lockstep in which the lockstep processor pair **12** is then held as a hot spare is described further in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/972,796 (U.S. Published Patent Application 20060107106) titled "SYSTEM AND METHOD FOR MAINTAINING IN A MULTI-PROCESSOR SYSTEM A SPARE PROCESSOR THAT IS IN LOCKSTEP FOR USE IN RECOVERING FROM LOSS OF LOCKSTEP FOR ANOTHER PROCESSOR", the disclosure of which is hereby incorporated herein by reference.

[0042] FIG. 2 shows a block diagram of one embodiment of the above system **10**, which is implemented for the IPF processor architecture and is labeled as system $\mathbf{10}_A$. The quintessential model of the traditional IPF architecture is given in the *Intel IA-64 Architecture Software Developer's Manual, Volume 2: IA-64 System Architecture*, in section 11.1 *Firmware Model*, the disclosure of which is hereby incorporated herein by reference. Accordingly, in this example embodiment of system $\mathbf{10}_A$, firmware **18**, labeled as firmware $\mathbf{18}_A$, includes processor abstraction layer (PAL) **201** and platform/system abstraction layer (SAL) **202**. In general, PAL **201** is firmware provided by Intel for its processors, and SAL **202** is developed by an original equipment manufacturer (OEM) for the specific system/platform in which the processors are to be employed. PAL **201**, SAL **202**, as well as an extended firmware interface (EFI) layer (not shown), together provide, among other things, the processor and system initialization for an OS boot in an IPF system.

[0043] The boot-up process of a traditional IPF system, for example, proceeds as follows: When the system is first powered on, there are some sanity checks (e.g., power on self-test) that are performed by microprocessors included in the system platform, which are not the main system processors that run applications. After those checks have passed, power and clocks are given to a boot processor (which may, for example, be master processor **12**A). The boot processor begins executing code out of the system's Read-Only Memory (ROM) (not specifically shown in FIG. 2). The code that executes is the PAL **201**, which gets control of system **10**. PAL **201** executes to acquire all of the processors in system $\mathbf{10}_A$ (recall that there may be many lockstep processor pairs, such as pairs **12** and **15**) such that the processors begin executing concurrently through the same firmware.

[0044] After it has performed its duty of initializing the processor(s), PAL **201** passes control of system $\mathbf{10}_A$ to SAL **202**. It is the responsibility of SAL **202** to discover what hardware is present on the system platform, and initialize it to make it available for the OS **11**. When main memory is initialized and functional, the firmware $\mathbf{18}_A$ is copied into the main memory. Then, control is passed to EFI (not shown), which is responsible for activating boot devices, which typically includes the disk. The EFI reads the disk to load a program into memory, typically referred to as an operating system loader. The EFI loads the OS loader into memory, and then passes it control of system $\mathbf{10}_A$ by branching the boot processor into the entry point of such OS loader program.

[0045] The OS loader program then uses the standard firmware interfaces to discover and initialize system $\mathbf{10}_A$ further for control. One of the things that the OS loader typically has to do in a multi-processor system is to retrieve control of the other processors (those processors other than the boot processor). For instance, at this point in a multi-processor system, the other processors may be executing in do-nothing loops. In an ACPI-compatible system, OS **11** makes ACPI calls to parse the ACPI tables to discover the other processors of a multi-processor system in a manner as is well-known in the art. Then OS **11** uses the firmware interfaces to cause those discovered processors to branch into the operating system code. At that point, OS **11** controls all of the processors and the firmware $\mathbf{18}_A$ is no longer in control of system $\mathbf{10}_A$.

[0046] As OS **11** is initializing, it has to discover from the firmware $\mathbf{18}_A$ what hardware is present at boot time. And in the ACPI standards, it also discovers what hardware is present or added or removed at run-time. Further, the supporting firmware (PAL, SAL, and EFI) are also used during system runtime to support the processor. For example, OS **11** may access a particular function of master processor **12**A via the supporting firmware $\mathbf{18}_A$, such as querying PAL **201** for the number, size, etc., of the processor's cache **14**A. Some other well-known firmware functions that OS **11** may employ during runtime include: (a) PAL **201** may be invoked to configure or change processor features such as disabling transaction queuing (PAL_BUS_SET_FEATURES); (b) PAL **201** may be invoked to flush processor caches (PAL_CACHE_FLUSH); (c) SAL **202** may be invoked to retrieve error logs following a system error (SAL_GET_STATE_INFO, SAL_CLEAR_STATE_INFO); (d) SAL **202** may be invoked as part of hot-plug sequences in which new I/O cards are installed into the hardware (SAL_PCI_CONFIG_READ, SAL_PCI_CONFIG_WRIT); (e) EFI may be invoked to change the boot device path for the next time the system reboots (SetVariable); (f) EFI may be invoked to change the clock/calendar hardware settings; and (g) EFI may be invoked to shutdown the system (ResetSystem).

[0047] A "device tree" is provided, which is shown as device tree **203** in this example. Device tree **203** is stored in SRAM (Scratch RAM) on the cell, which is RAM that is reinitialized. Firmware **18**A builds the device tree **203** as it discovers what hardware is installed in the system. Firmware then converts this information to the ACPI tables format and presents it to OS **11** so that OS **11** can know what is installed in the system. The ACPI device tables (not shown) are only consumed by OS **11** at boot time, so they are never updated as things change. For OS **11** to find the current status, it calls an ACPI "method" to discover the "current status". The _STA method described above is an example of such an ACPI method. When _STA is called, the AML can look for properties on the device specified in the firmware device tree and convert that into the Result Code bitmap described above. So, if lockstep has been lost on a processor, firmware **15**A will set the device tree property that indicates loss of lockstep, then when OS **11** calls _STA for that device, the "lockstep lost" property directs the AML code to return to "0" in the "functioning properly" bit so that OS **11** can know there is a problem with that processor.

[0048] If a lost lockstep is recovered in accordance with the recovery technique described herein, firmware **18**A can indicate that lockstep has been recovered in the device tree **203**. Then when _STA is called on that device responsive to the OS receiving the "check for device" ACPI method, the present, enabled and functioning bits will all be set and OS **11** will know the CPU is safe to use.

[0049] Further, device tree **203** includes information designating which processor is the boot processor, as well as a spare processor for the boot processor. A simple example of device tree **203** is shown below in Table 1:

[0050]

TABLE 1

Device	Status	Lockstep Enabled	Boot Processor?	Spare for Boot Processor?
Processor A	Present, Enabled, and Functioning	Yes	Yes	No
Processor B	N/A	Yes	No	Yes
Processor C	Present, Enabled, and Functioning	Yes	No	No
. . .	. . .	. . .	. . .	. . .

[0051] In the above example of Table 1, 3 processors "Processor A," "Processor B," and "Processor C" are identified in the device tree as being included in the multi-processor system. The status of each of processors Processor A and Processor B is identified as "Present, Enabled, and Functioning." In this example Processor B is a hot spare, so it is held in reserve and does not show up in the static ACPI tables that are built when the system boots. Accordingly the OS does not know about Processor B and does not have a device handle to call the _STA method on. As described above, the _STA method is a known ACPI method for checking the status of the corresponding device. Therefore, the status of Processor B is designated N/A (not applicable) in this example because the _STA method will not be called for this processor as long as it is the hot spare. Of course, upon such spare assuming the role of boot processor, its status is updated to "Present, Enabled, and Functioning." In other embodiments in which Processor B is an active spare, the OS will be aware of such processor and accordingly its status may be also indicated as "Present, Enabled, and Functioning."

[0052] Further, lockstep is identified in this example as enabled for these processors. In certain embodiments, the system administrator can designate, via user interface **20**, whether lockstep mode is to be enabled for the system's processors. Further, Processor A is designated as the system's boot processor in the example device tree shown in Table 1, and Processor B is designated as the spare processor. Thus, upon detection of LOL for one of the processors in the system, the firmware can access this device tree to determine if the processor for which LOL is detected is the system's boot processor; and if determined that the processor for which LOL is detected is the system's boot processor, the firmware can further determine from this device tree the designated spare processor to use in the lockstep recovery process.

[0053] In the example of FIG. 2, a user interface **20** is included, which allows a user (e.g., system administrator) to designate the system's boot processor and spare processor. In certain embodiments, the user interface **20** further allows the user to designate other information, such as whether lockstep mode is enabled for the processors.

[0054] One example system in which firmware 18_A of FIG. 2 may be employed is shown in FIG. 3. FIG. 3 shows system **30** that includes multi-processor cells A-D, labeled 31_A-31_D, respectively. Cell A 31_A is shown in more detail, and should be understood that cells B-D 31_B-31_D have substantially the same architecture as that of cell A 31_A in this example. The architecture of cell A 31_A described hereafter corresponds to that of Hewlett-Packard's Superdome™ systems. As shown, multi-processor cell A 31_A includes coherency controller **304** that is communicatively coupled to two buses, labeled **305** and **306**, respectively. A plurality of processor modules are included within cell A 31_A, shown as CPU **0**, CPU **1**, CPU **2** and CPU **3** (and labeled **300**-**303**, respectively). More particularly, a plurality of processors modules are communicatively coupled to each of the buses **305** and **306**. In this example, CPU **0** (**300**) and CPU **1** (**301**) are each communicatively coupled to the first bus **305**, and CPU **2** (**302**) and CPU **3** (**303**) are each communicatively coupled to the second bus **306**. It should be understood that each of CPU **0** (**300**), CPU **1** (**301**), CPU **2** (**302**), and CPU **3** (**303**) is in actuality a lockstep processor pair, such as the lockstep processor pair **12** of FIGS. 1-3. Thus, for instance, CPU **0** (**300**) includes both a master processor and a slave processor, as with processors **12**A and **12**B included in the lockstep processor pair **12** of FIGS. 1-2.

[0055] Coherency controller **304** decodes the address an interrupt is targeted towards and determines which bus (**305** or **306**) the interrupt packet should be delivered to. Each CPU on the destination bus sees the interrupt and compares the interrupt target address with its internal "LID" register to determine if it is the targeted CPU. If the match is correct, the CPU responds to the interrupt. If the match is not correct, the CPU ignores the interrupt packet.

[0056] As shown, the example firmware 18_A described above with FIGS. 1-2 may be employed to manage the recovery from detected LOLs for the processors of cells A-D (31_A-31_D). In this example architecture, when designating a spare for the boot processor, such spare is selected to be another processor communicatively coupled to the same bus as the boot processor. For instance, suppose that during the boot-up of system **30**, CPU **2** (**302**) is designated as the boot processor. Accordingly, firmware 18^A will identify another processor that is communicatively coupled to bus **306** to designate as a spare for such boot processor, such as CPU **3** (**303**). During runtime if a LOL is detected for CPU **2** (**302**), then firmware 18_A may operate according to the process described above in FIG. 1 to switch the designation of boot processor to a designated spare, CPU **3** (**303**). In performing this switch the LID register of the CPU that lost lockstep is copied into the LID register of the spare CPU and the spare CPU pair will then assume the ID of the failed CPU and start responding to interrupts directed to that ID. The LID of the failed CPU is quickly re-written to a different value so that it will no longer respond to interrupts directed to that ID. CPU **2** (**302**) can then be reset to recover its lockstep, and such CPU **2** (**302**) may then be designated as a spare for the new boot processor CPU **3** (**303**).

[0057] It should be recognized that in the example system of FIG. 3, if a hot spare were used for recovering from LOL for all processors, rather than just for the boot processor, much of the potential processing resources would be required to be held in reserve as spare resources. For instance, a hot spare would be required on each of the buses of each cell. Because there are two processors coupled to each bus in this example (i.e., CPU **0** and CPU **1** coupled to bus **305**, and CPU **2** and CPU **3** coupled to bus **306**), reserving a hot spare processor for each bus would result in holding half of the total processing resources in reserve. This is an undesirably expensive and wasteful solution. Thus, according to the illustrated embodiment, a hot spare processor is reserved only for the boot processor. Thus, continuing with the above example in which CPU **2** is the boot processor and CPU **3** is held as a "hot" spare, both of CPU **0** and CPU **1** are active (i.e., neither are held in spare). Further, none of the processors of cells B-D need to be held in reserve as hot spares. In the event of a recoverable LOL occurring for any of non-boot processors of the cells, the lockstep recovery process of FIG. 1 (i.e., operational blocks **104**-**108**) can be utilized without requiring a spare for lockstep recovery for those non-boot processors.

[0058] While in the above example embodiment, a hot spare CPU is maintained for recovery from LOL for the system's boot processor, in an alternative embodiment a hot spare need not be held as such, but rather a processor can be dynamically made a "spare" when needed for recovery from LOL encountered for the system boot processor. That is, assuming as in the above example of FIG. 3 that CPU **2** is the system's boot processor, CPU **3** does not have to be held in reserve (idled) as a hot spare for CPU **2**, but instead CPU **3** can be active and dynamically idled and made a "spare" upon detection of LOL in CPU **2**. Accordingly, CPU **3** may be designated as a spare for the boot processor, but such CPU **3** need not be held in reserve during normal system operation. Instead, the CPU **3** may be dynamically (during system run-time) transformed into a spare processor (e.g., by causing the OS to idle such processor) in the event that a LOL is detected for the boot processor.

[0059] Thus, the resource of this spare CPU need not be wasted during normal runtime, but can be dynamically turned into a spare processor that is available for recovering from LOL encountered on CPU **2** in the manner described above. As an example of this embodiment, the firmware may send an "eject request" on a healthy processor pair (e.g., CPU **3**), and lie to the OS by indicating (e.g., in response to an ACPI _STA method for the healthy processor pair) that such processor pair is not functioning. This would cause the OS to eject that processor pair as though it had encountered a LOL. This processor would become the "hot spare" and be used to replace the boot processor. The system boot processor role would be assumed by the idled healthy processor pair, just as described above in the example in which the healthy CPU **3** is held idle for the system boot processor, CPU **2**. The only difference being that now the firmware had to take some initial action to turn the healthy processor pair into a "hot spare" by idling it and making it appear to the OS as though it was not functioning. Thus, the LIDs would be swapped. That is, the LID register of the boot processor that lost lockstep is copied into the LID register of the now spare processor pair (CPU **3** in this example), and the LID register of this spare processor pair would be copied into the LID register of the original boot processor. Lockstep is then reestablished on the original boot processor, and it can be reintroduced to the OS with the ID of the original non-boot processor that replaced it as the system boot processor. The processor that was formerly the boot processor may be designated, in device tree **203**, as the spare for the now boot processor.

[0060] Turning to FIG. 4, an exemplary operational flow diagram for switching the role of boot processor to a hot spare processor according to one embodiment is shown. In this case, an LOL has been detected for the boot processor. In operational block **401**, the system firmware allocates space to save the status of the good processor of the lockstep pair of processors for which LOL was detected. Exemplary techniques for determining which of the lockstep pair of processors had the error are disclosed in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/972,835 (U.S. Published Patent Application 20060107114) titled "SYSTEM AND METHOD FOR USING INFORMATION RELATING TO A DETECTED LOSS OF LOCKSTEP FOR DETERMINING A RESPONSIVE ACTION," the disclosure of which is hereby incorporated herein by reference. Accordingly, an appropriate amount of data storage space for storing the cache of the "good" processor in the lockstep pair of processors is allocated in block **401**. For instance, an amount of SRAM may be allocated for storing the registers from the good processor.

[0061] In operational block **402**, the state of the good processor is stored to the allocated storage space. For instance, in this illustrative example, PAL is called to dump the processor state of the good processor of the current boot processor (CBP) module (for which LOL was detected) to the allocated storage space.

[0062] In operational block **403**, the LID of the current boot processor (for which LOL was detected) is reprogrammed to a different value. For instance, the current boot processor may be assigned a LID that is reserved just for this situation (i.e., for switching the role of boot processor). Accordingly, the boot processor for which LOL was detected will not respond to the interrupts addressed for its original LID because the boot processor's LID value is reprogrammed to a different value.

[0063] In operational block **404**, the state of the good processor that was stored in block **402** to the allocated storage space is stored to the hot spare processor. In this example, the PAL is called to "Restore Processor State" on the hot spare processor, wherein the PAL is provided a pointer to the address to which the good processor's state was stored in block **402**. Accordingly, all the values from that storage space will be read into the registers of the spare processor. As a result, the LID value of the spare processor is programmed with the value of the boot processor that was saved to the allocated storage space. Accordingly, the spare processor automatically, as a result of receiving this stored state, receives the Local ID corresponding to the boot processor for interrupt handling.

[0064] In operational block **405**, the system firmware updates its device tree information to reflect that the hot spare is assuming the role of boot processor. The system firmware further updates its device tree, in this example, to indicate that the former boot processor (for which LOL was detected) is assuming the role of hot spare for the new boot processor.

[0065] In operational block **406**, PAL_MC_RESUME (which is a standard PAL call for resuming normal execution after a machine check (MC)) is called for the hot spare (which is assuming the role of boot processor) to cause the OS to recognize the hot spare processor as available for processing instructions, wherein the OS recognizes the hot spare as the boot processor. Thus, the hot spare has now assumed the role of the boot processor, taking the place of the former boot processor for which LOL was detected. Accordingly, in operational block **407** the former hot spare (which has assumed the role of boot processor) resumes normal operation as the system's boot processor with its lockstep protection enabled. Thus, in this exemplary embodiment, upon detection of LOL for a boot processor, the role of boot processor is switched to a hot spare that has lockstep protection, wherein the system effectively recovers lockstep for its boot processor without requiring a system shut down (i.e., without shutting down the OS). In certain implementations, the firmware may go on to take action to attempt to reestablish lockstep for the former boot processor (e.g., by resetting the former boot processor), and if successful in reestablishing its lockstep, such former boot processor may be held in reserve as a hot spare processor for the new boot processor in the event that LOL is detected for this new boot processor.

[0066] Turning to FIG. 5, an exemplary operational flow diagram for switching the role of boot processor to an active spare processor according to one embodiment is shown. Again, an LOL has been detected for the boot processor. In operational block **501**, the system firmware, responsive to such detection of LOL for the boot processor, examines the device tree to determine which processor module is assigned the role of active spare. In operational block **502** the system firmware (e.g., SAL) asserts a General Purpose Event Interrupt (GPE) by writing to the appropriate GPE register in the ACPI register space. This causes the OS to execute its interrupt handler, which will run. an AML Method associated with this particular interrupt. In this example, this AML method that is executed responsive to the interrupt generated by the GPE includes an AML instruction called "Notify". This "Notify" operation indicates the determined active spare processor object and that an Eject is being requested as arguments. Accordingly, an AML method is used to request the OS to eject the active spare that was determined in block **501**. In response, the OS idles the active spare processor and returns control over the active spare processor to the system firmware, in block **503**.

[0067] In operational block **504**, the system firmware allocates space to save the status of the good processor of the lockstep pair of processors for which LOL was detected (i.e., the lockstep pair of processors of the CBP). Exemplary techniques for determining which of the lockstep pair of processors had the error are disclosed in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/972,835 (U.S. Published Patent Application No. 20060107114) titled "SYSTEM AND METHOD FOR USING INFORMATION RELATING TO A DETECTED LOSS OF LOCKSTEP FOR DETERMINING A RESPONSIVE ACTION," the disclosure of which is hereby incorporated herein by reference. Accordingly, an appropriate amount of data storage space for storing the cache of the "good" processor in the lockstep pair of processors is allocated in block **504**. For instance, an amount of SRAM may be allocated for storing the registers from the good processor.

[0068] In operational block **505**, the state of the good processor is stored to the allocated storage space. For instance, in this illustrative example, PAL is called to dump the processor state of the good processor of the current boot processor (CBP) module (for which LOL was detected) to the allocated storage space.

[0069] In operational block **506**, the LID of the current boot processor (for which LOL was detected) is reprogrammed to a different value. For instance, the current boot processor may be assigned a LID that is reserved just for this situation (i.e., for switching the role of boot processor). Accordingly, the boot processor for which LOL was detected will not respond to the interrupts addressed for its original LID because the boot processor's LID value is reprogrammed to a different value.

[0070] In operational block **507**, the state of the good processor that was stored in block **505** to the allocated storage space is stored to the idled spare processor. In this example, the PAL is called to "Restore Processor State" on the active spare processor, wherein the PAL is provided a pointer to the address to which the good processor's state was stored in block **505**. Accordingly, all the values from that storage space will be read into the registers of the spare processor. As a result, the LID value of the spare processor is programmed with the value of the boot processor that was saved to the allocated storage space. Accordingly, the spare processor automatically, as a result of receiving this stored state, receives the Local ID (LID) corresponding to the boot processor for interrupt handling.

[0071] In operational block **508**, the system firmware updates its device tree information to reflect that the active spare is assuming the role of boot processor. The system firmware further updates its device tree, in this example, to indicate that the former boot processor (for which LOL was detected) is assuming the role of active spare for the new boot processor.

[0072] In operational block **509**, PAL_MC_RESUME (which is a standard PAL call for resuming normal execution after a machine check (MC)) is called for the active spare (which is assuming the role of boot processor) to cause the OS to recognize the active spare processor as available for processing instructions, wherein the OS recognizes the active spare as the boot processor. Thus, the active spare has now assumed the role of the boot processor, taking the place of the former boot processor for which LOL was detected. Accordingly, in operational block **510** the former active spare (which has assumed the role of boot processor) resumes normal operation as the system's boot processor with its lockstep protection enabled. Thus, in this exemplary embodiment, upon detection of LOL for a boot processor, an active spare processor is idled, and the role of boot processor is switched to such idled spare that has lockstep protection, wherein the system effectively recovers lockstep for its boot processor without requiring a system shut down (i.e., without shutting down the OS). In certain implementations, the firmware may go on to take action to attempt to reestablish lockstep for the former boot processor (e.g., by resetting the former boot processor), and if successful in reestablishing its lockstep, such former boot processor may be reintroduced to the OS as an active spare processor for the new boot processor in the event that LOL is detected for this new boot processor.

[0073] The exemplary procedures of FIGS. 4 and 5 switch the role of boot processor to a spare processor in accordance with certain embodiments. While exemplary operational flows are provided in FIGS. 4 and 5, embodiments hereof are not limited to those illustrative examples. For instance, FIG. 6 shows a more general operational flow diagram for certain embodiments. In operational block **601**, LOL is detected for a processor in a multi-processor system. In operational block **602**, the system (e.g., system firmware) determines whether the processor for which LOL is detected is the system's boot processor. As described above, this is determined in certain embodiments by accessing the node of the device tree corresponding to the processor for which LOL is detected and determining if the information in that node of the device tree indicates that the processor is the system's boot processor.

[0074] In operational block **603**, responsive to determining that the processor for which LOL is detected is the system's boot processor, the system's firmware switches the role of boot processor to a spare processor, which may be either a hot spare or an active spare, as described above. Accordingly, the system's OS recognizes the spare processor as the system boot processor. In view of the above, embodiments hereof enable the role of boot processor to be dynamically switched, during system runtime, to a spare processor responsive to a detected LOL for the boot processor, thus enabling recovery from the LOL without requiring that the system be crashed or re-booted.

Original

Classifications

IPC	CPC	US Class	ECLA	ICO
G06F11/00	G06F11/1641	714/12	G06F11/20P10	S06F11:20P4
	G06F11/1658	714/11	G06F11/16C6	
	G06F11/2028		G06F11/16C8	
	G06F11/2035		G06F11/16D	
	G06F11/2038		G06F11/20P2E	
	G06F11/2043		G06F11/20P6	

Persons

Ultimate Owner

Original

FORAS TECHNOLOGIES {/api/control?client=&query=uo:{"FORAS%20TECHNOLOGIES"}&autosearch=true}

Original Assignees

Original	Standardized	Normalized
Scott L. Michaelis {/api/control?client=&query=pa:{"Scott%20L.%20Michaelis"}&autosearch=true} Plano, Texas, United States of America, US		
Anurupa Rajkumari {/api/control?client=&query=pa:{"Anurupa%20Rajkumari"}&autosearch=true} Round Rock, Texas, United States of America, US		
William B. McHardy {/api/control?client=&query=pa:{"William%20B.%20McHardy"}&autosearch=true} Fort Collins, Colorado, United States of America, US		

Current Assignees

Original	Standardized	Normalized
SONRAÍ MEMORY, LTD. {/api/control?client=&query=pacu:{"SONRA%C3%8D%20MEMORY%2C%20LTD."}&autosearch=true} SUITE 23, THE HYDE BUILDING, THE PARK, CARRICKMINE, DUBLIN 18, DUBLIN, IRELAND	SONRAI MEMORY {/api/control?client=&query=pacus:{"SONRAI%20MEMORY"}&autosearch=true}	

Inventor(s)

Original

Scott L. Michaelis {/api/control?client=&query=in:{"Scott%20L.%20Michaelis"}&autosearch=true}
Plano, Texas, United States of America, US

Anurupa Rajkumari {/api/control?client=&query=in:{"Anurupa%20Rajkumari"}&autosearch=true}
Round Rock, Texas, United States of America, US

William B. McHardy {/api/control?client=&query=in:{"William%20B.%20McHardy"}&autosearch=true}
Fort Collins, Colorado, United States of America, US

Legal information

Standardized Post issuance INPADOC

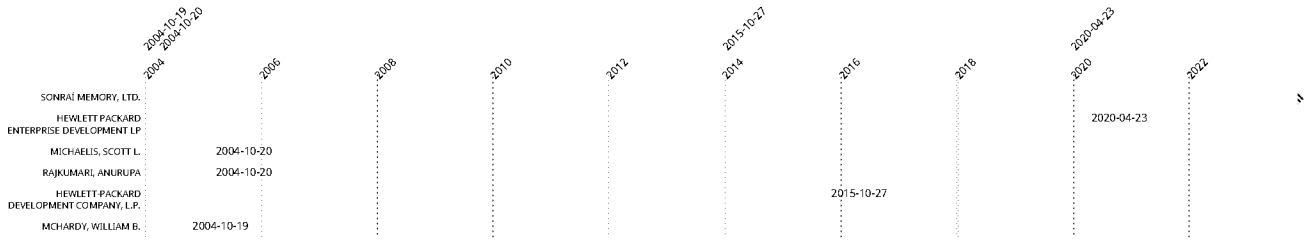
Legal status

Event	Date	Description
Granted	2009-11-24	GRANTED
Filed	2004-10-25	FILED

Designated states

Date	Event code	Description

Ownership



Payment status

Date	Status	Event code	Description

2021-05-12	Fee paid	M1553	PAYMENT OF MAINTENANCE FEE, 12TH YEAR, LARGE ENTITY.
2017-06-28	Fee paid	M1552	PAYMENT OF MAINTENANCE FEE, 8TH YEAR, LARGE ENTITY.
2013-03-08	Fee paid	M1551	PAYMENT OF MAINTENANCE FEE, 4TH YEAR, LARGE ENTITY.

Litigation/Opposition

Docket number	Docket description	Court	Date Filed
1:23cv640	Foras Technologies Ltd. V. Nissan Motor Company, Ltd. Et Al		2023-06-05
2:23cv150	Foras Technologies Ltd. V. Toyota Motor North America, Inc. Et Al		2023-04-04

Page: 1

Pharmaceuticals

Product	Type	Active Ingredient	Applicant	Application No.	Application Type	Approval Date	Patent Expiry Date	Patent usage code	Exclusivity Code	Exclusivity Date	Dosage	Strength	Product Number	Route	RLD	Patent Delist Requested	Exclusivity Text	Therapeutic Equivalence

No pharmaceuticals data

Citations

Display in a list (/api/control?client=&citationsSearch=_)

☐	backward	US4358823A	By other than the examiner	1982-11-09	Double redundant processor	TRW	
☐	backward	US5226152A	By other than the examiner	1993-07-06	Functional lockstep arrangement for redundant processors	MOTOROLA	
☐	backward	US5249188A	By other than the examiner	1993-09-28	Synchronizing two processors as an integral part of fault detection	COMMUNICATION SYSTEMS	
☐	backward	US5537655A	By other than the examiner	1996-07-16	Synchronized fault tolerant reset	BOEING	
☐	backward	US5675794A	By other than the examiner	1997-10-07	Method and apparatus for configuring multiple agents in a computer system	INTEL	
☐	backward	US5751932A	By other than the examiner	1998-05-12	Fail-fast, fail-functional, fault-tolerant multiprocessor system	TANDEM COMPUTERS	
☐	backward	US5758058A	By other than the examiner	1998-05-26	Apparatus and method for initializing a master/checker fault detecting microprocessor	INTEL	
☐	backward	US5764660A	By other than the examiner	1998-06-09	Processor independent error checking arrangement	ELSAG INT	
☐	backward	US5896523A	By other than the examiner	1999-04-20	Loosely-coupled, synchronized execution	MARATHON TECHN	
☐	backward	US5915082A	Cited by the examiner	1999-06-22	Error detection and fault isolation for lockstep processor systems	LOCKHEED	
☐	backward	US5991900A	By other than the examiner	1999-11-23	Bus controller	SUN MICROSYSTEMS	
☐	backward	US6065135A	By other than the examiner	2000-05-16	Error detection and fault isolation for lockstep processor systems	LOCKHEED	
☐	backward	US6141770A	By other than the examiner	2000-10-31	Fault tolerant computer system	GEN DYNAMICS INF SYSTEMS	
☐	backward	US6148348A	By other than the examiner	2000-11-14	Bridge interfacing two processing sets operating in a lockstep mode and having a posted write buffer storing write operations upon detection of a lockstep error	SUN MICROSYSTEMS	
☐	backward	US6220581B1	By other than the examiner	2001-04-24	Bicycle seat gas spring adjustment system	MUELLER THOMAS L	
☐	backward	US6263452B1	By other than the examiner	2001-07-17	Fault-tolerant computer system with online recovery and reintegration of redundant components	COMPAQ COMPUTER	
☐	backward	US6438687B2	By other than the examiner	2002-08-20	Method and apparatus for improved storage of computer system configuration information	MICRON TECHNOLOGY	
☐	backward	US6473869B2	By other than the examiner	2002-10-29	Fault resilient/fault tolerant computing	MARATHON TECHNOLOGIES	
☐	backward	US6560682B1	By other than the examiner	2003-05-06	System and method for terminating lock-step sequences in a multiprocessor system	INTEL	
☐	backward	US6604177B1	By other than the examiner	2003-08-05	Communication of dissimilar data between lock-stepped processors	HEWLETT PACKARD DEVELOPMENT	

Non-patent literature

U.S. Appl. No. 10/972,796, filed concurrently herewith, entitled "System and Method for Maintaining in a Multi-Processor System a Spare Processor That is in Lockstep for Use in Recovering From Loss of Lockstep for Another Processor".
U.S. Appl. No. 10/973,077, filed concurrently herewith, entitled "System and Method for Establishing a Spare Processor for Recovering From Loss of Lockstep in a Boot Processor".
U.S. Appl. No. 10/972,835, filed concurrently herewith, entitled "System and Method for Using Information Relating to a Detected Loss of Lockstep for Determining a Responsive Action".
U.S. Appl. No. 10/973,076, filed concurrently herewith, entitled "System and Method for Providing Firmware Recoverable Lockstep Protection".
U.S. Appl. No. 10/973,004, filed concurrently herewith, entitled "System and Method for Configuring Lockstep Mode of a Prcessor Module".
U.S. Appl. No. 10/973,003, filed concurrently herewith, entitled "System and Method for Reestablishing Lockstep for a Processor Module for Which Loss of Lockstep is Detected".
U.S. Appl. No. 10/972,888, filed concurrently herewith, entitled "System and Method for System Firmware Causing an Operating System to Idle a Processor".
U.S. Appl. No. 10/973,075, filed concurrently herewith, entitled "System and Method for Reintroducing a Processor Module to an Operating System After Lockstep Recovery".
Notice of Allowance for U.S. Appl. No. 10/972,835 dated Nov. 20, 2008 in the United States.
Notice of Allowance for U.S. Appl. No. 10/972,835 dated Aug. 1, 2008 in the United States.
Office Action for U.S. Appl. No. 10/972,835 dated Mar. 3, 2008 in the United States.
Office Action for U.S. Appl. No. 10/972,835 dated Jul. 12, 2007 in the United States.
Office Action for U.S. Appl. No. 10/973,003 dated Mar. 5, 2008 in the United States.
Office Action for U.S. Appl. No. 10/973,003 dated Sep. 12, 2007 in the United States.
Notice of Allowance for U.S. Appl. No. 10/973,076 dated Nov. 28, 2008 in the United States.
Notice of Allowance for U.S. Appl. No. 10/973,076 dated Jul. 16, 2008 in the United States.
Notice of Allowance for U.S. Appl. No. 10/973,076 dated Jun. 3, 2008 in the United States.
Final Office Action for U.S. Appl. No. 10/973,076 dated Feb. 27, 2008 in the United States.
Office Action for U.S. Appl. No. 10/973,076 dated Sep. 12, 2007 in the United States.
Notice of Allowance for U.S. Appl. No. 10/972,888 dated Nov. 14, 2007 in the United States.
Office Action for U.S. Appl. No. 10/972,888 dated Jul. 18, 2007 in the United States.
Notice of Allowance for U.S. Appl. No. 10/973,004 dated Aug. 1, 2007 in the United States.
Office Action for U.S. Appl. No. 10/973,004 dated Apr. 10, 2007 in the United States.
Final Office Action for U.S. Appl. No. 10/973,075 dated May 12, 2009 in the United States.
Office Action for U.S. Appl. No. 10/973,075 dated Oct. 29, 2008 in the United State.
Office Action for U.S. Appl. No. 10/973,075 dated Mar. 12, 2008 in the United States.
Office Action for U.S. Appl. No. 10/973,075 dated Jul. 2, 2007 in the United States.
Notice of Allowance for U.S. Appl. No. 10/972,796 dated Dec. 19, 2007 in the United States.
Office Action for U.S. Appl. No. 10/972,796 dated Jul. 12, 2007 in the United States.
Shoichi Araki, Real-Time Tracking of Multiple Moving Object Contours in a Moving Camera Image Sequence, vol. E83-D IEICE Trans. Inf. & Syst, Jul. 7, 2000, 9 pages. (/intapi/scopus/0034229432) **42** Citations in Scopus
Jiang-Bin Zheng, Proceedings of the First International Conference on Machine Learning and Cybernetics, Beijing, Nov. 4-5, 2002, 4 pages.
Jong Ryul Kim, Automatic Localization and Trackign of Moving Objects Using Adaptive Snake Algorithm, ICICS-PCM 2003, Dec. 15-18, 2003, 5 pages, Singapore.
Jianhoa Meng, CVEPS-A Compressed Video Editing and Parsing System, Nov. 18, 1996, pp. 1-11.
Notice of Allowance dated Feb. 11, 2009 for U.S. Appl. No. 10/973,077 in the United States.
Notice of Allowance dated Sep. 23, 2008 for U.S. Appl. No. 10/973,077 in the United States.
Office Action dated Mar. 19, 2008 for U.S. Appl. No. 10/973,077 in the United States.
Office Action dated Sep. 20, 2007 for U.S. Appl. No. 10/973,077 in the United States.
Final Office Action for U.S. Appl. No. 10/972,835 dated Nov. 21, 2007 in the United States.
Office Action for U.S. Appl. No. 10/973,075 dated Dec. 13, 2007 in the United States.

Similar documents

Display in a list (/api/control?client=&similarSearch=_)

The similarity check is done by matching this document to all documents in the database. Only the title and abstract text are matched, resulting in a score. The 10 highest-scoring documents are displayed here.

All

System and method for maintaining in a multi-processor system a spare processor that is in lockstep for use in recovering from loss of lockstep for another processor

Publication number US7366948B2
Publication date 2008-04-29
Application date 2004-10-25

According to one embodiment, a method comprises assigning a first processor of a multi-processor system a role of spare processor for at least a second processor, and responsive to detecting loss of lockstep (LOL) for any of the at least a second processor, the first processor replaces the processor for which the LOL is detected. The method further comprises reestablishing lockstep for the processor for which the LOL is detected, and assigning the processor having its lockstep reestablished the role of spare processor for at least the first processor.

System and method for maintaining in a multi-processor system a spare processor that is in lockstep for use in recovering from loss of lockstep for another processor

Publication number US20060107106A1
Publication date 2006-05-18
Application date 2004-10-25

According to one embodiment, a method comprises assigning a first processor of a multi-processor system a role of spare processor for at least a second processor, and responsive to detecting loss of lockstep (LOL) for any of the at least a second processor, the first processor replaces the processor for which the LOL is detected. The method further comprises reestablishing lockstep for the processor for which the LOL is detected, and assigning the processor having its lockstep reestablished the role of spare processor for at least the first processor.

System and method for reestablishing lockstep for a processor module for which loss of lockstep is detected

Publication number US20060107116A1
Publication date 2006-05-18
Application date 2004-10-25

According to one embodiment, a method comprises, responsive to detection of loss of lockstep (LOL) for a processor module in a system, firmware requesting an operating system to idle the processor module. The method further comprises the operating system idling the processor module and returning control of the processor module to the firmware. According to one embodiment, a method comprises an operating system idling a processor module for which A loss of lockstep (LOL) is detected, and system firmware receiving control of the processor module. The method further comprises the system firmware determining whether a LOL was detected for the processor module, and if determined that LOL was detected for the processor module, the system firmware resetting the processor module to reestablish lockstep for the processor module.

System and method for establishing a spare processor for recovering from loss of lockstep in a boot processor

Publication number US7627781B2
Publication date 2009-12-01
Application date 2004-10-25

A system comprises a plurality of processors, and data storage storing information that assigns a role of boot processor to one of the plurality of processors and assigns a role of spare processor to another of the plurality of processors. The system further comprises logic operable, responsive to detecting loss of lockstep for the boot processor, for transferring, during system runtime, the role of boot processor to the spare processor.

System and method for establishing a spare processor for recovering from loss of lockstep in a boot processor

Publication number US20060107112A1
Publication date 2006-05-18
Application date 2004-10-25

A system comprises a plurality of processors, and data storage storing information that assigns a role of boot processor to one of the plurality of processors and assigns a role of spare processor to another of the plurality of processors. The system further comprises logic operable, responsive to detecting loss of lockstep for the boot processor, for transferring, during system runtime, the role of boot processor to the spare processor.

System and method for reintroducing a processor module to an operating system after lockstep recovery

Publication number US20060107111A1
Publication date 2006-05-18
Application date 2004-10-25

According to one embodiment, a method comprises, responsive to detection of loss of lockstep (LOL) for a processor module in a system, setting status information stored to the system for the processor module to indicate that the processor module has an error. The method further comprises reestablishing lockstep for the processor module without shutting down the system's operating system, and updating the status information for the processor module to indicate that the processor module no longer has the error. The method further comprises causing the system's operating system to recognize that the processor module having its lockstep reestablished is available for processing.

System and method for reintroducing a processor module to an operating system after lockstep recovery

Publication number US7818614B2
Publication date 2010-10-19
Application date 2004-10-25

According to one embodiment, a method comprises, responsive to detection of loss of lockstep (LOL) for a processor module in a system, setting status information stored to the system for the processor module to indicate that the processor module has an error. The method further comprises reestablishing lockstep for the processor module without shutting down the system's operating system, and updating the status information for the processor module to indicate that the processor module no longer has the error. The method further comprises causing the system's operating system to recognize that the processor module having its lockstep reestablished is available for processing.

System and method for using information relating to a detected loss of lockstep for determining a responsive action

Publication number US20060107114A1
Publication date 2006-05-18
Application date 2004-10-25

According to one embodiment, a method comprises detecting a loss of lockstep (LOL) for a processor module. The method further comprises determining a type of LOL that is detected, and, based at least in part on the determined type of LOL, determining a responsive action to take for the LOL. According to one embodiment, a method comprises detecting a loss of lockstep (LOL) for a processor module. The method further comprises using information identifying at least one of type of the detected LOL and source of the detected LOL to determine a responsive action to take for the LOL.

System and method for using information relating to a detected loss of lockstep for determining a responsive action

Publication number US7516359B2
Publication date 2009-04-07
Application date 2004-10-25

According to one embodiment, a method comprises detecting a loss of lockstep (LOL) for a processor module. The method further comprises determining a type of LOL that is detected, and, based at least in part on the determined type of LOL, determining a responsive action to take for the LOL. According to one embodiment, a method comprises detecting a loss of lockstep (LOL) for a processor module. The method further comprises using information identifying at least one of type of the detected LOL and source of the detected LOL to determine a responsive action to take for the LOL.

System and method for system firmware causing an operating system to idle a processor

Publication number US7356733B2
Publication date 2008-04-08
Application date 2004-10-25

According to one embodiment, a method comprises system firmware instructing a system's operating system to idle a processor, and responsive to the instructing, the operating system idling the processor and returning control over the processor to the system firmware. According to one embodiment, a method comprises detecting loss of lockstep (LOL) for a processor module in a system, and responsive to the detecting LOL for the processor module, system firmware instructing an operating system to idle the processor module.

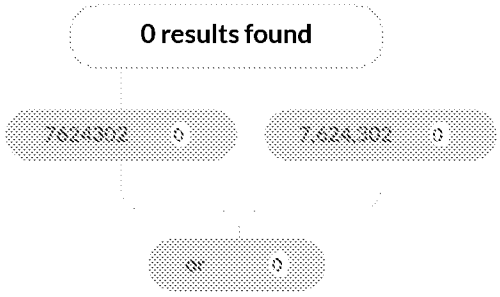
Cases 0

Filters

Cases Clear

No documents found in Cases filtered by Cases.

Search Tree:



Try the following:

- Remove one or more filters: Cases
- Review your results in the other categories.
- Modify your search (edit or remove some search terms or add synonyms).
- Check for spelling errors.
- Use the options available from the Actions menu above to modify your search.

Copyright © 2023 LexisNexis.

User Name: Renee Preston
Date and Time: Friday, September 1, 2023 9:15:00PM EDT
Job Number: 204922881

Document (1)

1. *Patent No. 7,624,302 Issued on Nov. 24, Assigned to HP Development for Boot Processor Role Method (Texas , Colorado Inventors)*

Client/Matter: -None-
Search Terms: 7624302 or 7,624,302
Search Type: Terms and Connectors
Narrowed by:

Content Type	Narrowed by
News	-None-

Patent No. 7,624,302 Issued on Nov. 24, Assigned to HP Development for Boot Processor Role Method (Texas, Colorado Inventors)

US Fed News

November 25, 2009 Wednesday 10:44 AM EST

Copyright 2009 HT Media Ltd. All Rights Reserved

Length: 302 words

Dateline: ALEXANDRIA, Va.

Body

ALEXANDRIA, Va., Nov. 30 -- Anurupa Rajkumari of Round Rock, Texas, William B. McHardy of Fort Collins, Colo., and Scott L. Michaelis of Plano, Texas, have developed a method for switching the role of boot processor to a spare processor responsive to detection of loss of lockstep in a boot processor. The inventors were issued U.S. Patent No. ***7,624,302*** on Nov. 24.

The patent has been assigned to Hewlett-Packard Development Co. LP, Houston.

According to the abstract released by the U.S. Patent & Trademark Office: "According to one embodiment, a method comprises detecting loss of lockstep (LOL) for a processor in a multi-processor system. The method further comprises determining that the processor for which the LOL is detected is assigned the role of boot processor, and switching the role of boot processor to a spare processor without shutting down the system's operating system. In another embodiment, a method comprises system firmware determining that an LOL is detected for a lockstep pair of processors that are assigned the role of boot processor in a system. The method further comprises determining one of the lockstep pair of processors that is not the cause of the LOL, and copying the state of the determined one of the lockstep pair of processors that is not the cause of the LOL to a spare processor. The method further comprises switching the role of boot processor to the spare processor."

The original application was filed on Oct. 25, 2004.

For further information please visit:*http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&p=1&u=%2Fnetahtml%2FPTO%2Fsearch-bool.html&r=1&f=G&l=50&co1=AND&d=PTXT&s1=7624302&OS=7624302&RS=7624302* For more information about US Fed News contract awards please contact: Sarabjit Jagirdar, US Fed News, Email:-*htsyndication@hindustantimes.com*

For any query with respect to this article or any other content requirement, please contact Editor at *htsyndication@hindustantimes.com*

Load-Date: November 30, 2009

End of Document

User Name: Renee Preston
Date and Time: Friday, September 1, 2023 9:47:00PM EDT
Job Number: 204923340

Document (1)

1. *2:23cv150, Foras Technologies Ltd. V. Toyota Motor North America, Inc. Et Al*

Client/Matter: -None-
Search Terms: 7624302 or 7,624,302
Search Type: Terms and Connectors
Narrowed by:

Content Type	Narrowed by
Dockets	-None-

2:23cv150, Foras Technologies Ltd. V. Toyota Motor North America, Inc. Et Al

US District Court Docket

United States District Court, Texas Eastern

(Marshall)

This case was retrieved on 08/31/2023

Header

Case Number: 2:23cv150
Date Filed: 04/05/2023
Assigned To: District Judge Rodney Gilstrap
Nature of Suit: ***Patent*** (830)
Cause: ***Patent*** Infringement
Lead Docket: None
Other Docket: 2:23cv00218, 2:23cv00314
Jurisdiction: Federal Question

Class Code: Open
Statute: 35:145
Jury Demand: Both
Demand Amount: $0
NOS Description: ***Patent***

Participants

Litigants

Foras Technologies Ltd.
Plaintiff

Attorneys

Drew Bennett Hollander
ATTORNEY TO BE NOTICED
BC Law Group, P.C.
200 Madison Avenue 24th Floor
New York, NY 10016
USA
212-951-0100 Email:Dhollander@b-Clg.Com

Jonathan Randy Yim
ATTORNEY TO BE NOTICED
BC Law Group, P.C.
200 Madison Avenue 24th Floor
New York, NY 10016
USA
516-359-9668 Email:Jyim@b-Clg.Com

Justin Kurt Truelove
ATTORNEY TO BE NOTICED
Truelove Law Firm
207 N Wellington P O Box 1409
Marshall, TX 75671
USA
903-938-8321 Fax: 903-215-8510
Email:Kurt@truelovelawfirm.Com

Seth Raymond Hasenour
ATTORNEY TO BE NOTICED

Litigants	Attorneys
	BC Law Group, P.C. 607 Terrell Hill Drive Ste Unit B Austin, TX 78704 USA 516-359-9668 Email:Shasenour@b-Clg.Com Brett E Cooper ATTORNEY TO BE NOTICED BC Law Group, P.C. 200 Madison Avenue 24th Floor New York, NY 10016 USA 516-359-9668 Email:Bcooper@b-Clg.Com
Toyota Motor North America, Inc. [Terminated: 07/12/2023] **Defendant**	Steven Routh LEAD ATTORNEY;ATTORNEY TO BE NOTICED [Terminated: 07/12/2023] Orrick, Herrington & Sutcliffe LLP - Washington DC Columbia Center 1152 15th Street, Nw Washington, DC 20005-1706 USA 202-339-8509 Fax: 202-339-8500 Email:Srouth@orrick.Com
Denso Corporation **Defendant**	Steven Routh LEAD ATTORNEY;ATTORNEY TO BE NOTICED Orrick, Herrington & Sutcliffe LLP - Washington DC Columbia Center 1152 15th Street, Nw Washington, DC 20005-1706 USA 202-339-8509 Fax: 202-339-8500 Email:Srouth@orrick.Com Andrew Thompson (Tom) Gorham ATTORNEY TO BE NOTICED Gillam & Smith LLP 102 N College, Suite 800 Tyler, TX 75702 USA 9039348450 Fax: 9039349257 Email:Tom@gillamsmithlaw.Com Melissa Richards Smith ATTORNEY TO BE NOTICED Gillam & Smith, LLP 303 South Washington Avenue Marshall, TX 75670 USA 903-934-8450 Fax: 903-934-9257 Email:Melissa@gillamsmithlaw.Com Sten Anker Jensen ATTORNEY TO BE NOTICED Orrick, Herrington & Sutcliffe LLP - Washington DC Columbia Center 1152 15th Street, Nw Washington, DC 20005-1706 USA 202-339-8436 Fax: 202-339-8500 Email:Sjensen@orrick.Com
Kia Corporation Consolidated Civil Action 2:23cv219	Douglas E Lumish LEAD ATTORNEY;ATTORNEY TO BE NOTICED

Litigants	Attorneys
Consol Defendant	Latham & Watkins LLP - Menlo Park 140 Scott Drive Menlo Park, CA 94025-1008 USA 650-328-4600 Fax: 650-463-2600 Email:Doug.Lumish@lw.Com Adam Michael Greenfield ATTORNEY TO BE NOTICED Latham & Watkins LLP - D.C. 555 Eleventh Street, Nw Suite 1000 Washington, DC 20004 USA 202.637.2200 Fax: 202.637.2201 Email:Adam.Greenfield@lw.Com Andrew Thompson (Tom) Gorham ATTORNEY TO BE NOTICED Gillam & Smith LLP 102 N College, Suite 800 Tyler, TX 75702 USA 9039348450 Fax: 9039349257 Email:Tom@gillamsmithlaw.Com Jacob Vannette ATTORNEY TO BE NOTICED Latham & Watkins LLP 555 Eleventh Street, Nw Suite 1000 Washington, DC 20004 USA 202-637-2200 Email:Jake.Vannette@lw.Com Melissa Richards Smith ATTORNEY TO BE NOTICED Gillam & Smith, LLP 303 South Washington Avenue Marshall, TX 75670 USA 903-934-8450 Fax: 903-934-9257 Email:Melissa@gillamsmithlaw.Com
Kia America, Inc. Consolidated Civil Action 2:23cv219] **Consol Defendant**	Douglas E Lumish LEAD ATTORNEY;ATTORNEY TO BE NOTICED Latham & Watkins LLP - Menlo Park 140 Scott Drive Menlo Park, CA 94025-1008 USA 650-328-4600 Fax: 650-463-2600 Email:Doug.Lumish@lw.Com Adam Michael Greenfield ATTORNEY TO BE NOTICED Latham & Watkins LLP - D.C. 555 Eleventh Street, Nw Suite 1000 Washington, DC 20004 USA 202.637.2200 Fax: 202.637.2201 Email:Adam.Greenfield@lw.Com

Litigants

Attorneys

Andrew Thompson (Tom) Gorham
ATTORNEY TO BE NOTICED
Gillam & Smith LLP
102 N College, Suite 800
Tyler, TX 75702
USA
9039348450 Fax: 9039349257
Email:Tom@gillamsmithlaw.Com

Jacob Vannette
ATTORNEY TO BE NOTICED
Latham & Watkins LLP
555 Eleventh Street, Nw Suite 1000
Washington, DC 20004
USA
202-637-2200 Email:Jake.Vannette@lw.Com

Melissa Richards Smith
ATTORNEY TO BE NOTICED
Gillam & Smith, LLP
303 South Washington Avenue
Marshall, TX 75670
USA
903-934-8450 Fax: 903-934-9257
Email:Melissa@gillamsmithlaw.Com

Toyota Motor North America, Inc.
Consolidated Civil Action 2:23-321 |
Consol Defendant

Steven Routh
ATTORNEY TO BE NOTICED
Orrick, Herrington & Sutcliffe LLP - Washington DC
Columbia Center 1152 15th Street, Nw
Washington, DC 20005-1706
USA
202-339-8509 Fax: 202-339-8500
Email:Srouth@orrick.Com

Proceedings

#	Date	Proceeding Text	Source
1	04/05/2023	COMPLAINT against Denso Corporation, Toyota Motor North America, Inc. (Filing fee $ 402 receipt number ATXEDC-9437538.), filed by Foras Technologies Ltd.. (Attachments: # 1 Exhibit 1, # 2 Exhibit 2, # 3 Exhibit 3, # 4 Exhibit 4, # 5 Exhibit 5, # 6 Exhibit 6, # 7 Exhibit 7, # 8 Exhibit 8, # 9 Exhibit 9, # 10 Civil Cover Sheet)(Cooper, Brett) (Entered: 04/05/2023)	
2	04/05/2023	Notice of Filing of ***Patent***/Trademark Form (AO 120). AO 120 mailed to the Director of the U.S. ***Patent*** and Trademark Office. (Cooper, Brett) (Entered: 04/05/2023)	
3	04/05/2023	CORPORATE DISCLOSURE STATEMENT filed by Foras Technologies Ltd. (Cooper, Brett) (Entered: 04/05/2023)	
	04/05/2023	Case Assigned to District Judge Rodney Gilstrap. (nkl,) (Entered: 04/05/2023)	
	04/05/2023	In accordance with the provisions of 28 USC Section 636(c), you are hereby notified that a U.S. Magistrate Judge of this district court is available to conduct any or all proceedings in this case including a jury or non-jury trial and to order the entry of a final judgment. The form Consent to Proceed Before Magistrate Judge is available on our website. All signed consent forms, excluding pro se parties, should be filed electronically using the event Notice	

#	Date	Proceeding Text	Source
		Regarding Consent to Proceed Before Magistrate Judge. (nkl,) (Entered: 04/05/2023)	
4	04/05/2023	SUMMONS Issued as to Denso Corporation c/o Denso International America, Inc.c/o Corporation Service Company d/b/a CSC Lawyers Incorporating Service Company, Toyota Motor North America, Inc. c/o CT Corporation System. (Attachments: # 1 Summons(es))(nkl,) (Entered: 04/05/2023)	
5	04/11/2023	SUMMONS Returned Executed by Foras Technologies Ltd.. Toyota Motor North America, Inc. served on 4/6/2023, answer due 4/27/2023. (Cooper, Brett) (Entered: 04/11/2023)	
6	04/12/2023	SUMMONS Returned Executed by Foras Technologies Ltd.. Denso Corporation served on 4/6/2023, answer due 4/27/2023. (Cooper, Brett) (Entered: 04/12/2023)	
7	04/27/2023	***DEFICIENT DOCUMENT***Joint MOTION for Extension of Time to File Response to Complaint by Denso Corporation. (Attachments: # 1 Text of Proposed Order)(Routh, Steven) Modified on 4/27/2023 (ch,). (Entered: 04/27/2023)	
8	04/27/2023	***DEFICIENT DOCUMENT***Joint MOTION for Extension of Time to File Response to Complaint by Toyota Motor North America, Inc.. (Attachments: # 1 Text of Proposed Order)(Routh, Steven) Modified on 4/27/2023 (ch,). (Entered: 04/27/2023)	
	04/27/2023	NOTICE of Deficiency regarding the 7 submitted Needs Certificate of Conference. Correction should be made by one business day (ch,) (Entered: 04/27/2023)	
	04/27/2023	NOTICE of Deficiency regarding the 8 submitted Needs Certificate of Conference. Correction should be made by one business day (ch,) (Entered: 04/27/2023)	
9	04/27/2023	Joint MOTION for Extension of Time to File Response to Complaint by Denso Corporation. (Attachments: # 1 Text of Proposed Order)(Routh, Steven) (Entered: 04/27/2023)	
10	04/27/2023	Joint MOTION for Extension of Time to File Response to Complaint by Toyota Motor North America, Inc.. (Attachments: # 1 Text of Proposed Order)(Routh, Steven) (Entered: 04/27/2023)	
11	04/28/2023	ORDER granting 9 Motion for Extension of Time to File Response to Complaint. Signed by District Judge Rodney Gilstrap on 4/28/2023. (nkl,) (Entered: 05/01/2023)	
12	04/28/2023	ORDER granting 10 Motion for Extension of Time to File Response to Complaint. Signed by District Judge Rodney Gilstrap on 4/28/2023. (nkl,) (Entered: 05/01/2023)	
13	05/23/2023	NOTICE of Attorney Appearance by Drew Bennett Hollander on behalf of Foras Technologies Ltd. (Hollander, Drew) (Entered: 05/23/2023)	
14	05/23/2023	NOTICE of Attorney Appearance by Seth Raymond Hasenour on behalf of Foras Technologies Ltd. (Hasenour, Seth) (Entered: 05/23/2023)	
15	05/23/2023	NOTICE of Attorney Appearance by Jonathan Randy Yim on behalf of Foras Technologies Ltd. (Yim, Jonathan) (Entered: 05/23/2023)	
16	06/13/2023	CONSOLIDATION ORDER - The above-captioned cases are hereby ORDERED to be CONSOLIDATED for all pretrial issues with the LEAD CASE, Case No. 2:23-cv-00150. All parties are instructed to file any future filings in the LEAD CASE. Individual cases remain active for trial.. Signed by District Judge Rodney Gilstrap on 6/13/2023. (nkl,) (Entered: 06/13/2023)	
17	06/15/2023	ORDER setting Scheduling Conference for 7/20/2023 at 9:30 AM before District Judge Rodney Gilstrap. Signed by District Judge Rodney Gilstrap on 6/15/23. (tkd,) (Entered: 06/16/2023)	
18	06/22/2023	NOTICE of Attorney Appearance by Melissa Richards Smith on	

#	Date	Proceeding Text	Source
		behalf of Kia America, Inc., Kia Corporation (Smith, Melissa) (Entered: 06/22/2023)	
19	06/22/2023	Joint MOTION for extension of time for Toyota and DENSO to respond to the complaint and for Foras to serve infringement contentions by Foras Technologies Ltd.. (Attachments: # 1 Proposed Order)(Cooper, Brett) (Entered: 06/22/2023)	
20	06/23/2023	ORDER granting 19 Joint MOTION for extension of time for Toyota and DENSO to respond to the complaint and for Foras to serve infringement contentions. Denso Corporation answer due 7/21/2023; Toyota Motor North America, Inc. answer due 7/21/2023.. Signed by District Judge Rodney Gilstrap on 6/23/2023. (ch,) (Entered: 06/23/2023)	
21	06/27/2023	NOTICE of Attorney Appearance by Douglas E Lumish on behalf of Kia America, Inc., Kia Corporation (Lumish, Douglas) (Entered: 06/27/2023)	
22	06/27/2023	NOTICE of Attorney Appearance by Adam Michael Greenfield on behalf of Kia America, Inc., Kia Corporation (Greenfield, Adam) (Entered: 06/27/2023)	
23	06/27/2023	NOTICE of Attorney Appearance by Jacob Vannette on behalf of Kia America, Inc., Kia Corporation (Vannette, Jacob) (Entered: 06/27/2023)	
24	06/27/2023	NOTICE of Attorney Appearance by Justin Kurt Truelove on behalf of Foras Technologies Ltd. (Truelove, Justin) (Entered: 06/27/2023)	
25	06/28/2023	NOTICE of Attorney Appearance by Justin Kurt Truelove on behalf of Foras Technologies Ltd. (Truelove, Justin) (Entered: 06/28/2023)	
26	07/03/2023	Joint MOTION FOR EXTENSIONS OF TIME REGARDING INFRINGEMENT AND INVALIDITY CONTENTIONS by Kia America, Inc., Kia Corporation. (Attachments: # 1 Proposed Order)(Smith, Melissa) (Entered: 07/03/2023)	
27	07/07/2023	ORDER granting 26 Motion for Extensions of Time Regarding Infringement and Invalidity Contentions. Signed by District Judge Rodney Gilstrap on 7/6/2023. (nkl,) (Entered: 07/07/2023)	
28	07/10/2023	ANSWER to 1 Complaint, by Toyota Motor North America, Inc.,(Routh, Steven) (Entered: 07/10/2023)	
29	07/10/2023	Fed. R. Civ. P. 7.1(a)(1) Disclosure Statement filed by Toyota Motor North America, Inc. identifying Corporate Parent Toyota Motor Corporation for Toyota Motor North America, Inc.. (Routh, Steven) (Entered: 07/10/2023)	
30	07/10/2023	ANSWER to 1 Complaint, by Denso Corporation.(Routh, Steven) (Entered: 07/10/2023)	
31	07/10/2023	Fed. R. Civ. P. 7.1(a)(1) Disclosure Statement filed by Denso Corporation identifying Corporate Parent Toyota Motor Corporation for Denso Corporation. (Routh, Steven) (Entered: 07/10/2023)	
32	07/10/2023	Joint MOTION to Sever and Stay Claims Against Toyota by Denso Corporation, Toyota Motor North America, Inc.. (Attachments: # 1 Proposed Order)(Routh, Steven) (Additional attachment(s) added on 7/11/2023: # 2 Revised Proposed Order) (nkl,). (Entered: 07/10/2023)	
	07/10/2023	In accordance with the provisions of 28 USC Section 636(c), you are hereby notified that a U.S. Magistrate Judge of this district court is available to conduct any or all proceedings in this case including a jury or non-jury trial and to order the entry of a final judgment. The form Consent to Proceed Before Magistrate Judge is available on our website. All signed consent forms, excluding pro se parties, should be filed electronically using the event Notice	

#	Date	Proceeding Text	Source
		Regarding Consent to Proceed Before Magistrate Judge. (nkl,) (Entered: 07/10/2023)	
33	07/12/2023	ORDER granting 32 Motion to Sever and Stay Claims Against Toyota. Signed by District Judge Rodney Gilstrap on 7/11/2023. (nkl,) (Entered: 07/12/2023)	
34	07/12/2023	NOTICE of Attorney Appearance by Melissa Richards Smith on behalf of Denso Corporation (Smith, Melissa) (Entered: 07/12/2023)	
35	07/19/2023	NOTICE of Attorney Appearance by Andrew Thompson (Tom) Gorham on behalf of Denso Corporation, Kia America, Inc., Kia Corporation (Gorham, Andrew) (Entered: 07/19/2023)	
	07/20/2023	Minute Entry for proceedings held before District Judge Rodney Gilstrap (Lead 2:23-cv-150; Members 2:23-cv-219; 2:23-cv-321_stayed): Scheduling Conference/Case Management Conference held on 7/20/2023. Counsel for the parties appeared. Court asked whether they consented to trial before the U.S. Magistrate Judge. Court then gave counsel Claim Construction and Jury Selection/Trial dates. (Court Reporter Shawn McRoberts) (aeb) (Entered: 07/24/2023)	
36	07/21/2023	NOTICE by Foras Technologies Ltd. of Compliance Regarding Disclosure of Asserted Claims and Infringement Contentions (Cooper, Brett) (Entered: 07/21/2023)	
37	07/24/2023	ANSWER to Complaint by Kia America, Inc., Kia Corporation.(Smith, Melissa) (Entered: 07/24/2023)	
38	07/24/2023	Fed. R. Civ. P. 7.1(a)(1) Disclosure Statement filed by Kia America, Inc., Kia Corporation (Smith, Melissa) (Entered: 07/24/2023)	
39	07/24/2023	DEMAND for Trial by Jury by Kia America, Inc., Kia Corporation. (Smith, Melissa) (Entered: 07/24/2023)	
	07/24/2023	In accordance with the provisions of 28 USC Section 636(c), you are hereby notified that a U.S. Magistrate Judge of this district court is available to conduct any or all proceedings in this case including a jury or non-jury trial and to order the entry of a final judgment. The form Consent to Proceed Before Magistrate Judge is available on our website. All signed consent forms, excluding pro se parties, should be filed electronically using the event Notice Regarding Consent to Proceed Before Magistrate Judge. (nkl,) (Entered: 07/24/2023)	
40	08/03/2023	Joint MOTION for Entry of Agreed Docket Control Order by Foras Technologies Ltd.. (Attachments: # 1 Proposed Order)(Cooper, Brett) (Entered: 08/03/2023)	
41	08/03/2023	Joint MOTION for Entry of Agreed Discovery Order by Foras Technologies Ltd.. (Attachments: # 1 Proposed Order)(Cooper, Brett) (Entered: 08/03/2023)	
42	08/09/2023	DOCKET CONTROL ORDER re 40 Joint MOTION for Entry of Agreed Docket Control Order filed by Foras Technologies Ltd.., Pretrial Conference set for 12/2/2024 at 09:00 AM before District Judge Rodney Gilstrap., Amended Pleadings due by 5/2/2024., Jury Selection set for 1/6/2025 at 09:00AM before District Judge Rodney Gilstrap., Markman Hearing set for 7/18/2024 at 09:00 AM before District Judge Rodney Gilstrap., Motions in Limine due by 11/12/2024., Proposed Pretrial Order due by 11/25/2024. Signed by District Judge Rodney Gilstrap on 8/8/2023. (nkl,) (Entered: 08/09/2023)	
43	08/09/2023	DISCOVERY ORDER re 41 Motion for Entry of Agreed Discovery Order. Signed by District Judge Rodney Gilstrap on 8/8/2023. (nkl,) (Entered: 08/09/2023)	
44	08/09/2023	Joint MOTION to Amend Docket Control Order re 42 Order,,.	

#	Date	Proceeding Text	Source
		Terminate Motions,,, Scheduling Order,, by Foras Technologies Ltd.. (Attachments: # 1 Proposed Order)(Cooper, Brett) (Entered: 08/09/2023)	
45	08/10/2023	NOTICE of Attorney Appearance by Sten Anker Jensen on behalf of Denso Corporation (Jensen, Sten) (Entered: 08/10/2023)	
46	08/11/2023	FIRST AMENDED DOCKET CONTROL ORDER re 44 Motion to Amend Docket Control Order. Signed by District Judge Rodney Gilstrap on 8/11/2023. (nkl,) (Entered: 08/11/2023)	
47	08/17/2023	Joint MOTION for Entry of Agreed Protective Order by Foras Technologies Ltd.. (Attachments: # 1 Proposed Order)(Cooper, Brett) (Entered: 08/17/2023)	
48	08/17/2023	NOTICE by Foras Technologies Ltd. of Compliance Regarding Initial Disclosures (Cooper, Brett) (Entered: 08/17/2023)	
49	08/17/2023	NOTICE of Discovery Disclosure by Denso Corporation (Routh, Steven) (Entered: 08/17/2023)	
50	08/18/2023	NOTICE by Kia America, Inc., Kia Corporation of Compliance Regarding Initial and Additional Disclosures (Smith, Melissa) (Entered: 08/18/2023)	
51	08/21/2023	PROTECTIVE ORDER. Signed by District Judge Rodney Gilstrap on 8/21/2023. (nkl,) (Entered: 08/21/2023)	

Patents

Number	Title	Issued	Class	Subclass
7,502,958	System and method for providing firmware recoverable lockstep protection	03/10/2009	714	12
7,624,302	System and method for switching the role of boot processor to a spare processor responsive to detection of loss of lockstep in a boot processor	11/24/2009	714	12
7,627,781	System and method for establishing a spare processor for recovering from loss of lockstep in a boot processor	12/01/2009	714	13

Copyright © LexisNexis CourtLink, Inc. All Rights Reserved.
*** THIS DATA IS FOR INFORMATIONAL PURPOSES ONLY ***

End of Document



User Name: Renee Preston
Date and Time: Friday, September 1, 2023 9:46:00PM EDT
Job Number: 204923317

Document (1)

1. *1:23cv640, Foras Technologies Ltd. V. Nissan Motor Company, Ltd. Et Al*

Client/Matter: -None-
Search Terms: 7624302 or 7,624,302
Search Type: Terms and Connectors
Narrowed by:

Content Type	Narrowed by
Dockets	-None-

1:23cv640, Foras Technologies Ltd. V. Nissan Motor Company, Ltd. Et Al

US District Court Docket

United States District Court, Texas Western

(Austin)

This case was retrieved on 08/31/2023

Header

Case Number: 1:23cv640
Date Filed: 06/06/2023
Assigned To: Judge Robert Pitman
Nature of Suit: ***Patent*** (830)
Cause: ***Patent*** Infringement
Lead Docket: None
Other Docket: 6:23cv00336
Jurisdiction: Federal Question

Class Code: Open
Statute: 35:271
Jury Demand: Plaintiff
Demand Amount: $0
NOS Description: ***Patent***

Participants

Litigants

Foras Technologies Ltd.
Plaintiff

Attorneys

Jonathan Randy Yim
LEAD ATTORNEY;ATTORNEY TO BE NOTICED
BC Law Group, P.C.
200 Madison Avenue 24th Floor
New York, NY 10016
USA
516-359-9668 Email:Jyim@b-Clg.Com

Drew B Hollander
ATTORNEY TO BE NOTICED
BC Law Group, P.C.
200 Madison Avenue 24th Floor
New York, NY 10016
USA
212-951-0100 Email:Dhollander@b-Clg.Com

Seth Raymond Hasenour
ATTORNEY TO BE NOTICED
BC Law Group, P.C.
607 Terrell Hill Drive Ste Unit B
Austin, TX 78704
USA
516-359-9668 Email:Shasenour@b-Clg.Com

Brett E Cooper
ATTORNEY TO BE NOTICED
BC LAW GROUP, P.C.
200 Madison Avenue, 24th Floor
New York, NY 10016

Litigants

Nissan Motor Company, Ltd.
Defendant

ZF Friedrichshafen AG
Defendant

Attorneys

USA
516-359-9668 Email:Bcooper@b-Clg.Com

Proceedings

#	Date	Proceeding Text	Source
1	06/06/2023	COMPLAINT (Filing fee $ 402 receipt number ATXWDC-17522685), filed by Foras Technologies Ltd.. (Attachments: # 1 Exhibit 1, # 2 Exhibit 2, # 3 Exhibit 3, # 4 Exhibit 4, # 5 Exhibit 5, # 6 Exhibit 6, # 7 Exhibit 7, # 8 Exhibit 8, # 9 Exhibit 9, # 10 Exhibit 10, # 11 Exhibit 11, # 12 Civil Cover Sheet)(Cooper, Brett) (Entered: 06/06/2023)	
2	06/06/2023	Notice of Filing of ***Patent***/Trademark Form (AO 120). AO 120 forwarded to the Director of the U.S. ***Patent*** and Trademark Office. (Cooper, Brett) (Entered: 06/06/2023)	
3	06/06/2023	RULE 7 DISCLOSURE STATEMENT filed by Foras Technologies Ltd.. (Cooper, Brett) (Entered: 06/06/2023)	
4	06/06/2023	NOTICE Notice of Related Cases by Foras Technologies Ltd. (Cooper, Brett) (Entered: 06/06/2023)	
5	06/06/2023	REQUEST FOR ISSUANCE OF SUMMONS by Foras Technologies Ltd.. as to Nissan Motor Company, Ltd. (Cooper, Brett) (Entered: 06/06/2023)	
6	06/06/2023	REQUEST FOR ISSUANCE OF SUMMONS by Foras Technologies Ltd.. as to ZF Friedrichshafen AG (Cooper, Brett) (Entered: 06/06/2023)	
	06/06/2023	Case assigned to Judge Robert Pitman. CM WILL NOW REFLECT THE JUDGE INITIALS AS PART OF THE CASE NUMBER. PLEASE APPEND THESE JUDGE INITIALS TO THE CASE NUMBER ON EACH DOCUMENT THAT YOU FILE IN THIS CASE. (pg) (Entered: 06/09/2023)	
	06/06/2023	If ordered by the court, all referrals and consents in this case will be assigned to Magistrate Judge Howell (pg) (Entered: 06/09/2023)	
7	06/09/2023	Summons Issued as to Nissan Motor Company, Ltd. (pg) (Entered: 06/09/2023)	
8	06/09/2023	Summons Issued as to ZF Friedrichshafen AG. (pg) (Entered: 06/09/2023)	
9	06/09/2023	Report on ***Patent***/Trademark sent to U.S. ***Patent*** and Trademark Office with Complaint. (pg) (Entered: 06/09/2023)	

Patents

Number	Title	Issued	Class	Subclass
7,502,958	System and method for providing firmware recoverable lockstep protection	03/10/2009	714	12
7,624,302	System and method for switching the role of boot processor to a spare processor responsive to detection of loss of lockstep in a boot processor	11/24/2009	714	12
7,627,781	System and method for establishing a spare processor for recovering from loss of lockstep in a boot processor	12/01/2009	714	13

Copyright © LexisNexis CourtLink, Inc. All Rights Reserved.
*** THIS DATA IS FOR INFORMATIONAL PURPOSES ONLY ***

End of Document

US 20050172164A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2005/0172164 A1

Fox et al. (43) Pub. Date: Aug. 4, 2005

(54) AUTONOMOUS FAIL-OVER TO HOT-SPARE PROCESSOR USING SMI

(75) Inventors: **Thomas James Fox**, Apex, NC (US); **Eric R. Kern**, Durham, NC (US); **Michael Scott Rollins**, Durham, NC (US)

Correspondence Address:
DILLON & YUDELL LLP
8911 N. CAPITAL OF TEXAS HWY., SUITE 2110
AUSTIN, TX 78759 (US)

(73) Assignee: **International Business Machines Corporation**, Armonk, NY (US)

(21) Appl. No.: **10/761,721**

(22) Filed: **Jan. 21, 2004**

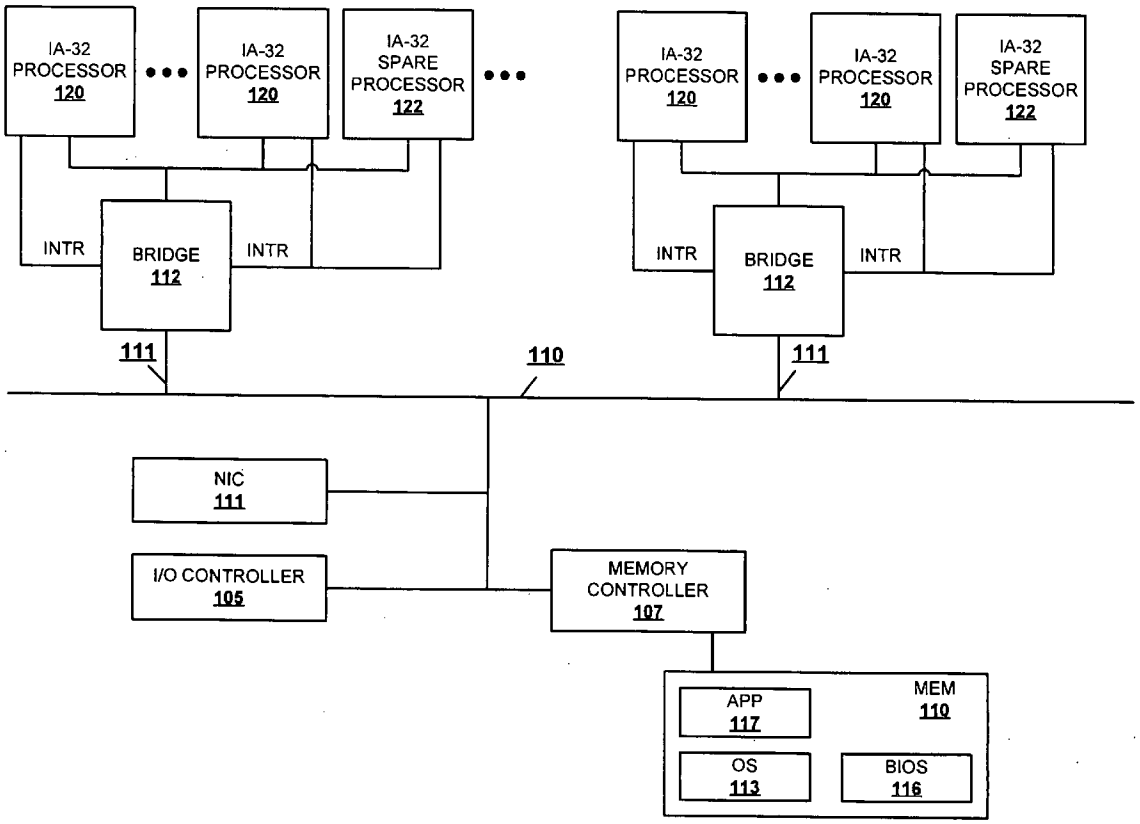
Publication Classification

(51) **Int. Cl.**[7] **G06F 11/00**

(52) **U.S. Cl.** **714/13**

(57) **ABSTRACT**

A method and system for dynamically replacing a failing processor in a server system configured with IA-32 architecture without requiring hardware changes to the IA-32 architecture or administrative effort. At least one processor of the multiprocessor system (MP) is initially provided as a reserve (or hot-spare) processor that remains in an idle, off, or low-power mode. While in that mode, the OS is prevented from initially utilizing the hot-spare processor. When a processor failure is detected, SMI code running on a good processor instructs the OS to hold off allocating processes to the failing processor. Contemporaneously, the SMI (and OS) activates and completes an initialization of the hot-spare processor to prepare it to begin receiving the held-off processes. Control is then returned to the OS, which updates the "active" processor list and allocates the threads that were running on the failing processor to the hot-spare processor.



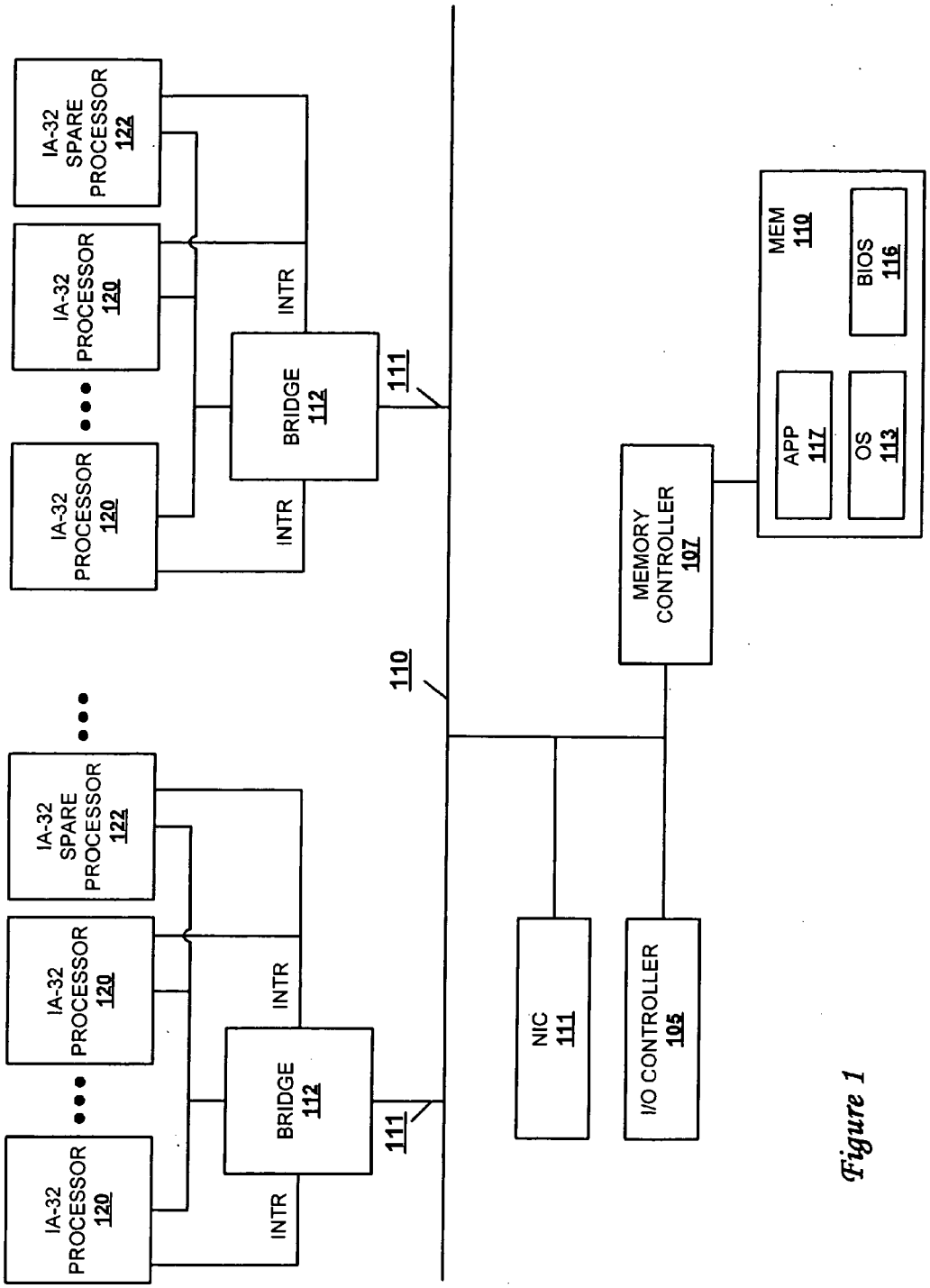

IA-32 SPARE PROCESSOR 122
IA-32 PROCESSOR 120
IA-32 PROCESSOR 120
INTR
BRIDGE 112
INTR
111
IA-32 SPARE PROCESSOR 122
IA-32 PROCESSOR 120
IA-32 PROCESSOR 120
INTR
BRIDGE 112
INTR
111
110
MEMORY CONTROLLER 107
MEM 110
APP 117
OS 113
BIOS 116
NIC 111
I/O CONTROLLER 105

Figure 1

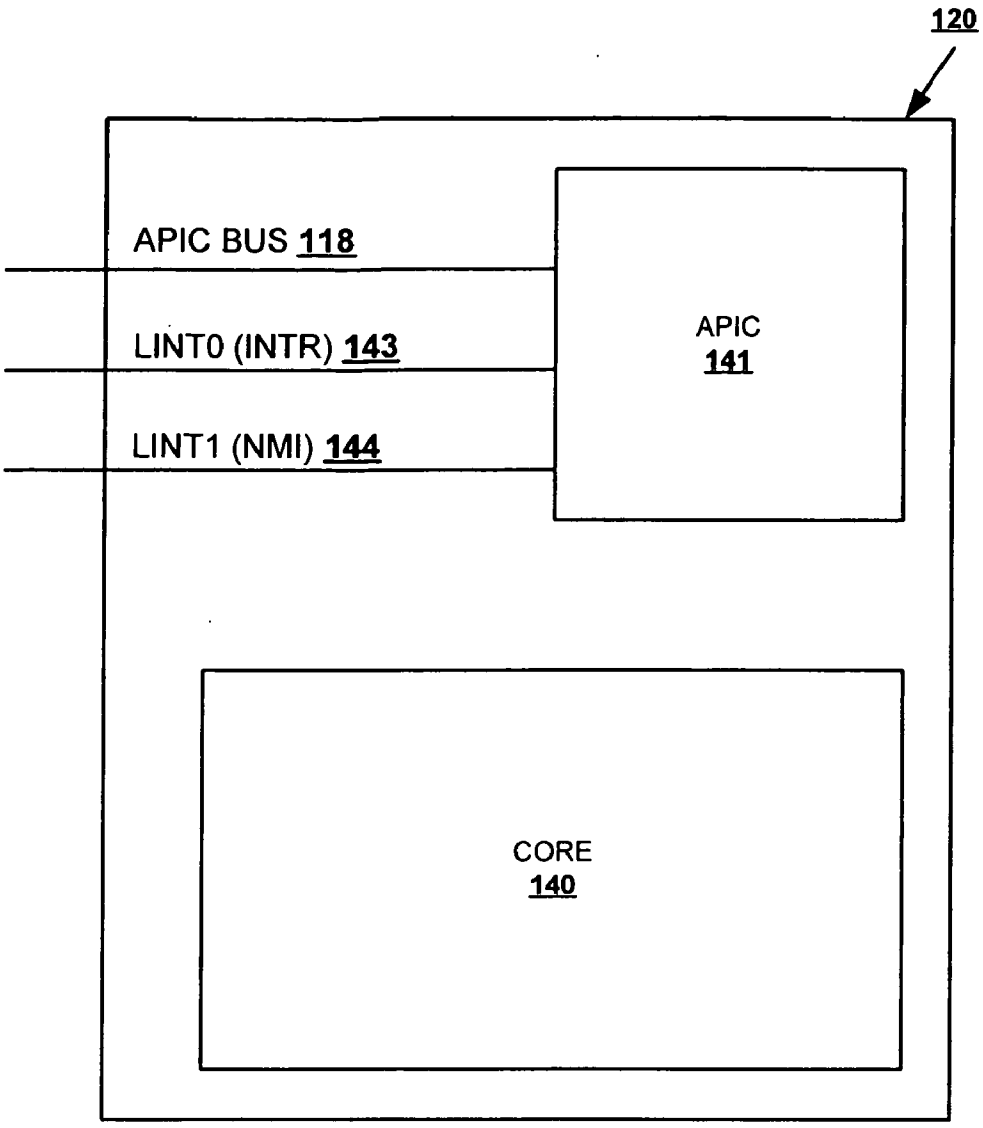

120
APIC BUS 118
LINT0 (INTR) 143
LINT1 (NMI) 144
APIC
141
CORE
140

Figure 2A

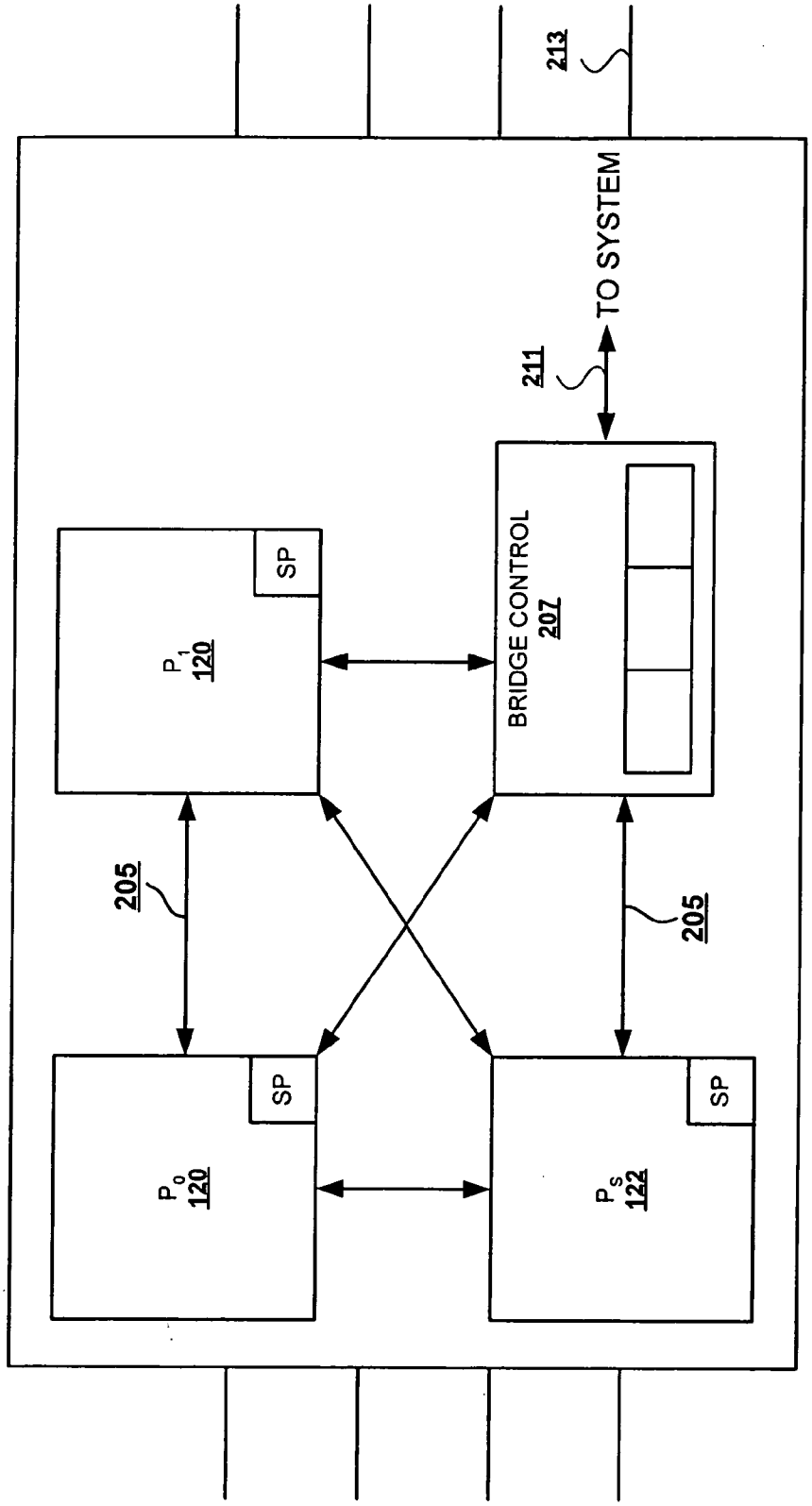

213
TO SYSTEM
211
BRIDGE CONTROL
207
SP
P_1
120
205
205
SP
P_o
120
SP
P_s
122

Figure 2B

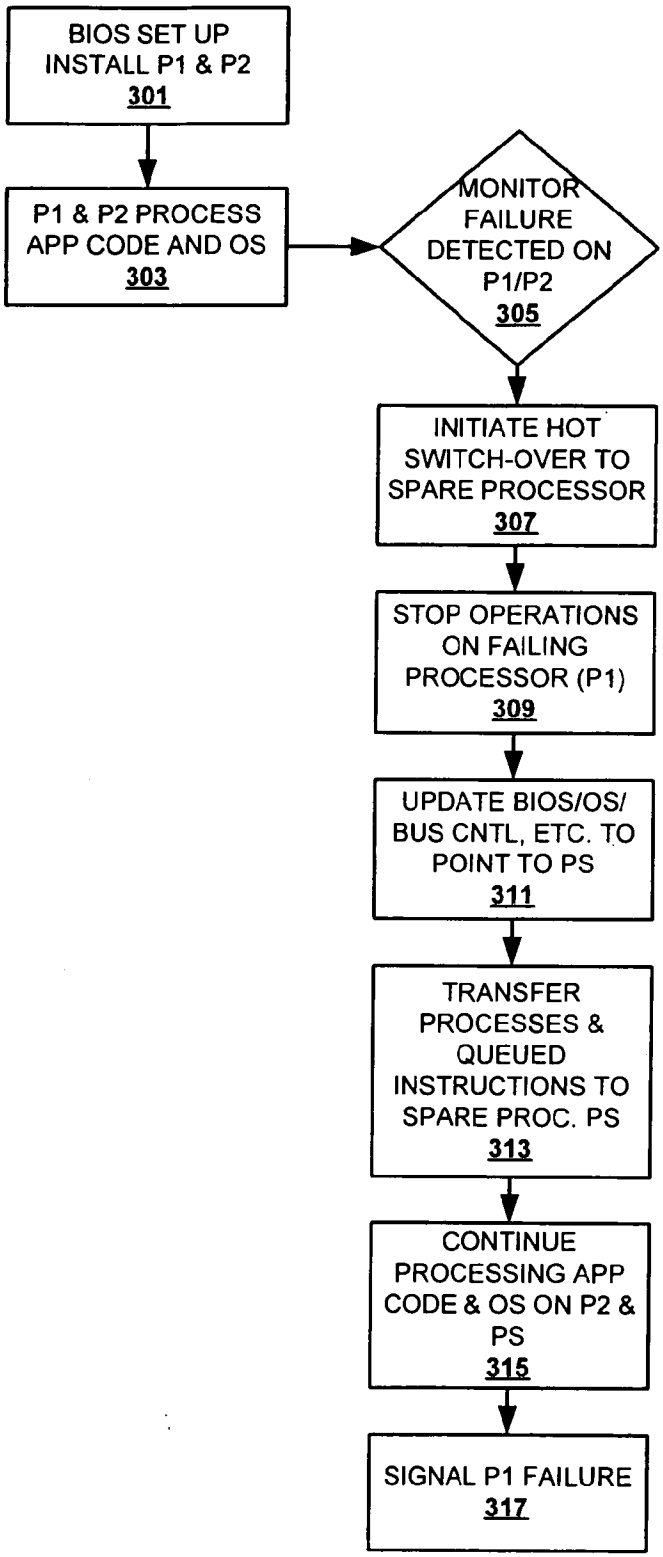

BIOS SET UP
INSTALL P1 & P2
301
P1 & P2 PROCESS
APP CODE AND OS
303
MONITOR
FAILURE
DETECTED ON
P1/P2
305
INITIATE HOT
SWITCH-OVER TO
SPARE PROCESSOR
307
STOP OPERATIONS
ON FAILING
PROCESSOR (P1)
309
UPDATE BIOS/OS/
BUS CNTL, ETC. TO
POINT TO PS
311
TRANSFER
PROCESSES &
QUEUED
INSTRUCTIONS TO
SPARE PROC. PS
313
CONTINUE
PROCESSING APP
CODE & OS ON P2 &
PS
315
SIGNAL P1 FAILURE
317

Figure 3

AUTONOMOUS FAIL-OVER TO HOT-SPARE PROCESSOR USING SMI

BACKGROUND OF THE INVENTION

[0001] 1. Technical Field

[0002] The present invention relates generally to data processing systems and in particular to system response to processor failure in a multi-processor data processing system. Still more particularly, the present invention relates to a method and system for dynamically activating a spare processor when processor failure is detected in a multi-processor data processing system.

[0003] 2. Description of the Related Art

[0004] Conventional data processing systems are often configured with multiple processors interconnected to each other and other system components. These multiple processors may exist on a single chip or may be manufactured on separate chips. The processors operate in tandem to efficiently complete tasks associated with application code being executed. Those skilled in the art are familiar with the various configuration and operations of multiprocessor systems (MPs).

[0005] Occasionally, while a system is running and the processors are executing application code, one (or more) of the processors may fail (i.e., begin to provide inaccurate processing results and/or begin operating outside of pre-established or "ideal" operating parameters, etc.). When such processor failure occurs, the system's technician has to replace the failing processor (or processor chip) with a new one in order to maintain the level of processing desired for the system. This changing of processors is normally a manual operation, which requires the technician to halt all executing processes (across the system including the operating system (OS)), shut down the MP, obtain a new/replacement processor, complete the switch out of the failing processors, reboot the system, and then restart the executing processes across the MP.

[0006] During the system reboot, the replacement processor is recognized by the MP's BIOS (basic input/output system) and activated for operation within the system. Conventionally, the failed (or failing) processor is physically detached (or removed) from the system bus (or interconnect), and the replacement processor is connected (plugged-in) to the interconnect in place of the removed processor. This replacement method is convenient when non-critical processes are being completed on the MP; however, the time required to replace the processor and downtime in processing is un-acceptable for critical processes that require continuous up-time of the MP.

[0007] Also, with current replacement methods, a separate replacement processor is required to be plugged-in after the failure condition is detected. This requires a technician to swap out the failed processor with the replacement, and as described above, the OS and executing processes are halted until the swap of the processors is completed.

[0008] The traditional method of responding to processor failure severely limits the ability of larger systems (e.g., multiprocessor server systems) with non-failing processors to continue executing despite the presence of the failing processor. In lager server systems that require continuous up-time, replacement of a failing processor has to be completed without shutting down the entire system. Typically, in a server system, when a processor begins to fail, the processor must be taken out of the processing pipeline and replaced by another processor to avoid the entire MP crashing. Depending on the built-in redundancy and complexity of the MP, such a temporary removal may have wide ranging effects, from slightly degrading the overall performance of the MP to temporarily removing the MP from service.

[0009] Currently, manufacturers of server systems provide different types of server architectures, with common architectures being the S/390 architecture and the Intel Architecture-32 bit (IA-32). The S/390 architecture has a machine instruction for switching to a backup processor, while IA-32 does not have a similar machine instruction. Rather, IA-32 is designed with the functionality to generate an SMI (Systems Management Interrupt) after a CPU fault. Generation of SMIs for standard system management tasks is unique to the IA-32 and the process is described in detailed in U.S. Pat. No. 6,625,679.

[0010] Realizing that shutting down the entire MP and then restarting all processes is an unacceptable method of handling single processor failures, manufacturers designed some conventional MPs with a failure response mechanism that involves a hot-spare processor and hardware changes to the system architecture to support processor failure conditions. Implementation of the failure response mechanism is based on the type of server architecture.

[0011] U.S. Pat. No. 6,115,829 provides a hot spare processor for solving processor problems relating to the S/390 architecture. The solution involves the utilization of a hardware instruction built into the processor that is usable only by millicode. Since the problems in the S/390 architecture are specific to that architecture, the above solution is not available to the IA-32 architecture, which has a different processor configuration and exhibits a different set of processor problems. Those skilled in the art are familiar with the functional and architectural differences in the two types of architectures and appreciate that different response methods unique to each architecture must be implemented for processor failure.

[0012] As another example, U.S. Pat. No. 4,819,232 provides a hardware instruction for software programs to utilize when completing fault recovery to a spare processor from the primary processor. Implementation of this process in an IA-32 system would require architecture (hardware) changes to current IA-32. Another patent, U.S. Pat. No. 5,155,729 provides a redundant processor that engages in a ping-ponging process with the primary processor during a hot swap condition. This process is also specific to the S/390 architecture and not provided within the IA-32 architecture. Finally, U.S. Pat. No. 6,370,657 places the system into standby prior to hot-swapping the processors; however, the response mechanism does not provide a hot-spare nor does it provide a means to keep the OS running during the switch between processors.

[0013] With server systems, processor downtime is an undesirable condition, and thus a fast fault-response mechanism/scheme is required. It would be desirable for such a scheme to include the ability to detect when one of the multiple processors has failed. Additionally, when a processor failure has been detected, it would also be desirable for

the fault-response mechanism to quickly respond to the failure by providing a replacement processor without the system having to suspend processing and with minimal disruption to the overall system. The present invention recognizes that it would be desirable to provide a processor failure response mechanism that provides a replacement processor in a seamless manner so that executing processes and the OS continue executing during dynamic replacement of the failed processor.

SUMMARY OF THE INVENTION

[0014] Disclosed is a method and system for dynamically replacing a failing processor in a server system configured with IA-32 architecture without requiring hardware changes to the IA-32 architecture or administrative effort. The method enables a system to remain operational, i.e., continue executing operating system (OS) processes and application processes on the other processors, while the failing processor is automatically de-activated and a replacement processor is dynamically activated. Negligible system downtime occurs during the transfer of executing threads/processes to the replacement processor, unlike other methods which require temporary halting of the OS processing and, in some instances, halting all other processes across the system.

[0015] Implementation of the invention involves the utilization of Systems Management Interrupt (SMI) functionality provided within basic input output system (BIOS) of the IA-32 architecture. At least one processor of the multiprocessor system (MP) is initially provided as a reserve (or hot-spare) processor that remains in an idle, off, or low-power mode. While in that mode, the OS is prevented from initially utilizing the hot-spare processor. When a processor failure is detected, SMI code running on a good processor instructs the OS to hold off allocating processes to the failing processor. Contemporaneously, the SMI (and OS) activates and completes an initialization of the hot-spare processor to prepare it to begin receiving the held-off processes. Control is then returned to the OS, which updates the "active" processor list and allocates the threads that were running on the failing processor to the hot-spare processor.

[0016] Thus, a processor is able to autonomically fail-over to a hot-spare processor without affecting the OS or executing applications. The invention substantially eliminates the problems related to high availability during single processor failure without crashing all of the processors. Further, the invention completes this processor replacement without requiring specialized OS or middleware features. Finally, the invention provides a method for dynamic, hot swap of spare processors without the need of processor architecture changes. That is, no special hardware instruction is built into the processor, as in other methods.

[0017] The above as well as additional objects, features, and advantages of the present invention will become apparent in the following detailed written description.

BRIEF DESCRIPTION OF THE DRAWINGS

[0018] The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself however, as well as a preferred mode of use, further objects and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

[0019] **FIG. 1** is a block diagram of a multiprocessor computer system (MP) designed according to IA-32 architecture and a spare processor according to one illustrative embodiment of the present invention;

[0020] **FIG. 2A** is a block diagram depicting two major internal components of one of the IA-32 processors of **FIG. 1A**, according to one illustrative embodiment of the present invention;

[0021] **FIG. 2B** is a block diagram of a multiprocessor chip designed with dual IA-32 processors and a spare IA-32 processor to enable dynamic switching out of a filed processor during failure response in accordance with one embodiment of the present invention; and

[0022] **FIG. 3** is a flow chart illustrating the process of detecting and responding to processor failure with the spare processor configuration of **FIGS. 1 and 2**B according to one embodiment of the present invention.

DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENT(S)

[0023] The present invention provides a method for dynamically replacing failing processors in a IA-32 configured multiprocessor computer system (MP) by providing the MP with a spare processor that is initially held-off and is later brought into the group of executing processors whenever one of the main processors is detected as failing. The invention provides a substantially seamless transfer of executing processes from the failing processor to the replacement processor, without halting the execution of the operating system (OS) and/or applications on the overall MP. That is, a failing processor is able to autonomically fail-over to a hot-spare processor without affecting the OS or executing applications. The failure response method of the invention is completed without requiring special hardware instructions built into the processors.

[0024] With reference now to the figures and in particular to **FIG. 1**, there is illustrated a multiprocessor system (MP) designed with a spare processor and other components that enable the implementation of the various features of the invention. MP **100** is designed according to IA-32 architecture and comprises an IA-32 processor system **120** coupled to system components by a bridge **112**. The IA-32 processor system **11** may include one or more processors **120, 122** that perform various computing functions. The processors **14** are coupled to a common bus **114**, and may be coupled to a local advanced programmable interrupt controller (APIC) bus (described below). The processors **120, 122** share access to the common bus **114**, and may also share other resources such as memory, input/output (I/O) devices, and interrupt handlers, for example. The system components provide enhanced functionality, and can be used with the existing IA-32 processors **120, 122**, provided that appropriate hardware and/or software is used to ensure compatibility between the IA-32 processor system and the system components

[0025] MP **100** also comprises a spare processor **122**. The processors are interconnected to each other via processor interconnect **114**. Each processor **120, 122** comprises service processor logic (not shown), which communicates with

other service processors of other processors as well as bus/interconnect logic (e.g., bridge **112**) to allow the processors **120**, **122** to operate in a coherent manner with each other and the rest of the MP **100**.

[**0026**] As shown by **FIG. 2A**, each processor **120**, **122** includes a core unit **140** that controls processing on the processor **120**, **122** and an APIC **141** that processes external and normal interrupts that the processor **120**, **122** may receive at its interrupt pins **143** and **144**. APIC **141** also includes a connection to the APIC bus **18**, used to deliver interrupts to the application processors. The interrupt pins **143** and **144** may be programmed to receive specific types of interrupts.

[**0027**] The system components include one or more I/O controllers **105**, which are coupled to a memory controller (MC) **107**. The I/O controllers **105** are coupled to I/O device buses **109**. The buses **109** may include a peripheral computer interface (PCI) bus, for example, the buses **109** may connect standard computer peripherals such as a keyboard.

[**0028**] Coupled to MC **107** is a system memory **110**, within which is stored software for controlling the MP and executing application processes. These software include: the OS **113**, BIOS **116**, which includes SMI, and program applications **117**. SMI notifies the OS of additional processor resources within the MP (i.e., increase/decrease in number of processors) as well as addition/removal of other system resources (i.e., memory and I/O, etc.).

[**0029**] A system bus **110** connects the memory controller **107** and other components to IA-32 processors **120**, **122**, via bridges **112**, and processor buses **114**. The IA-32 processors **120**, **122** and the bridges **112** are grouped at nodes **111**.

[**0030**] The bridge **112** takes the interrupts that are being delivered over the system bus **110** and signals the interrupt to the appropriate IA-32 processor **120**, **122**. To ensure that a particular interrupt reaches the appropriate IA-32 processor **120**, **122**, all interrupt transactions on the system bus **110** contain target node **111** and IA-32 processor identification.

[**0031**] Software may initialize the I/O controllers **105** to transfer an interrupt transaction in a manner that identifies the IA-32 processor **120**, **122** to be interrupted. The bridge **112** that is connected to the same processor bus **114** as the destination IA-32 processor **120**, **122** recognizes the interrupt transaction, and asserts an interrupt pin at the targeted IA-32 processor **120**, **122**.

[**0032**] During operation, the OS and software code from the program application are executed by the processors to complete specific tasks or functions associated with the application. The processors are interconnected to the other components (memory, etc.) via system interconnect **110**. Interconnect fabric **110** includes wires and control logic for routing communication between the components as well as controlling the response of the OS of MP **100** to changes in the hardware configuration. In the present invention, BIOS comprises processor failure response logic **207**, which generates the SMI, and configuration setting logic **209**.

[**0033**] **FIG. 2B** provides a block diagram representation of a multiprocessor chip designed with main IA-32 processors **120** and a spare processor **122**. Also provided with MP chip **200** is a register **209** for tracking active processors on MP chip **200**, as described below. In one alternate implementation, processors **120**, **122** are interconnected to each other via processor-to-processor buses **203**. Processors **120**, **122** are also connected to bridge (bus controller) **207** via individual control buses **205**. Control buses enable transfer of control data from the bus controller that indicates which processor is active within MP chip **200**. Control bus then provides a connector **211** that couples MP chip **200** to larger processing system such as MP **100** of **FIG. 1** via pins **213**.

[**0034**] When the MP is initially activated for processing, one IA-32 processor is held off as a spare processor. Holding off the processor involves having the basic input output system (BIOS) recognize the processor as a spare during BIOS boot (i.e., power on self test (POST), etc. of the MP and having the BIOS respond by not enabling the spare processor to be initially allocated for OS and application processing. This particular set-up of the MP may require some administrative input and support. However, in one embodiment, the processor initialization/identification code includes a software tag to identify the processor as a spare processor to the OS and other components. The other processors are initiated and identified to the OS for regular processing.

[**0035**] In one embodiment, during boot, BIOS utilizes advanced configuration and power interface (ACPI) specification and/or the S3 state to put the processor in a low-power, standby, or "off" state. The S3 state is a computer sleep state defined by the ACPI specification in which the system is not powered nor running, but the memory remains intact, powered and in a continuous refresh cycle. This initial removal of the spare processor from the group of executing processors occurs before the BIOS hands over control to the OS. In another embodiment the system is made to enable the "hot spare processor" feature on-the-fly by the OS with a driver-OS-SMI handshake. Tracking which processors are active may be completed by an software table of active processors, which is available to be read by the OS, when determining work load allocation to each of the active processors.

[**0036**] Notably, the present invention maintains system uptime even while a processor in the MP is failing. This feature of the invention is particularly applicable to operations in a critical server environment that requires continuous uptime of the system.

[**0037**] The spare processor remains hidden until one of the primary processors fails and the spare processor is needed to replace the failing processor. As illustrated in **FIG. 2B**, the spare processor may be built on the same board as the main processors. However, in another implementation, the spare processor is provided on a separate processing board to make serviceability easier. For example, with the spare processor operation for Series x440 models, the bottom-most board with CPU's is identified as the ideal board on which to reserve a hot-spare CPU(s), since the bottom CPU board is not as accessible. This enables a failed CPU to be disabled while the OS remains running.

[**0038**] The spare processor is interconnected similarly to each primary processor with the exception that BIOS does not initialize the spare nor initially present the spare processor to the OS following the POST. During POST, BIOS hides the spare processor from the OS. This is accomplished either through a setup of the MP table and/or an ACPI setup. The BIOS brings the spare processor all the way online, then

puts it into a spin lock (i.e., a tight software loop in the SMI handler). This has the effect of completely hiding the processor from the OS until the spare processor is needed. Once processor failure occurs and the spare processor is activated, the spare processor is then placed under complete OS control.

[0039] Possible triggers for the failure-response features of the invention includes the following: (1) CPU thermal trends are observed by a system management utility, which may be hardware (HW), firmware (FW), and/or software (SW). In one implementation, the system manager utility also provides an SMI handler itself; (2) The CPU temperature goes over or near some operational breaking point; and (3) The system management entity causes an SMI to be invoked, indicating which CPU is to be off-loaded to the hot spare CPU.

[0040] Implementation of the invention involves the utilization of Systems Management Interrupt (SMI) functionality provided within basic input output system (BIOS) of the IA-32 systems. At least one processor of the multiprocessor system is reserved as a hot-spare processor that remains in an idle, off, or low-power mode. While in that mode, the OS is prevented from initially utilizing the hot-spare processor. When a processor failure is detected, SMI code running on a good processor instructs the OS to hold off allocating processes to the failing processor. Contemporaneously, the SMI (and OS) activates and completes an initialization of the hot-spare processor to prepare it to begin receiving the held-off processes. Control is then returned to the OS, which updates the "active" processor list and allocates the threads that were running on the failing processor to the hot-spare processor.

[0041] The SMI is generated by hardware during various processor failure conditions. During an SMI, the system state of each processor is completely saved to memory and the caches and pipelines are flushed. A BIOS interrupt handler (i.e., the SMI handler) then takes control of the system until a resume (RSM) instruction is performed by the OS. The SMI handler is utilized to allow BIOS to swap out processors. The BIOS places the failing processor in a halt state and contemporaneously brings the standby processor out of the held-off state and into an active state. Additionally, the BIOS switches the memory save state from the old processor to the new processor. In one embodiment, the above processes are accomplished by providing a software register of active processors in the OS and setting the register to reflect the spare processor as active and the failing processor as inactive.

[0042] In the present invention also, SMI allows BIOS to detect a failure and control the reallocation of processing from a good CPU to the hot spare CPU. The SMIs are generated through hardware, and an internal monitoring mechanism is provided for each processor or group of processors that generates the SMI. In the described embodiment, the specific processor-fault SMI is generated when a processor is failing. The SMI identifies to the OS and the BIOS which one of the processors is failing. SMIs are invoked periodically and/or asynchronously by the hardware or other system events, and possibly by software control. A description of the functionality and utilization of the SMI and SMI handler is described in detail in U.S. Pat. No. 6,463,492.

[0043] Whenever a CPU fault occurs, an SMI is automatically generated by the existing hardware, as described in the above referenced patent. The SMI is utilized as a trigger that initiates the activation of the spare processor by the BIOS. Notably, unlike with other conventional methods, no hardware switching logic is needed to accomplish the fail-over process of the invention. That is, the invention provides a software-only solution for processor failure response.

[0044] Thus, a processor is able to autonomically fail-over to a hot-spare processor without affecting the OS or executing applications. The invention substantially eliminates the problems related to high availability during single processor failure without crashing all of the processors. Further, the invention completes this solution in a way that also does not require specialized OS or middleware features. Finally, the invention provides a method for hot swap spare processors without the need of processor architecture changes. That is, no special hardware instruction is required built into the processor as in other methods.

[0045] **FIG. 3** is a flow chart illustrating the process of failure detection and response according to one implementation of the invention. For simplicity, the process is described with reference to the MP and processor configuration of **FIGS. 1 and 2**A, 2B. As provided at block **301**, the process begins with the BIOS setting up and installing the main processors, P**0** and P**1**, during the initial POST of the MP. The presence of the spare processor is registered in the BIOS and that processor is initially left in power-down (held-off) or inactive mode. The OS and applications are not provided with an indication of the spare processor, so no operations are initially allocated to that processor. The primary processors commence processing application code and OS operations on the MP, as shown at block **303**. The system monitors the processors for a detection of failure on one of the processors, and a determination is made at block **305** whether a failure is detected. In one implementation, the processors detect failure within their subsystem, generating a processor fault SMI, and message the failure to the SMI handle. If no failure occurs, the processing of application and OS continues as normal on the main processors.

[0046] If failure is detected, however, the receipt of a processor fault SMI by the OS immediately activates a switch over response to replace the failing processor with the spare processor as indicated at block **307**. The spare processor is brought out of the held-off state. Concurrently, processes (instructions, threads, etc.) initially allocated to the failing processor are held by the OS and processing on the failing processor is stopped as shown at block **309**. Once the spare processor has been successfully brought on line, the BIOS (table) is updated and the OS, bus controllers, etc. are messaged to change their configurations to reflect a processor change from the failing processor to the spare processor as shown at block **311**. The processes initially allocated to the failing processor are sent to the spare processor as indicated at block **313**, and the application code and OS continue processing on the processors that are now functional (i.e., the other main processor and the spare processor) as provided at block **315**. Notably, for this seamless transfer to be possible, all processors in the system must be identical or process instructions in an identical manner. Otherwise a degradation or improvement in processing may become noticeable and affect overall throughput. Once the fail-over to the spare processor has been

completed, a signal/message is generated indicating (to the administrator) that the processor has failed and is replaced with a spare, as shown at block **317**. Then the process ends as depicted at block **319**.

[**0047**] The invention provides a method for hot swap spare processors without the need of processor architecture changes, i.e., without requiring a hardware instruction built into the processor. The invention provides high availability of processing resources and eliminates delays due to personal switching out failing processors manually. Further, the invention completes this solution in a way that also does not require specialized OS or middleware features.

[**0048**] The present invention provides several advantages over other available methods including: (1) no specialized software is needed because SMI is in BIOS, and the reaction to a failing processor is automatic. Also, (2) using an SMI, a service processor is able to identify a failing processor & halt it's execution from an outside perspective, unlike the limitation with a software implementation. Furthermore, (3) an alert about a hot-spare processor switching event can be generated.

[**0049**] The other software methods do not utilize SMIs or SMI functionality and encounter several drawbacks, including: (1) customers must purchase the OS/software license that provides the hot-spare-processor feature, since not all OS support this feature; and (2) the OS has to makes the decision to switch to a hot spare processor may be running decision threads atop the faulting processor, leading to incorrect code execution.

[**0050**] While the invention has been particularly shown and described with reference to a preferred embodiment, it will be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention.

What is claimed is:

1. In an multiprocessor data processing system (MP) configured according to IA-32 architecture, a method for dynamically providing spare processor resources when an operating processor fails, said method comprising:

holding-off a spare processor during a POST (power on self test), wherein said spare processor is available within said MP along with at least two operating processors, and said spare processor is not allocated any processing load by the operating system (OS) following the POST;

when one of the operating processors is determined to be failing, dynamically activating said spare processor to replace the failing operating processor, wherein the processing load of the failing processor is automatically sent to the spare processor for processing.

2. The method of claim 1, wherein said MP includes a basic input/output system (BIOS) and a processor register linked to said BIOS, which indicates which processors among said operating processors and said spare processors are currently available to said OS for allocating load, wherein said holding off of the spare processor comprises:

setting a bit within said register corresponding to each of said operating processors to an active state during said initial POST, wherein said active state indicates to said OS that the corresponding operating processor is available for allocating load; and

setting a bit corresponding to said spare processor an inactive state.

3. The method of claim 2, wherein said dynamically activating step comprises:

re-setting a bit corresponding to the failing processor to an inactive state; and

setting said bit corresponding to said spare processor to an active state.

4. The method of claim 1, wherein said holding off step comprises:

placing said spare processor in a low-power, standby state during system boot utilizing advanced configuration and power interface (ACPI) of the BIOS; and

subsequently handing off control of said MP from said BIOS to the OS.

5. The method of claim 1, wherein said holding off step comprises:

placing said spare processor in an S3 state during system boot utilizing advanced configuration and power interface (ACPI) of the BIOS; and

subsequently handing off control of said MP from said BIOS to the OS.

6. The method of claim 1, wherein said holding off step further comprises:

bringing said spare processor online during BIOS boot and then placing the spare processor into a spin lock.

7. The method of claim 1, further comprising generating a message signaling said processor failure to a system administrator.

8. The method of claim 1, wherein said dynamically activating step comprises completing a driver-OS-SMI handshake operation.

9. The method of claim 1, further comprising:

activating a system management interrupt (SMI) when said failing processor is determined to be failing, wherein said SMI messages said OS to hold off allocation of said processing load to said failing processor.

10. The method of claim 9, further comprising:

saving a system state of each processor to memory;

activating an SMI handler to temporarily take control of said system from said OS while said spare processor is being activated;

swapping out the failing processor for the spare processor;

updating system configuration parameters, including bus controller, memory and OS parameters, to reflect switch over to said spare processor from said failing processor; and

restoring control to said OS once said spare processor has been activated.

11. The method of claim 9, further comprising enabling OS updates to an active processor lists, wherein said OS begins allocating threads previously running on said failed processor to said spare processor.

12. A multiprocessor data processing system (MP) configured according to IA-32 architecture, comprising:

a plurality of processors having at least two main operating processors and a spare processor;

a memory coupled to said plurality of processors;

an operating system (OS) that controls work allocation to each of said plurality of processors;

a basic input output system (BIOS) that supports system management interrupt (SMI);

means for holding-off the spare processor during a POST (power on self test), wherein said spare processor is not allocated any processing load by the operating system (OS) following a power-on self test (POST); and

means, when one of the operating processors is determined to be failing, for dynamically activating said spare processor to replace the failing operating processor, wherein the processing load of the failing processor is automatically sent to the spare processor for processing.

13. The MP of claim 12, further comprising:

a processor register linked to said BIOS, which indicates which processors among said operating processors and said spare processors are currently available to said OS for allocating load; and

means for setting a bit within said register corresponding to each of said operating processors to an active state during said initial POST, wherein said active state indicates to said OS that the corresponding operating processor is available for allocating processing load, wherein further a bit of said spare processor is set to an inactive state.

14. The MP of claim 13, wherein said means for dynamically activating comprises:

means for re-setting a bit corresponding to the failing processor to an inactive state; and

means for setting the bit corresponding to said spare processor to an active state.

15. The MP of claim 12, wherein said means for holding off comprises:

means for placing said spare processor in a low-power, standby state during system boot utilizing advanced configuration and power interface (ACPI) of the BIOS; and

means for subsequently handing off control of said MP from said BIOS to the OS.

16. The MP of claim 12, wherein said means for holding off comprises:

means for placing said spare processor in an S3 state during system boot utilizing advanced configuration and power interface (ACPI) of the BIOS; and

means for subsequently handing off control of said MP from said BIOS to the OS.

17. The MP of claim 12, wherein said means for holding off further comprises BIOS means for bringing said spare processor online and then placing the spare processor into a spin lock.

18. The MP of claim 11, further comprising generating a message signaling said processor failure to a system administrator.

19. The MP of claim 12, wherein said means for dynamically activating comprises means for completing a driver-OS-SMI handshake operation.

20. The MP of claim 1, further comprising:

means for activating a system management interrupt (SMI) when said failing processor is determined to be failing, wherein said SMI messages said OS to hold off allocation of said processing load to said failing processor.

21. The MP of claim 20, further comprising:

means for saving a system state of each processor to memory;

means for activating an SMI handler to temporarily take control of said system from said OS while said spare processor is being activated;

means for swapping out the failing processor for the spare processor;

means for updating system configuration parameters, including bus controller, memory and OS parameters, to reflect switch over to said spare processor from said failing processor; and

means for restoring control to said OS once said spare processor has been activated.

22. The MP of claim 21, further comprising means for enabling OS updates to an active processor lists, wherein said OS begins allocating threads previously running on said failed processor to said spare processor.

23. The MP of claim 12, wherein further:

said plurality of processors are configured with said at least two main processors on a first processing board and said spare processor on a second processing board, wherein said second processing board is a processing board among a plurality of processing boards containing a processor within the MP that is most difficult to access.

* * * * *

US005915082A

United States Patent [19]

Marshall et al.

[11] Patent Number: 5,915,082

[45] Date of Patent: Jun. 22, 1999

[54] **ERROR DETECTION AND FAULT ISOLATION FOR LOCKSTEP PROCESSOR SYSTEMS**

[75] Inventors: **Joseph R. Marshall**; **Dale G. Langston**, both of Manassas, Va.

[73] Assignee: **Lockheed Martin Corporation**, Bethesda, Md.

[21] Appl. No.: **08/660,640**

[22] Filed: **Jun. 7, 1996**

[51] **Int. Cl.**[6] **G06F 11/08**

[52] **U.S. Cl.** **395/182.09**

[58] **Field of Search** 395/185.01, 185.02, 395/185.05, 182.09, 182.1, 200.19; 370/228, 221, 217, 218, 220

[56] **References Cited**

U.S. PATENT DOCUMENTS

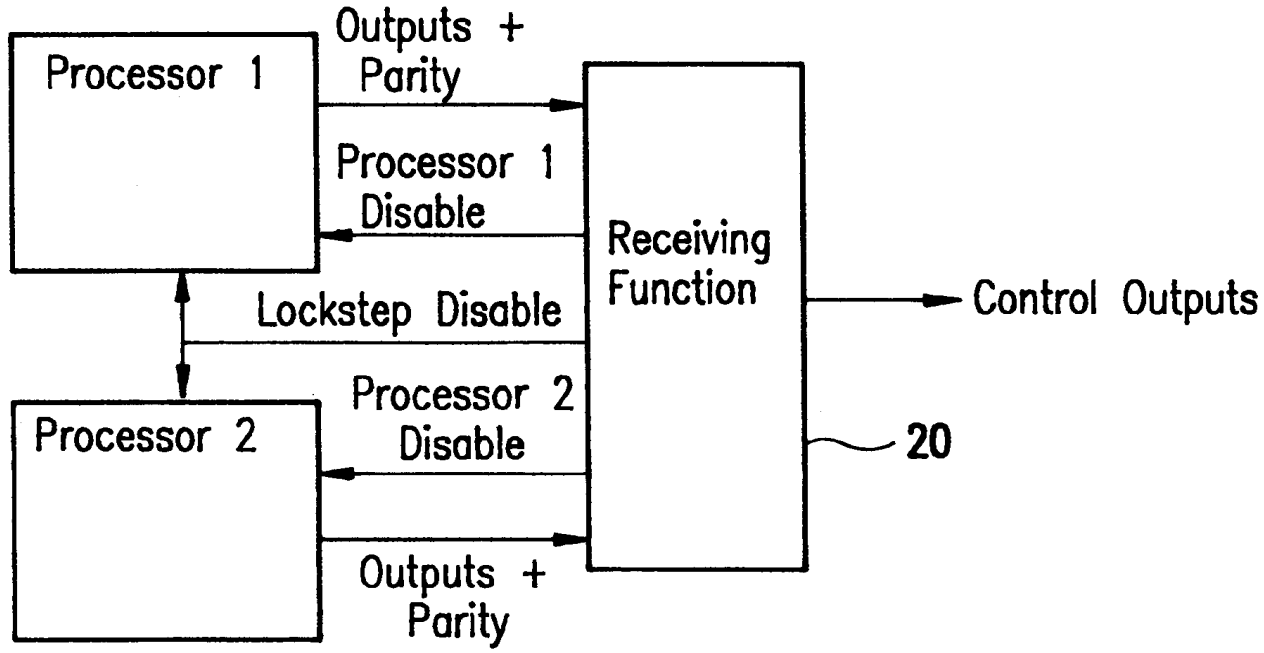
4,438,494	3/1984	Budde et al.	364/200
4,503,535	3/1985	Budde et al.	371/11
4,823,256	4/1989	Bishop et al.	364/200
4,916,704	4/1990	Bruckert et al.	371/68.3
5,138,708	8/1992	Vosbury	395/575
5,226,152	7/1993	Klug et al.	395/575
5,233,615	8/1993	Goetz	371/36
5,249,187	9/1993	Bruckert et al.	371/68.1
5,249,188	9/1993	McDonald	371/68.3
5,255,367	10/1993	Bruckert et al.	395/200
5,313,575	5/1994	Beethe	395/159
5,339,408	8/1994	Bruckert et al.	395/575
5,353,436	10/1994	Horst	395/550
5,640,521	6/1997	Whetsel	395/311

Primary Examiner—Joseph E. Palys
Assistant Examiner—Pierre E. Elisca
Attorney, Agent, or Firm—Lane, Aitken and McCann

[57] **ABSTRACT**

A lockstep processor system adds error detection, isolation, and recovery logic to one or more lockstep processor system functions; namely, control outputs, processor inputs, I/O busses, memory address busses, and memory data busses.

2 Claims, 8 Drawing Sheets



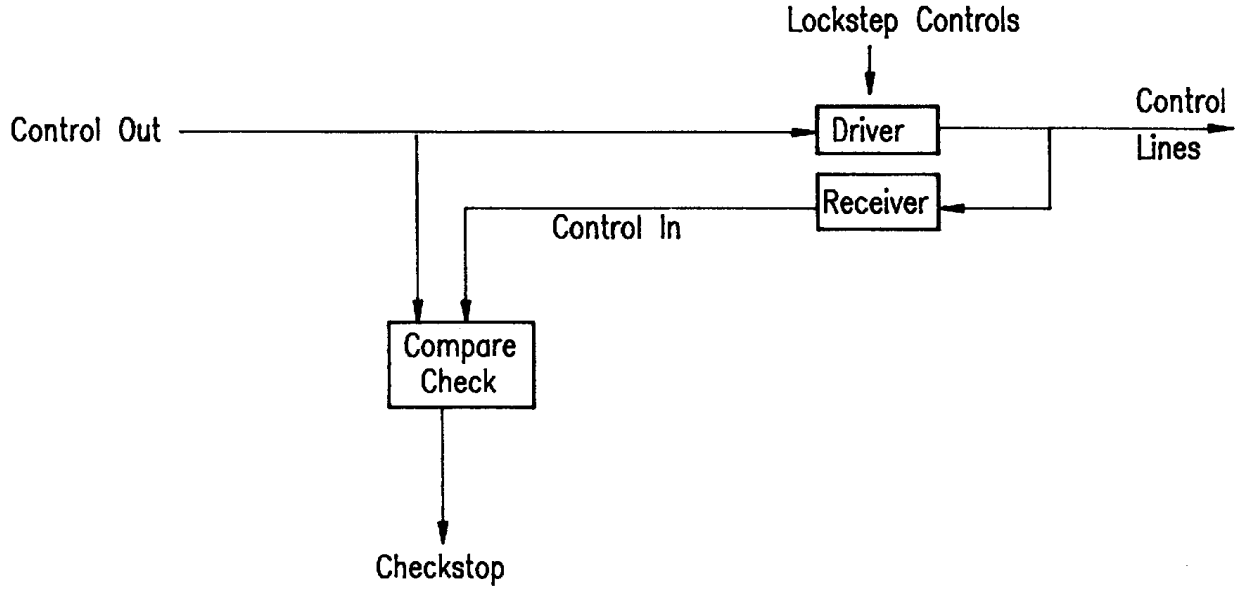


FIG.1
PRIOR ART

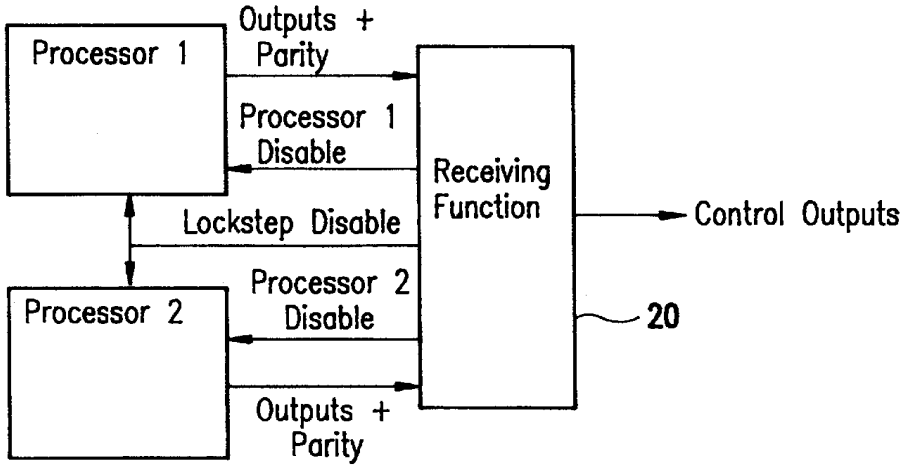


FIG.2

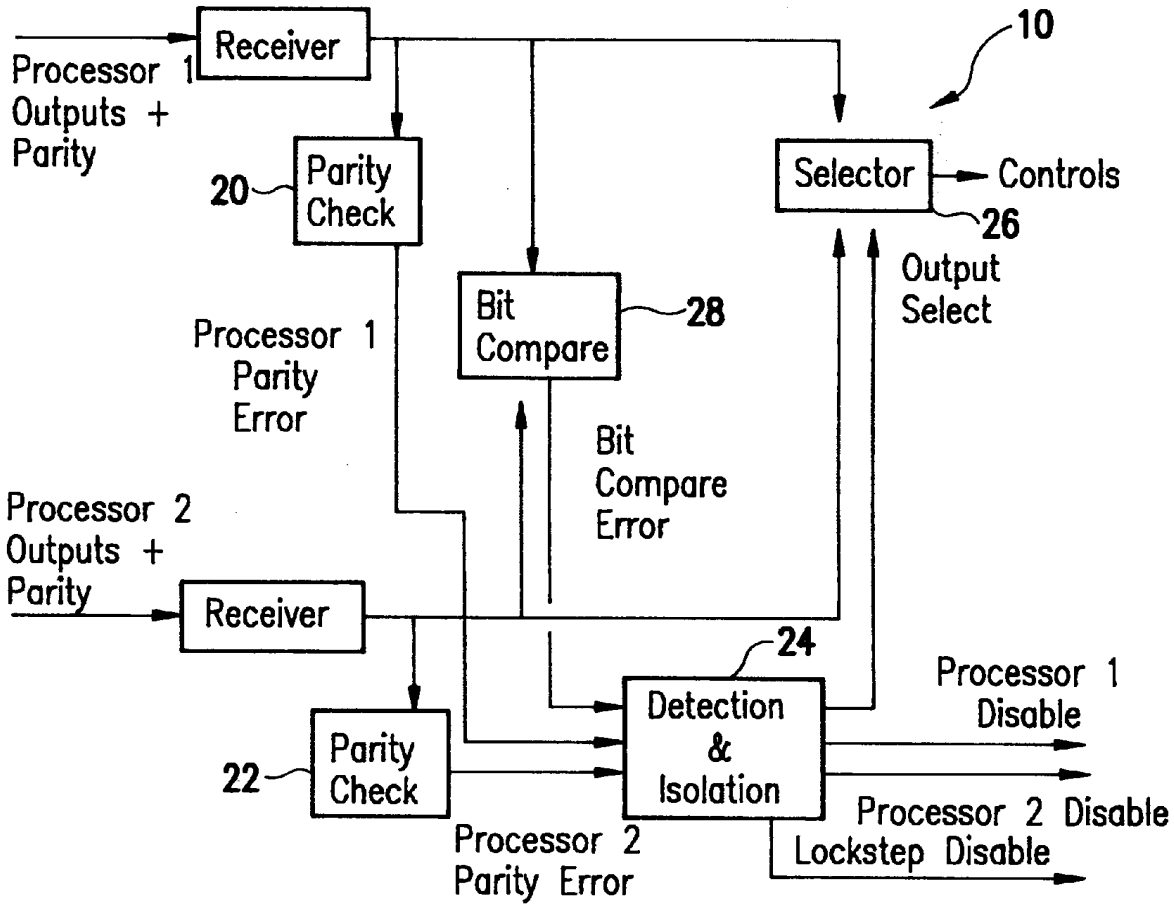

Receiver
Processor 1 Outputs + Parity
10
Parity Check
20
Selector
Controls
26
Output Select
Bit Compare
28
Processor 1 Parity Error
Bit Compare Error
Processor 2 Outputs + Parity
Receiver
24
Detection & Isolation
Processor 1 Disable
Parity Check
22
Processor 2 Disable
Lockstep Disable
Processor 2 Parity Error

FIG.3

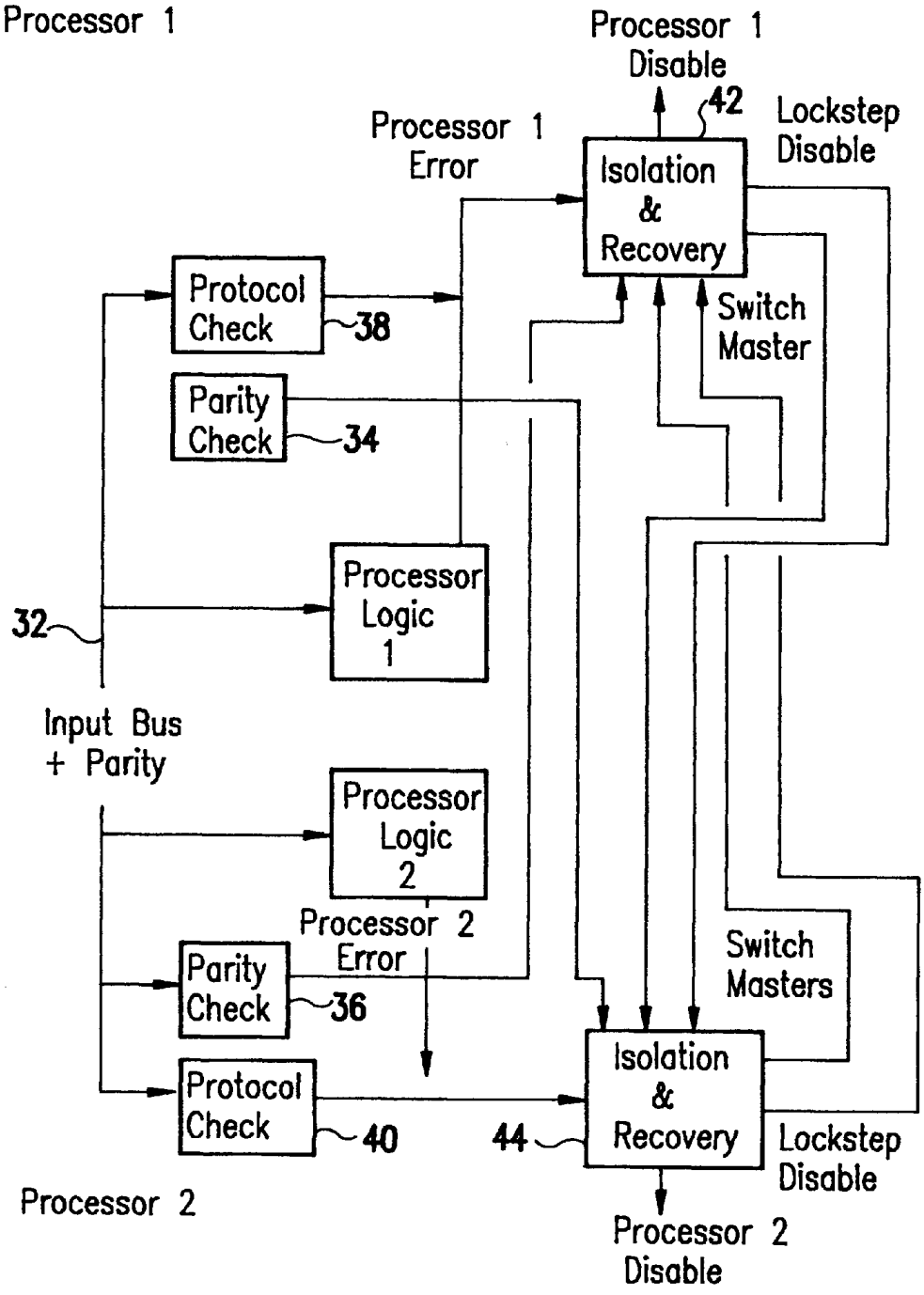

Processor 1
Processor 1
Disable
42
Lockstep
Disable
Processor 1
Error
Isolation
&
Recovery
Protocol
Check
38
Switch
Master
Parity
Check
34
Processor
Logic
1
32
Input Bus
+ Parity
Processor
Logic
2
Processor 2
Error
Switch
Masters
Parity
Check
36
Isolation
&
Recovery
Protocol
Check
40
44
Lockstep
Disable
Processor 2
Processor 2
Disable

FIG.4

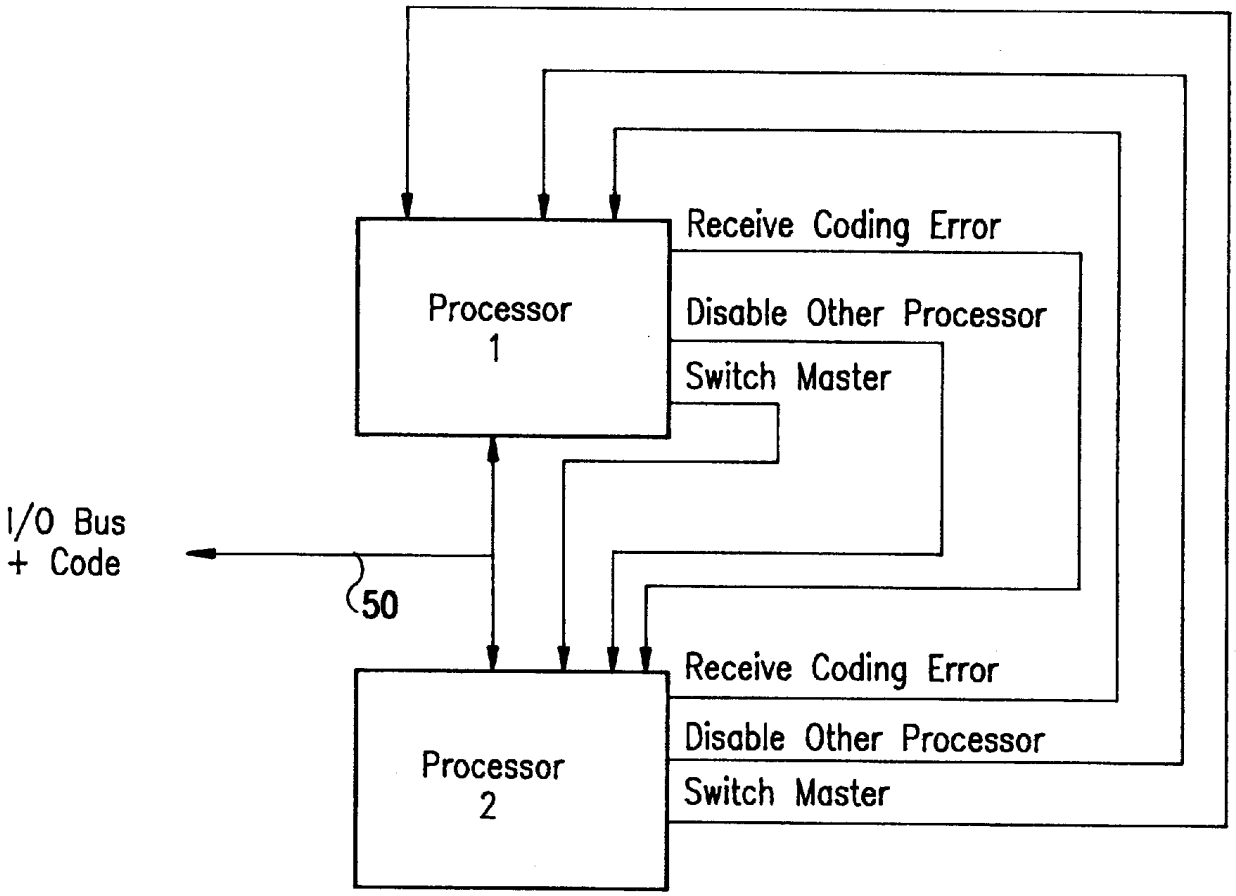

Processor
1
Receive Coding Error
Disable Other Processor
Switch Master
I/O Bus
+ Code
50
Processor
2
Receive Coding Error
Disable Other Processor
Switch Master

FIG.5

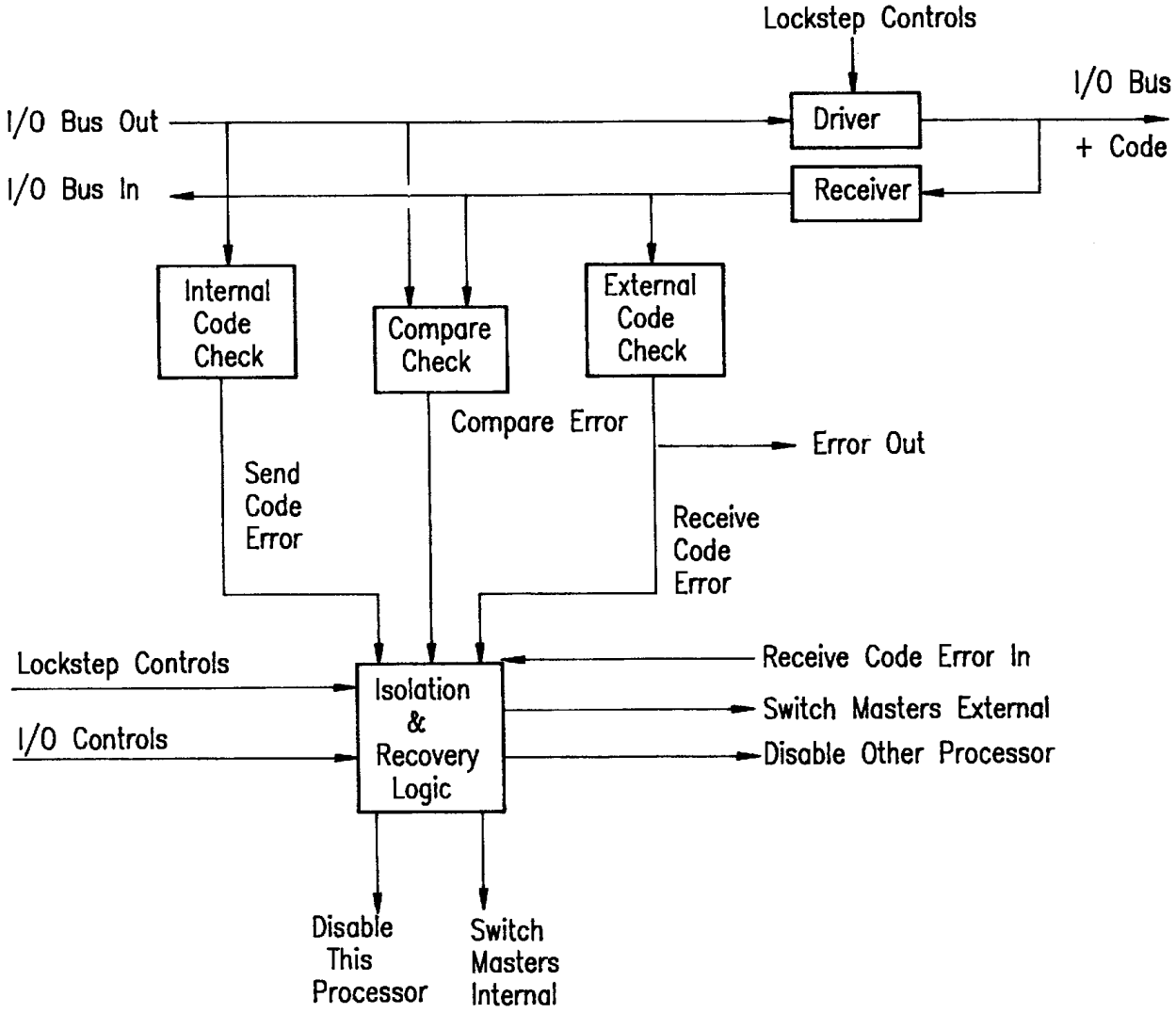

Lockstep Controls
I/O Bus
I/O Bus Out
Driver
+ Code
I/O Bus In
Receiver
Internal Code Check
Compare Check
External Code Check
Compare Error
Error Out
Send Code Error
Receive Code Error
Lockstep Controls
Receive Code Error In
Isolation & Recovery Logic
Switch Masters External
I/O Controls
Disable Other Processor
Disable This Processor
Switch Masters Internal

FIG.6

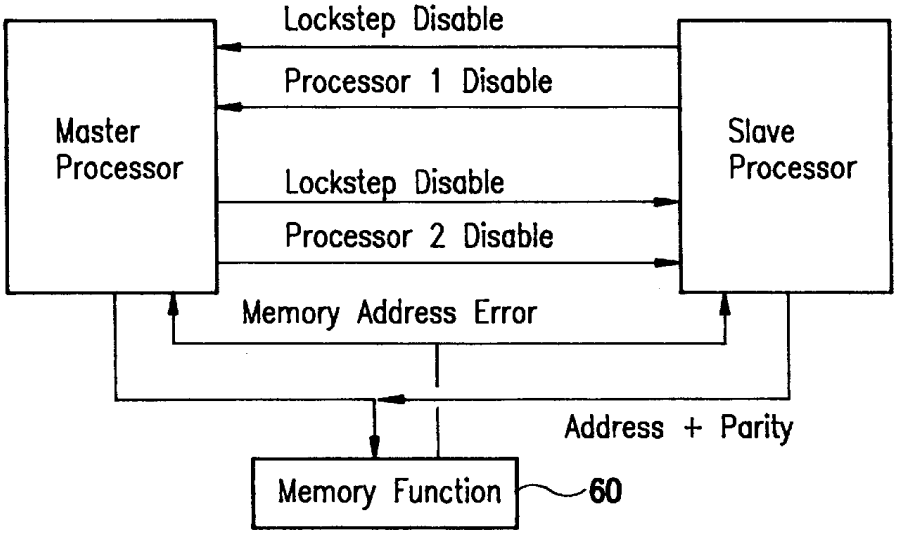

Lockstep Disable
Processor 1 Disable
Master Processor
Slave Processor
Lockstep Disable
Processor 2 Disable
Memory Address Error
Address + Parity
Memory Function
60

FIG.7

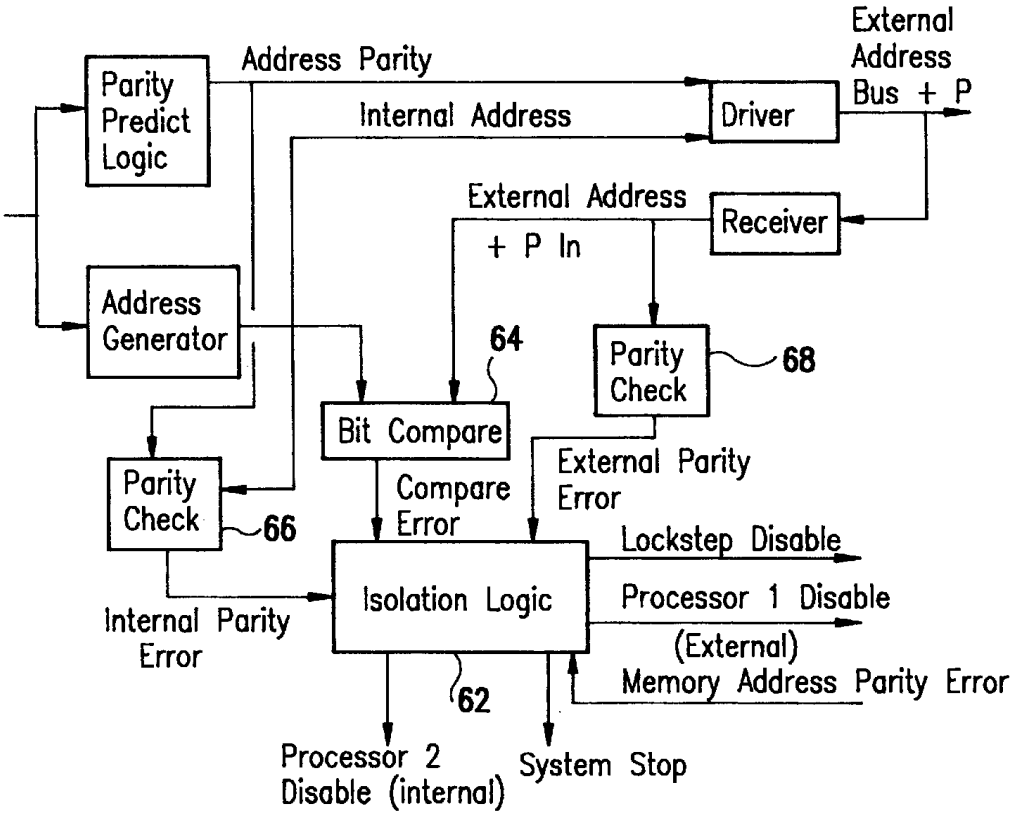

External Address Bus + P
Address Parity
Parity Predict Logic
Internal Address
Driver
External Address + P In
Receiver
Address Generator
64
Parity Check
68
Bit Compare
External Parity Error
Parity Check
66
Compare Error
Lockstep Disable
Isolation Logic
Processor 1 Disable (External)
Internal Parity Error
Memory Address Parity Error
62
Processor 2 Disable (internal)
System Stop

FIG.8

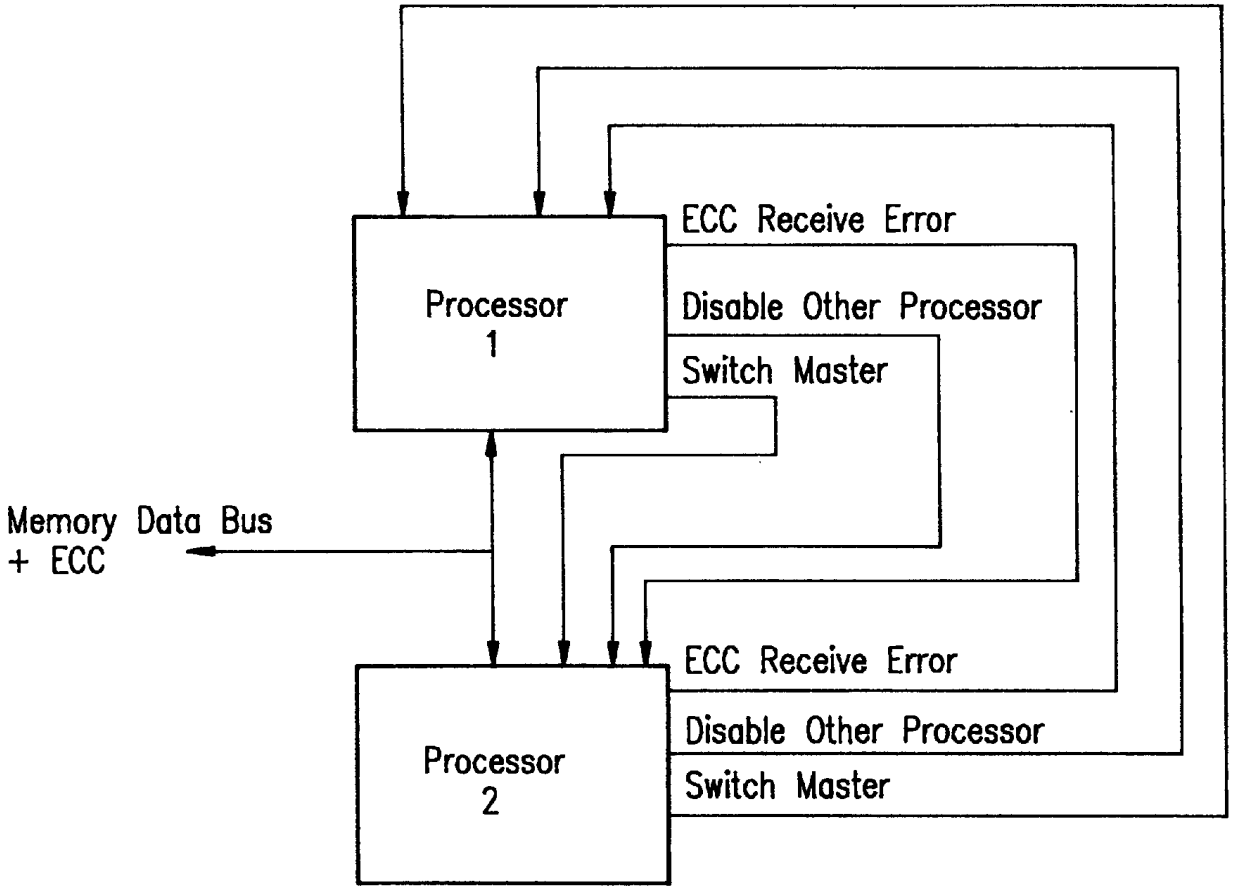

ECC Receive Error
Processor 1
Disable Other Processor
Switch Master
Memory Data Bus + ECC
ECC Receive Error
Processor 2
Disable Other Processor
Switch Master

FIG.9

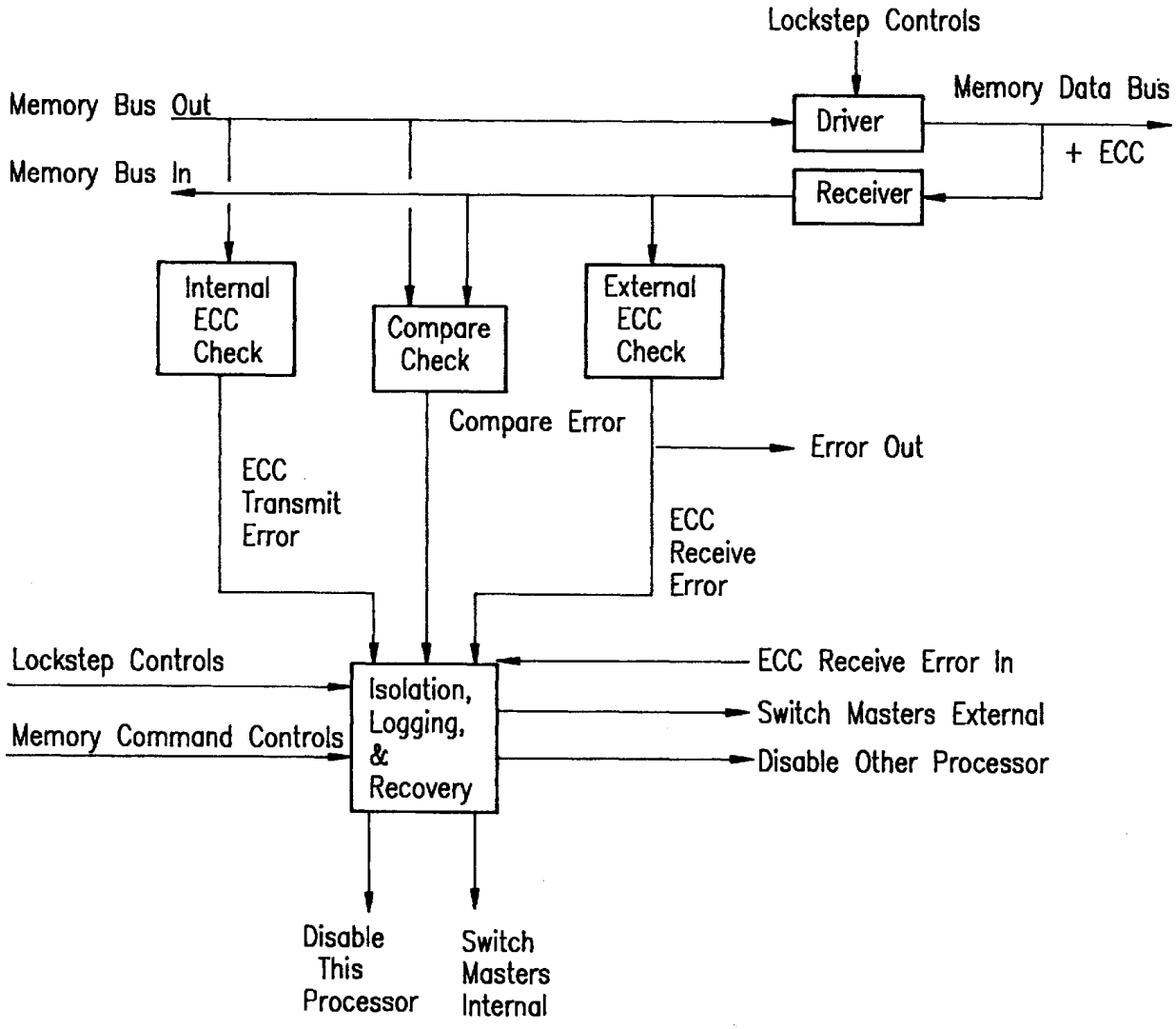

Lockstep Controls
Memory Bus Out
Memory Bus In
Driver
Receiver
Memory Data Bus
+ ECC
Internal ECC Check
Compare Check
External ECC Check
Compare Error
Error Out
ECC Transmit Error
ECC Receive Error
Lockstep Controls
Memory Command Controls
Isolation, Logging, & Recovery
ECC Receive Error In
Switch Masters External
Disable Other Processor
Disable This Processor
Switch Masters Internal

FIG.10

ERROR DETECTION AND FAULT ISOLATION FOR LOCKSTEP PROCESSOR SYSTEMS

This invention was made with Government support under Contract F29601-89-C-0089, awarded by the U.S. Air Force. The Government has certain rights in this invention.

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to so-called lockstep processor systems. That is, systems with at least two processors (commonly designated as master and slave) processing independently, but in lockstep, the same task with their independently generated results compared in order to detect an error originating in one processor. More particularly, the invention relates to an improved lockstep processor system that provides partial fault isolation in addition to error detection.

2. Description of the Prior Art

In general, prior art lockstep processor systems are designed to detect a failure, and do not typically include hardware and/or software to isolate the source of the error or the processor (master or slave) where the error occurred.

More specifically, in lockstep processors, typically the outputs are tied together. One of the processors is declared as the master and is allowed to drive the outputs, while the other processor is the slave and is only allowed to receive on its outputs. The slave processor compares the master's outputs with its internal outputs to ensure that they are the same. If on a line-by-line basis the outputs do not match, the whole system is stopped due to an error. The problem with this method is the lack of ability to isolate the failure to either the master or the slave processor, since typically code checking is not used on these lines and a consequential lack of ability to recover from a failure by operating in a degraded mode but without stopping the processor.

A classical lockstep technique, implemented with respect to the central output bus, is shown in FIG. **1**. Classical lockstep techniques are only concerned about control lines during cycles they are transmitting data. A bit-by-bit compare is performed between the data that is internally generated by one processor (slave processor) and data that has been placed on the external lines by the other processor (master processor). This comparison is valid only during cycles where data is being sent from the master processor. The comparison logic is activated in the slave processor and upon detecting any difference between the processor generated outputs on corresponding lines, the slave processor stops the system and waits for some higher level to initiate isolation and recovery actions.

In lockstep processors, the I/O bus is typically compared only during the cycles that the processor is transmitting data to an I/O device. When the processors are receiving data from an I/O device, the lockstep processors rely on internal data checking to find errors or rely on the processors eventually getting out of sync as a result of the errors. Once out of sync, the processors are stopped since there is no method in the prior art lockstep processor system to determine which processor is at fault. One object of this invention is to determine whether the master or slave processor has failed, to recover from the failure by decoding error detection, and, in addition, to provide lockstep coverage when receiving data from an I/O device.

In the prior art, detection of errors on input lines to lockstep processors, after their synchronization, has essentially been ignored. An assumption has been that both processors saw the same error and would respond in the same manner. If only one processor saw an error, the processors would no longer be synchronized. This lack of synchronization would eventually be detected at an output; but the only recovery action would be to reset the system. Another object of this invention is to detect error inputs to dual lockstep processors, isolate error inputs to a single processor before it corrupts the system, and disable the processor with error inputs.

Lines to a processor that are only inputs typically use either interrupt/error busses or control busses. In general such busses are protected from errors by at least one of the following four methods: parity checking, code checking, duplication, or protocol checking. A typical input bus of this type, with its corresponding checking, is shown in FIG. **1**.

In the classical dual lockstep processor configuration, one processor is the master, and it communicates with memory and I/O devices, while the other processor is the slave and it compares the outputs from the master to its internally generated signals to ensure that the master processor is synchronized with the slave. If the processors are not synchronized, the system is stopped.

Due to the large width of the memory address bus, a popular method of detecting any address failure is to perform lockstep comparison on the processor chips. The master processor and slave processor share the memory address lines. The master processor is allowed to drive the address and the slave processor compares what the master has driven to what the slave has generated internally on a bit-by-bit basis. If an error is detected, the system is stopped since comparison checks indicate only that there was a failure but not who failed. With this prior art method there is no isolation and the system cannot automatically change to a degraded mode of operation. A significant delay is incurred to attempt to isolate the error in prior art lockstep processor systems. In addition, the current method can only isolate the failure if the failure is detectable by the diagnostic routines; i.e. the error is not transient.

In prior art lockstep processor systems, the memory data bus is compared only during the cycles that the processor is writing memory. When the processor is reading memory, the lockstep processors rely on internal checking of the data to find errors or rely on the processors eventually getting out of sync. Once out of sync, the processors will be stopped since there is no method to determine who is at fault. An object of this invention is to determine whether the master or slave processor has failed, to recover from the failure by degrading error detection, and to provide error coverage during subsequent memory reads.

SUMMARY OF THE INVENTION

The previously stated specific objects of the invention can be summarized as the provision of a lockstep processor system which has error isolation built into the system and provides, in a large number of instances of detected error, high speed recovery with only small changes to current lockstep processor systems.

Briefly, this invention contemplates the provision of a lockstep processor system which adds error detection, isolation, and recovery logic to one or more lockstep processor system functions; namely, control outputs, processor inputs, I/O busses, memory address busses, and memory data busses.

For the control outputs, both lockstep processors and a receiver function are used to detect an error, isolate the

failure causing the error, and recover from the failure. Both processors and the receiver function use a common clocking source. Both processors generate parity over their respective control lines in sending control commands to the receiving function. The receiving function receives the control lines from each processor independently, checks the lines, performs a bit compare to detect and isolate any error, and transmits good control outputs for later use. As part of the process of isolating a source of an error as occurring in the master or slave processor, the receiving function also determines if the system can recover from the error and, if recoverable, the receiving function notifies the processors to disable the lockstep function and disable the failing processor.

The processor inputs are checked using checking logic and communication signals between the processors. Each processor notifies the other processor if it has detected an error, isolates the failure, and initiates recovery actions. To ensure that the processors are synchronized from a cycle perspective, a common clock source is used to drive both processors.

I/O busses are large and typically include some form of redundancy checking, such as parity. Both processors have the same I/Os, allowing communication between the processors concerning any detected errors. This allows both processors to be identical and allows either to be declared the master and the other processor declared the slave. When transmitting data to an I/O device, error detection and fault isolation from a lockstep error is the responsibility of the slave processor, while recovery is primarily performed by the slave processor with help from the master processor. When receiving data from an I/O device, detection, isolation, and recovery is jointly performed by both the master and slave processors.

Due to the large width of the memory address bus, a commonly used method to detect any address failure is to perform a lockstep comparison on the processor chips. Faults are isolated by adding parity generation to the memory address lines, and checking functions to memory function in addition to the lockstep processors. The memory data bus uses a suitable error correcting code for the memory busses in the lockstep processor system. Checking, fault isolation, and fault recovery occur on the processors themselves; for memory write operations, error detection and fault isolation is the responsibility of the slave processor and recovery is primarily performed by the slave processor. For memory read operations, detection, isolation, and recovery is jointly performed by the master and slave processor.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects, aspects and advantages will be better understood from the following detailed description of a preferred embodiment of the invention with reference to the drawings, in which:

FIG. **1** is a simplified block diagram of a classical implementation of a lockstep processor system.

FIG. **2** is a block diagram of a lockstep processor system with fault isolation in the control output busses.

FIG. **3** is a block diagram of the receiver function shown in FIG. **2**.

FIG. **4** is a block diagram of a lockstep processor system with fault isolation in the control input busses.

FIG. **5** is a block diagram of a lockstep processor system with fault isolation in the control I/O busses.

FIG. **6** is a block diagram of detection, isolation, and recovery logic for a processor shown in FIG. **5**.

FIG. **7** is a block diagram of a lockstep processor system with fault isolation in the control memory function address busses.

FIG. **8** is a block diagram of detection, isolation, and recovery logic for a processor shown in FIG. **7**.

FIG. **9** is a block diagram of a lockstep processor system with fault isolation in the control memory data busses.

FIG. **10** is a block diagram of detection, isolation, and recovery logic for a processor shown in FIG. **9**.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT OF THE INVENTION

Referring now to FIG. **2**, in accordance with the teachings of this invention, a receiver function **10** serves to detect and isolate faults in the control outputs of lockstep processors **1** and **2**. A common clock source (not shown) is coupled to the processors **1** and **2** and to the receiver function **10**. Each of the lockstep processors generates a parity bit for its control output, which parity bit is transmitted to the receiving function along with its control output. Referring now to FIG. **3**, the receiver function **10** receives each control output and its parity signal independently. Parity logic **20** detects any parity error in the control output of processor **1** and parity logic **22** detects any parity error in the control output of processor **2**. The outputs of parity logic **20** and **22** are coupled as inputs to error detection and isolation logic **24**, whose outputs include a control output select signal coupled to selector **26**, a processor **1** disable output, a processor **2** disable output, and a lockstep disable output. In addition to the parity check logic, the receiver function **10** includes bit compare logic **28** which performs a bit-by-bit comparison of control outputs of lockstep processors **1** and **2** The output of the bit compare logic **28** is coupled as another input to the error detection and isolation logic **24**.

For each control output line set, parity generation is added to the processor chip. In the simplest case, a parity generator is fed by the control signals. But a preferable method is to predict parity of the control lines from the same signals used to generate the control signals.

In operation, the isolation function determines what recovery action must be invoked. If an error is found in parity checking a particular processor's lines, and no bit compare error is found, either the processor's parity generation logic or its checking logic has failed; therefore, the lockstep feature and that processor are disabled and the other processor is made active, if it is not already the active processor. If an error is found in parity checking a processor's lines and a bit compare error is detected, either the processor's control generation logic or the receiving function receivers have failed; therefore, recovery is performed as in the previous case. If no parity checks are detected but a bit compare error is detected, the processors are out of sync and the system must be stopped. If parity checks are detected on both processor's lines, the system is stopped due to a major failure in either both processors or the receiving function.

Referring now to FIG. **4**, it shows the checking, isolation, notification, and recovery logic for a typical set of input busses to the lockstep processors, which in this exemplary embodiment, includes parity check bits. An input bus **32** (including parity) is driven to each processor **1** and **2** simultaneously to ensure that the processors are synchronized. Using the same net to drive the inputs, the processors **1** and **2** see the exact same signal in the same clock cycle. Since either processor can be master or slave, the same logic is used for both. In addition to parity checking logic (i.e.

parity check logic **34** for processor **1** and parity check logic **36** for processor **2**) this embodiment of the invention also includes protocol check logic **38** for processor **1** and protocol check logic **40** for processor **2**. The outputs of parity check logic **34** and protocol check logic **38** for processor **1** are coupled to isolation and recovery logic **42** for processor **1**. Similarly, the outputs of parity check logic **36** and protocol check logic **40** for processor **2** are coupled to isolation and recovery logic **44** for processor **2**. In addition, any error signal from processor **1** error detection logic is coupled to isolation and recovery logic **44** simultaneously as it is coupled to isolation and recovery logic **42**. Similarly, an error signal from processor **2** error detection logic is coupled to isolation and recovery logic **42** simultaneously as it is coupled to isolation and recovery logic **44**.

If either processor has detected an error on its inputs, both processors analyze the error to isolate the failure and to determine what corrective action is to be taken. The isolation method will be explained from the point of view of processor **1**. If processor **1** detects an error, and processor **2** does not detect an error, it is assumed that processor **1** has an internal problem. If processor **1** does not detect a failure, and processor **2** notifies processor **1** that it has detected an error, it is assumed that processor **2** has an internal problem. If both processors detect an error, it is assumed that either the driving function has failed or that there is a failure on the processor system board. Similarly, the same isolation decisions are made when we look at processor **2**.

Having isolated the failure, the system determines automatically what recovery action should be initiated. If the failure is determined to be in the driving function, or on the board, the master processor initiates a normal recovery technique for that error or stops the system since no recovery is possible. Looking at the recovery actions for a particular processor requires knowing which processor is master and which processor is slave. If the slave processor has determined that it has failed, the slave processor degrades the system fault detection by simply disabling itself and disabling the lockstep feature on both processors. If the master processor has determined that the slave has failed, the master processor degrades the system fault detection by disabling the lockstep feature on both processors and notifying the slave processor to disable itself. If the master processor detects that it is in error, the master processor degrades the system fault detection by disabling the lockstep feature on both processors, disabling itself, and then notifying the slave processor to become the active processor. If the slave processor detects that the master processor has failed, the slave processor degrades the system fault detection by disabling the lockstep feature on both processors, disabling the master processor, and then allowing itself to become the active processor. The corrective action to each potential error condition is shown in Table 1.

TABLE 1

Input Error Scenarios

Master	Slave	Action	Explanation
No Error	No Error	None	
No Error	Error	Degrade Fault Detection	Slave Processor Has Failed
Error	No Error	Degrade Fault Detection, Slave Made Active	Master Processor Has Failed
Error	Error	Stop the System	Error External to the Processors

With the exception of stopping the system, the recovery actions are invisible to the system. For errors that cause the system to be stopped, a checkstop signal is raised as soon as possible to ensure that no errors are propagated to the rest of the system. Any areas that the error may have propagated will require resetting to a known condition. If the slave processor is in error, the master processor ignores all lines from the slave processor starting with the cycle after the failure is detected. The slave processor is then stopped as quickly as possible. If the master processor has failed, the degradation of fault tolerance and making the slave processor active processor is performed on a cycle boundary and is completed before the corrupted data can affect the outputs of the master processor. Referring now to FIG. **5**, processors **1** and **2** each have the same inputs and the same outputs over I/O bus **50**. This allows communications between the processors concerning any errors detected; either processor can be designated master and the other designated slave. Since I/O busses are large and typically are covered by some form of redundancy checking such as parity, all checking, fault isolation, and fault recovery will occur on the processors themselves. When transmitting data to an I/O device, error detection and fault isolation from a lockstep error will be the responsibility of the slave processor, while recovery will be primarily performed by the slave processor with help from the master processor. When receiving data from an I/O device, detection, isolation, and recovery will be jointly performed by both the master and the slave processor.

To simplify discussion of the detection, isolation, and recovery actions, the internal design of a single processor is shown in FIG. **6**. As shown in FIG. **6**, the I/O transmit and receive paths are normally separate until the paths physically reach the transceiver; thus, simplifying the logic for send and receive fault detection and isolation. On the master processor, the transceiver direction is controlled directly by the processor I/O controller; thus acting as it would in a non-lockstep mode. On the slave processor, the transceiver direction will be controlled by both the processor I/O controller as well as the lockstep control logic. When receiving data from an I/O device, both the master and slave processors are in a receive mode to assure that the data is received during the same processor cycle. Since only the master is allowed to physically send data to an I/O device, the slave processor remains in receive mode during I/O writes to allow the slave processor to monitor the data being driven from the master processor.

Since the detection, isolation and recovery activities are independent for send and receive operations, the two will be described separately. When receiving, the data is read by both processors during the valid cycle. Each processor will perform a coding check to detect if any errors have occurred. After performing the check each processor will notify the other processor of the status of its receive function. If neither processor detects an error, processing will continue as normal. If both detect an error the system will stop since the error will be either an I/O device failure or a processor system board failure. If a processor detects an error when the other processor detected no error, the detecting processor will be assumed to have failed, will be stopped, and lockstep disabled. If the slave detects the error, the other processor continues as normal; if the master detects the error, the slave processor will be made the active processor and operation will continue in a degraded error detection mode. The error recovery scenarios for I/O receive operations are shown in Table 2.

TABLE 2

Error Scenarios for I/O Receive Operations

Master Code Check	Slave Code Check	Action	Explanation
No Error	No Error	None	
No Error	Error	Degrade Fault Tolerance, Disable Slave Processor	Failure in Slave Processor
Error	No Error	Degrade Fault Tolerance, Switch Masters	Failure in Master Processor
Error	Error	Stop System	Major I/O Device or Board Failure

When initiating the above actions, care was taken to ensure that the changes are invisible to the system. If the slave is determined to have failed, the master ignores all lines from the slave the cycle after the failure is detected. If the master is determined to have failed when receiving data, the degradation of the fault tolerance and the changing the slave processor to the active processor occurs on a cycle boundary and is completed before the corrupted data can affect any output lines. The stopping of the system due to a major failure occurs as quickly as possible to prevent the error from being propagated in the system.

During an I/O send operation, only the slave processor is involved with lockstep error detection, isolation and recovery. During the send, the slave processor's transceivers are placed in receive mode and the data is sampled only when it is valid on the bus. A coding check is performed on the data sampled from the bus to detect the presence of single errors. A coding check is also performed on the data generated by the slave processor. This check is performed during the same cycle as the cycle in which data from the master processor is checked. In addition, a bit-by-bit compare between the internally generated data and the external data is performed to aid in fault isolation. For example, if no coding check is detected on either bus but a miscompare occurs, the system is stopped because the processors are out of sync. If an error is detected on only one of the data busses and a miscompare is detected, the failed processor is stopped and the system continues to operate in a degraded error detection mode. If both coding checks detect errors, but a no miscompare is detected, the system is stopped since the same error has occurred in each lockstep processor. Table 3 below describes the error conditions and the recovery methods for each of the potential error combinations.

TABLE 3

Error Scenarios for I/O Send Operations

Master Coding Check	Slave Coding Check	Compare Check	Action	Explanation
No Error	No Error	No Error	None	
No Error	No Error	Error	Stop System	Processors Out of Sync
Error	Error	No Error	Stop System	Same Error Both Processors
No Error	Error	Error	Degrade Fault Tolerance, Disable Slave Processor	Slave Processor Failed
Error	No Error	Error	Degrade Fault Tolerance, Switch Masters	Master Processor Failed

TABLE 3-continued

Error Scenarios for I/O Send Operations

Master Coding Check	Slave Coding Check	Compare Check	Action	Explanation
Error	Error	Error	Stop System	Major Hardware Failure

As with I/O read operations, the recovery actions for I/O write operations must also be invisible to the system. On major errors that cause the system to be stopped, the checkstop signal is activated as soon as possible to prevent propagation of the failure throughout the system. Areas where the error may have reached will require a reset to a known condition. If a slave failure is detected, the master processor ignores all lines from the slave effectively the cycle after the failure was detected. On an I/O write operation, recovery of a failure of the master processor is the most time critical action. Since this failure is also detected by the I/O device, completion of the disabling of the master and changing the slave processor to the active processor is completed before the I/O device begins its normal recovery activities.

To incorporate fault isolation to the memory address function in accordance with the teachings of this invention, parity generation is added to the memory address lines and check functions are added to the memory function and to the lockstep processors. FIG. **7** is a block diagram at the system level of this lockstep processor memory address function. The memory function **60** receives the address with parity and performs a parity check on those lines. If no error is detected, memory operations continue as normal. If an error is detected, the memory function then notifies the processor chips that it has detected an error, which causes the processors to initiate a retry operation.

Since either processor can be defined as master, the parity generation function and the error detection and isolation logic are found on both processors. These functions, with the exception of the parity predict logic, are disabled in the master processor and the transceivers of the slave processor are placed in receive mode. During a valid memory operation, the slave processor receives the address lines from the master processor, performs a parity check on the bus, and performs a bit compare on each data bit of the address bus. To aid in the isolation of failures, the slave processor also parity checks its internal address bus. The block diagram of the parity generation, error detection, and error isolation functions found on each processor is shown in FIG. **8**.

As will be apparent from an inspection of FIG. **8**, the fault isolation logic **62** (here shown for processor **1** but duplicated in processor **2**) receives bit compare error input from bit compare logic **64**, which compares bit by bit the memory address of processor **1** (here labeled internal address) and the memory address of processor **2** (here labeled external address). Isolation logic **62** also receives any parity error indication in the memory address of processor **1** (internal parity error from parity check logic **66**) and any parity error indication in the memory address of processor **2** (external parity error from parity check logic **68**). Upon detecting an error, the slave processor either retries the memory operation, degrades the fault tolerance of the system by turning off one of the processors, or stops the system. If the slave determines that the failure is in its own logic, the slave

disables lockstep feature and disables itself. If the slave determines the failure is the master processor, the slave disables the lockstep feature, disables the master processor, and makes the slave processor the active processor. If the slave processor determines the failure to be in the memory function, the processors initiate a retry of the memory operation and then stop the system if the error persists. The error scenarios are found in Table 4.

TABLE 4

Error Scenarios for Lockstep Address Bus with Partial Isolation

Slave Parity Check	Master Parity Check	Bit Compare Check	Memory Parity Check	Action	Explanation
Error	No Error	No Error	No Error	Degrade Isolation, Ignore Parity Predict	Failure in Parity Predict Logic
Error	No Error	Error	No Error	Degrade Fault Tolerance. Disable Slave Processor	Failure in Slave's Address Generator
No Error	Error	No Frror	No Error	Degrade Isolation, Ignore Master Parity Error	Failure in Master Processor Parity Checker
No Error	Error	Error	No Error	Degrade Fault Tolerance, Disable Slave Processor	Failure in Address Line Receivers
No Error	No Error	Error	No Error	Stop the System	Processors out of sync
No Error	No Error	No Error	Error	Retry Operation	Memory Function Failure
No Error	Error	Error	Error	Degrade Fault Tolerance, Make Slave Processor Master	Master Processor Failure

The recovery actions for these failures are invisible to the system. On major errors that cause the system to be stopped, the checkstop signal is activated as soon as possible to prevent propagation of the failure throughout the system. If a slave failure is detected, the master processor is instructed to ignore all lines from the slave the cycle after the failure was detected. Recovery from failure of the master processor is completed by the time memory retry operation is ready to send the address. In addition, the recovery occurs on a clock boundary to ensure that the degradation of detection and switching active processors is invisible to the system.

Since memory data busses are large and most modern processors support error correcting code (ECC) for the memory busses, all checking, fault isolation, and fault recovery will occur on the processors themselves for memory data bases. For memory write operations, error detection and fault isolation is the responsibility of the slave processor and recovery is primarily performed by the slave processor with help from the master processor. For memory read operations, detection, isolation, and recovery is jointly performed by both the master and the slave processor. The design from a system point of view is shown in FIG. **9**. Both processors have the same inputs and outputs to allow communication between the processors concerning any errors detected. This allows both processors to be identical and allows either processor to be declared the master and the other processor declared as the slave processor. In addition, the processors share a common clocking source to eliminate any errors caused by clock skew.

To simplify discussion of the detection, isolation and recovery actions, the internal design of a single processor is shown in FIG. **10**. The memory load and memory store data paths are normally separate until the paths physically reach the transceiver; thus, simplifying the logic for load and store fault detection and isolation. On the master processor, the transceiver direction is controlled directly by the processor memory controller; thus acting as it would in a non-lockstep mode. On the slave processor, the transceiver direction will be controlled by both the processor memory controller as well as the lockstep control logic. When reading from memory, both the master and slave processors are in receive mode to assure that the data is received during the same processor cycle. Since only the master is allowed to physically write memory, the slave processor remains in receive mode during memory writes to allow the slave processor to monitor the data being driven from the master processor.

Since the detection, isolation and recovery activities are independent for read and write operations, the two will be described separately. On a memory read, the data is read by both processors during the valid memory cycle. Each processor will perform an ECC check to determine if any single bit or multiple bit errors have occurred. After performing the check each processor will notify the other processor of the status of its read. If neither processor detects an error or both processors detect a single bit error, processing will continue as normal. Since the error correction logic internal to the processor will correct the single bit error in each processor, they will remain in sync. If both detect a multiple bit error, the system will stop since the error will be either a memory failure or a processor system board failure. If a processor detects either a multiple bit error when the other detected no error or a single bit error or it detects a single bit error when the other detects no error, the detecting processor is assumed to have failed, is stopped, and lockstep is disabled. If the disabled processor is the slave, the other processor continues as normal. If the disabled processor is the master, the slave processor will be made the active processor and operation will continue in a degraded error detection mode. The error recovery scenarios for a memory read operation are shown in Table 5.

TABLE 5

Error Scenarios for Memory Data Bus Read Operations

Master ECC Check	Slave ECC Check	Action	Explanation
No Error	No Error	None	
No Error	Error	Degrade Fault Tolerance, Disable Slave Processor	Failure in Slave Processor
Error	No Error	Degrade Fault Tolerance, Switch Masters	Failure in Master Processor
Single Bit Error	Single Bit Error	Log Error	ECC Will Handle Error
Single Bit Error	Multiple Bit Error	Degrade Fault Tolerance Disable Slave Processor	Failure in Slave Processor
Multiple Bit Error	Single Bit Error	Degrade Fault Tolerance, Switch Masters	Failure in Master Processor
Multiple Bit Error	Multiple Bit Error	Stop System	Major Memory or Board Failure

During a memory write operation, only the slave processor is involved with lockstep error detection, isolation and recovery. During the write, the slave processor's transceivers are placed in receive mode and the data is sampled only when it is valid on the memory bus. An ECC check is performed on the data sampled from the memory bus to detect the presence of single or multiple bit errors. An ECC check is also performed on the data that was generated by the slave processor. This check is performed during the same cycle the data from the master processor is checked. In addition, a bit-by-bit compare between the internally generated data and the external data is performed to aid in fault isolation. For example, if no ECC check is detected on either data bus but a miscompare occurs, the system is stopped because the processors are out of sync. If a single bit error is detected on only one of the data busses and a single bit miscompare is detected, the system logs the failure but continues to operate since ECC will correct that error when the data is read. If both ECC checks detect a single bit error but a multiple bit miscompare is detected, the system is stopped since different errors have occurred in the lockstep processors. Table 6 describes the error conditions and the recovery methods from each of the potential error combinations.

While the invention has been described in terms of a single preferred embodiment, those skilled in the art will recognize that the invention can be practiced with modification within the spirit and scope of the appended claims.

Having thus described our invention, what we claim as new and desire to secure by letters patent is as follows:

1. A lockstep processor system including a master processor and a slave processor executing identical tasks independently in lockstep with one another, comprising in combination:

receiver means for comparing data on an output control bus of said master processor with data on a corresponding output control bus of said slave processor and for generating a compare error signal when corresponding bits on the corresponding busses miscompare;

said receiver means isolating a source of said compare error as originating with said master processor or with said slave processor;

said receiver means including:

means for generating an error detection code in said data on said output control bus of said master processor and means for generating an error detection code in said data on said output bus of said slave processor;

TABLE 6

Error Scenarios for Memory Data Bus Write Operations

Master ECC Check	Slave ECC Check	Compare Check	Action	Explanation
No Error	No Error	No Error	None	
No Error	No Error	Error	Stop System	Processors Out of Sync
No Error	Single Bit Error	Single Bit Error	Log Error	ECC Will Handle
Single Bit Error	No Error	Single Bit Error	Log Error	ECC Will Correct When Read
Single Bit Error	Single Bit Error	No Error	Stop System	Major Error
Single Bit Error	Single Bit Error	Multiple Bit Error	Stop System	Major Error
No Error	Multiple Bit Error	Error	Degrade Fault Tolerance, Disable Slave Processor	Slave Processor Failed
Single Bit Error	Multiple Bit Error	Error	Degrade Fault Tolerance, Disable Slave Processor	Slave Processor Failed, ECC Will Correct Master Error
Multiple Bit Error	No Error	Error	Degrade Fault Tolerance, Switch Masters	Master Processor Failed
Multiple Bit Error	Single Bit Error	Error	Degrade Fault Tolerance, Switch Masters	Master Processor Failed, ECC Will Correct Slave Error
Multiple Bit Error	Multiple Bit Error	Error	Stop System	Major Hardware Failure
Multiple Bit Error	Multiple Bit Error	No Error	Stop System	Major Hardware Failure

means responsive to said error detection code in said data on said output control bus of said master processor for generating a master processor error signal and means responsive to said error detection code in said data on said output control bus of said slave processor for generating a slave processor error signal; and

means responsive to said compare error signal, said master processor error signal, and said slave processor error signal for generating a lockstep disable signal and a processor disable signal.

2. A lockstep processor system as in claim **1** wherein said error detection code includes parity bits.

* * * * *

US007624302B2

(12) **United States Patent**
Michaelis et al.

(10) **Patent No.: US 7,624,302 B2**
(45) **Date of Patent: *Nov. 24, 2009**

(54) **SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR**

(75) Inventors: **Scott L. Michaelis**, Plano, TX (US); **Anurupa Rajkumari**, Round Rock, TX (US); **William B. McHardy**, Fort Collins, CO (US)

(73) Assignee: **Hewlett-Packard Development Company, L.P.**, Houston, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 513 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **10/972,588**

(22) Filed: **Oct. 25, 2004**

(65) **Prior Publication Data**

US 2006/0090064 A1 Apr. 27, 2006

(51) **Int. Cl.**
G06F 11/00 (2006.01)

(52) **U.S. Cl.** **714/12**; 714/11

(58) **Field of Classification Search** 714/10–13; 712/31

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,358,823	A		11/1982	McDonald et al.
5,226,152	A		7/1993	Klug et al.
5,249,188	A		9/1993	McDonald
5,537,655	A		7/1996	Truong
5,675,794	A		10/1997	Meredith
5,751,932	A		5/1998	Horst et al.
5,758,058	A		5/1998	Milburn
5,764,660	A		6/1998	Mohat
5,896,523	A		4/1999	Bissett et al.
5,915,082	A	*	6/1999	Marshall et al. 714/11
5,991,900	A		11/1999	Garnett
6,065,135	A		5/2000	Marshall et al.
6,141,770	A		10/2000	Fuchs et al.
6,148,348	A		11/2000	Garnett et al.
6,220,581	B1		4/2001	Mueller
6,263,452	B1		7/2001	Jewett et al.
6,438,687	B2		8/2002	Klein
6,473,869	B2		10/2002	Bissett et al.
6,560,682	B1		5/2003	Miller et al.

(Continued)

OTHER PUBLICATIONS

U.S. Appl. No. 10/972,796, filed concurrently herewith, entitled "System and Method for Maintaining in a Multi-Processor System a Spare Processor That is in Lockstep for Use in Recovering From Loss of Lockstep for Another Processor".

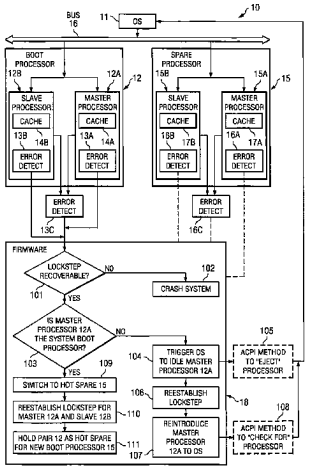
(Continued)

Primary Examiner—Dieu-Minh Le

(57) **ABSTRACT**

According to one embodiment, a method comprises detecting loss of lockstep (LOL) for a processor in a multi-processor system. The method further comprises determining that the processor for which the LOL is detected is assigned the role of boot processor, and switching the role of boot processor to a spare processor without shutting down the system's operating system. In another embodiment, a method comprises system firmware determining that an LOL is detected for a lockstep pair of processors that are assigned the role of boot processor in a system. The method further comprises determining one of the lockstep pair of processors that is not the cause of the LOL, and copying the state of the determined one of the lockstep pair of processors that is not the cause of the LOL to a spare processor. The method further comprises switching the role of boot processor to the spare processor.

27 Claims, 4 Drawing Sheets



U.S. PATENT DOCUMENTS

6,604,177	B1	8/2003	Kondo et al.	
6,615,366	B1	9/2003	Grochowski et al.	
6,625,749	B1 *	9/2003	Quach	714/10
6,675,324	B2	1/2004	Marisetty et al.	
6,687,851	B1	2/2004	Somers et al.	
6,754,787	B2	6/2004	Miller et al.	
6,920,581	B2 *	7/2005	Bigbee et al.	714/10
6,948,092	B2	9/2005	Kondo et al.	
7,003,691	B2	2/2006	Stafford et al.	
7,020,800	B2	3/2006	Fu et al.	
7,085,959	B2 *	8/2006	Safford	714/11
7,134,047	B2	11/2006	Quach	
7,155,721	B2	12/2006	Stafford et al.	
7,191,292	B2	3/2007	Chaudhry et al.	
7,225,355	B2	5/2007	Yamazaki et al.	
7,237,144	B2	6/2007	Stafford et al.	
7,251,746	B2 *	7/2007	Fox et al.	714/13
2002/0144177	A1	10/2002	Kondo et al.	
2002/0152420	A1	10/2002	Chaudhry et al.	
2003/0051190	A1	3/2003	Marisetty et al.	
2003/0070050	A1	4/2003	Miller et al.	
2003/0126498	A1	7/2003	Bigbee et al.	
2003/0135711	A1	7/2003	Shoemaker et al.	
2003/0140281	A1 *	7/2003	Fu et al.	714/38
2004/0006722	A1	1/2004	Safford	
2004/0019771	A1	1/2004	Quach	
2004/0078650	A1	4/2004	Stafford et al.	
2004/0078651	A1	4/2004	Stafford et al.	
2004/0153857	A1	8/2004	Yamazaki et al.	
2005/0172164	A1 *	8/2005	Fox et al.	714/13
2005/0240806	A1	10/2005	Bruckert et al.	
2005/0240811	A1	10/2005	Stafford et al.	
2005/0240829	A1 *	10/2005	Safford et al.	714/43
2006/0085677	A1	4/2006	Stafford et al.	
2006/0136672	A1	6/2006	Chaudhry et al.	
2006/0248384	A1	11/2006	Safford	
2006/0248684	A1	11/2006	Vosbikian et al.	

OTHER PUBLICATIONS

U.S. Appl. No. 10/973,077, filed concurrently herewith, entitled "System and Method for Establishing a Spare Processor for Recovering From Loss of Lockstep in a Boot Processor".

U.S. Appl. No. 10/972,835, filed concurrently herewith, entitled "System and Method for Using Information Relating to a Detected Loss of Lockstep for Determining a Responsive Action".

U.S. Appl. No. 10/973,076, filed concurrently herewith, entitled "System and Method for Providing Firmware Recoverable Lockstep Protection".

U.S. Appl. No. 10/973,004, filed concurrently herewith, entitled "System and Method for Configuring Lockstep Mode of a Prcessor Module".

U.S. Appl. No. 10/973,003, filed concurrently herewith, entitled "System and Method for Reestablishing Lockstep for a Processor Module for Which Loss of Lockstep is Detected".

U.S. Appl. No. 10/972,888, filed concurrently herewith, entitled "System and Method for System Firmware Causing an Operating System to Idle a Processor".

U.S. Appl. No. 10/973,075, filed concurrently herewith, entitled "System and Method for Reintroducing a Processor Module to an Operating System After Lockstep Recovery".

Notice of Allowance for U.S. Appl. No. 10/972,835 dated Nov. 20, 2008 in the United States.

Notice of Allowance for U.S. Appl. No. 10/972,835 dated Aug. 1, 2008 in the United States.

Office Action for U.S. Appl. No. 10/972,835 dated Mar. 3, 2008 in the United States.

Office Action for U.S. Appl. No. 10/972,835 dated Jul. 12, 2007 in the United States.

Office Action for U.S. Appl. No. 10/973,003 dated Mar. 5, 2008 in the United States.

Office Action for U.S. Appl. No. 10/973,003 dated Sep. 12, 2007 in the United States.

Notice of Allowance for U.S. Appl. No. 10/973,076 dated Nov. 28, 2008 in the United States.

Notice of Allowance for U.S. Appl. No. 10/973,076 dated Jul. 16, 2008 in the United States.

Notice of Allowance for U.S. Appl. No. 10/973,076 dated Jun. 3, 2008 in the United States.

Final Office Action for U.S. Appl. No. 10/973,076 dated Feb. 27, 2008 in the United States.

Office Action for U.S. Appl. No. 10/973,076 dated Sep. 12, 2007 in the United States.

Notice of Allowance for U.S. Appl. No. 10/972,888 dated Nov. 14, 2007 in the United States.

Office Action for U.S. Appl. No. 10/972,888 dated Jul. 18, 2007 in the United States.

Notice of Allowance for U.S. Appl. No. 10/973,004 dated Aug. 1, 2007 in the United States.

Office Action for U.S. Appl. No. 10/973,004 dated Apr. 10, 2007 in the United States.

Final Office Action for U.S. Appl. No. 10/973,075 dated May 12, 2009 in the United States.

Office Action for U.S. Appl. No. 10/973,075 dated Oct. 29, 2008 in the United State.

Office Action for U.S. Appl. No. 10/973,075 dated Mar. 12, 2008 in the United States.

Office Action for U.S. Appl. No. 10/973,075 dated Jul. 2, 2007 in the United States.

Notice of Allowance for U.S. Appl. No. 10/972,796 dated Dec. 19, 2007 in the United States.

Office Action for U.S. Appl. No. 10/972,796 dated Jul. 12, 2007 in the United States.

Shoichi Araki, Real-Time Tracking of Multiple Moving Object Contours in a Moving Camera Image Sequence, vol. E83-D lEICE Trans. Inf. & Syst, Jul. 7, 2000, 9 pages.

Jiang-Bin Zheng, Proceedings of the First International Conference on Machine Learning and Cybernetics, Beijing, Nov. 4-5, 2002, 4 pages.

Jong Ryul Kim, Automatic Localization and Trackign of Moving Objects Using Adaptive Snake Algorithm, ICICS-PCM 2003, Dec. 15-18, 2003, 5 pages, Singapore.

Jianhoa Meng, CVEPS-A Compressed Video Editing and Parsing System, Nov. 18, 1996, pp. 1-11.

Notice of Allowance dated Feb. 11, 2009 for U.S. Appl. No. 10/973,077 in the United States.

Notice of Allowance dated Sep. 23, 2008 for U.S. Appl. No. 10/973,077 in the United States.

Office Action dated Mar. 19, 2008 for U.S. Appl. No. 10/973,077 in the United States.

Office Action dated Sep. 20, 2007 for U.S. Appl. No. 10/973,077 in the United States.

Final Office Action for U.S. Appl. No. 10/972,835 dated Nov. 21, 2007 in the United States.

Office Action for U.S. Appl. No. 10/973,075 dated Dec. 13, 2007 in the United States.

Notice of Allowance, dated Jul. 23, 2009, U.S. Appl. No. 10/973,077, 20 pages.

* cited by examiner

FIG. 1

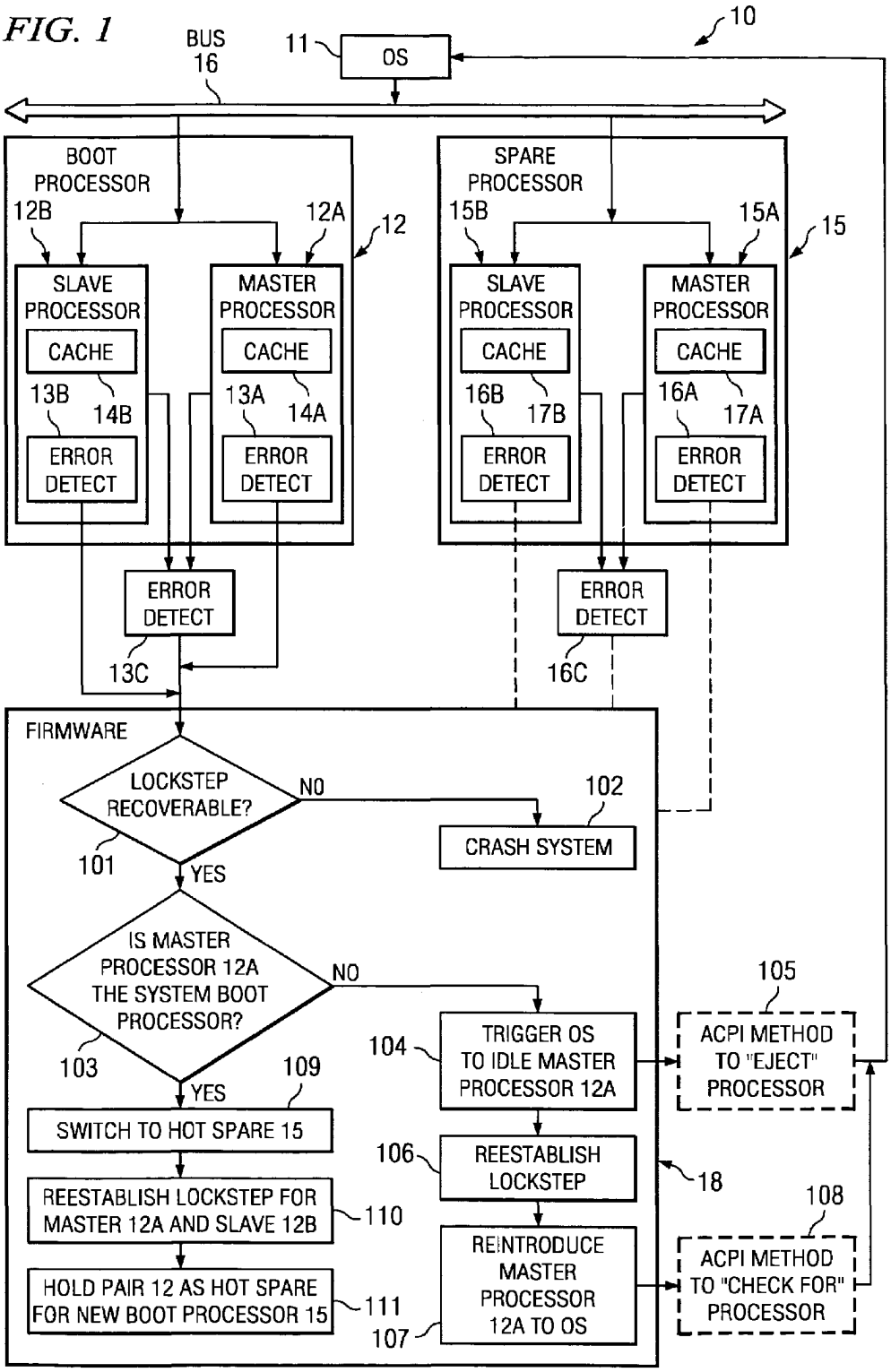

BUS
16
11
OS
10
BOOT
PROCESSOR
12B
12A
12
SPARE
PROCESSOR
15B
15A
15
SLAVE
PROCESSOR
MASTER
PROCESSOR
CACHE
13B
14B
13A
14A
16B
17B
16A
17A
ERROR
DETECT
13C
16C
FIRMWARE
LOCKSTEP
RECOVERABLE?
NO
102
CRASH SYSTEM
101
YES
IS MASTER
PROCESSOR 12A
THE SYSTEM BOOT
PROCESSOR?
103
NO
105
104
TRIGGER OS
TO IDLE MASTER
PROCESSOR 12A
ACPI METHOD
TO "EJECT"
PROCESSOR
109
YES
SWITCH TO HOT SPARE 15
106
REESTABLISH
LOCKSTEP
18
REESTABLISH LOCKSTEP FOR
MASTER 12A AND SLAVE 12B
110
108
REINTRODUCE
MASTER
PROCESSOR
12A TO OS
ACPI METHOD
TO "CHECK FOR"
PROCESSOR
HOLD PAIR 12 AS HOT SPARE
FOR NEW BOOT PROCESSOR 15
111
107

FIG. 2

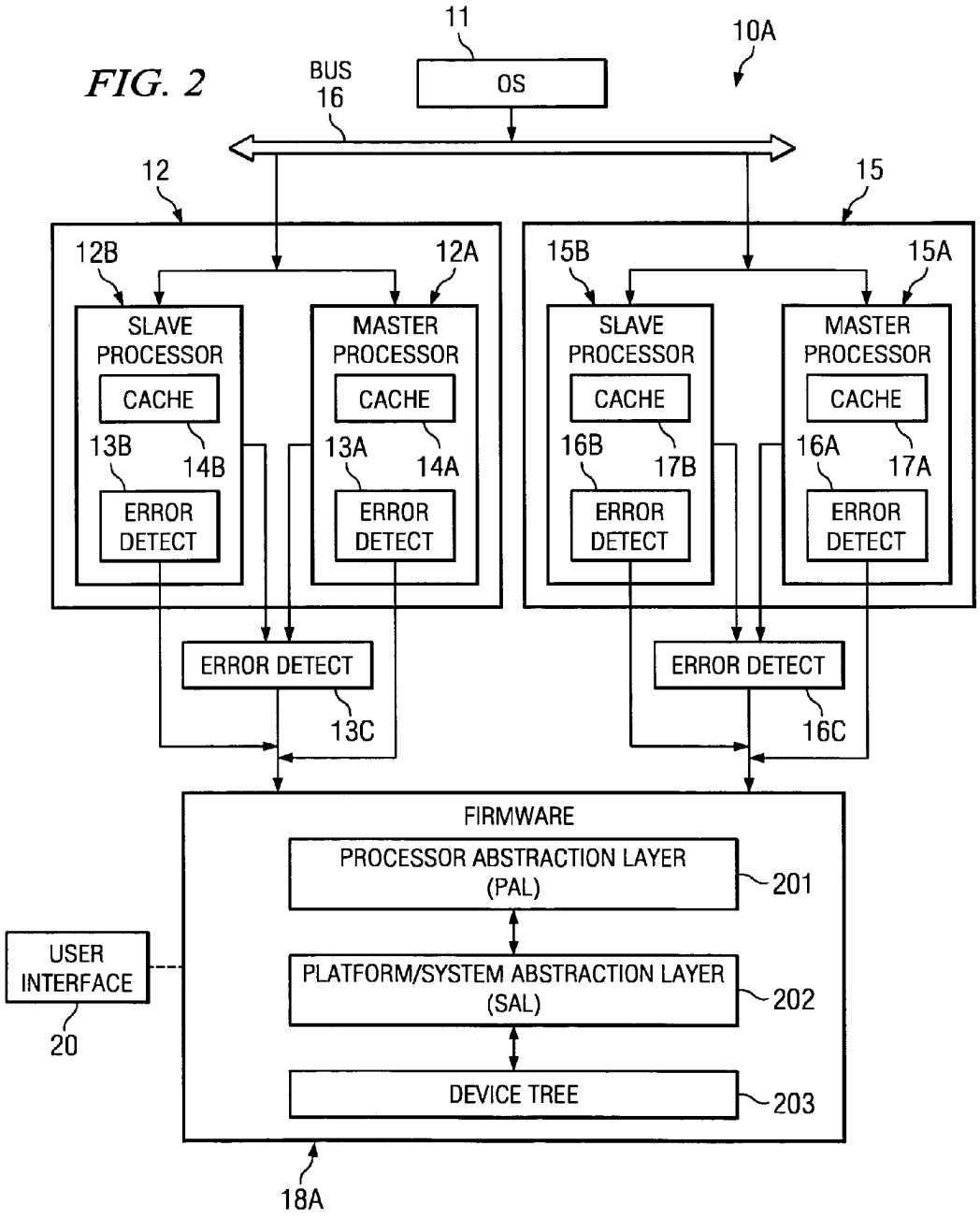

11
10A
BUS
16
OS
12
15
12B
12A
15B
15A
SLAVE PROCESSOR
MASTER PROCESSOR
CACHE
13B
14B
13A
14A
16B
17B
16A
17A
ERROR DETECT
ERROR DETECT
13C
16C
FIRMWARE
PROCESSOR ABSTRACTION LAYER (PAL)
201
PLATFORM/SYSTEM ABSTRACTION LAYER (SAL)
202
DEVICE TREE
203
USER INTERFACE
20
18A

FIG. 3

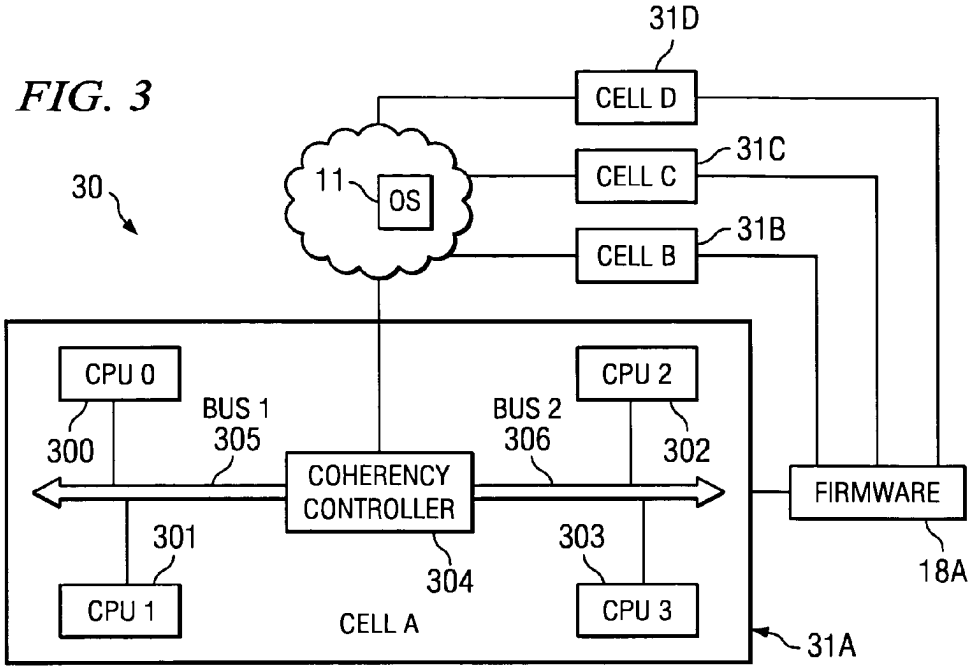

31D
CELL D
31C
CELL C
31B
CELL B
30
11
OS
CPU 0
300
BUS 1
305
BUS 2
306
CPU 2
302
COHERENCY
CONTROLLER
304
FIRMWARE
18A
301
CPU 1
303
CPU 3
CELL A
31A

FIG. 4

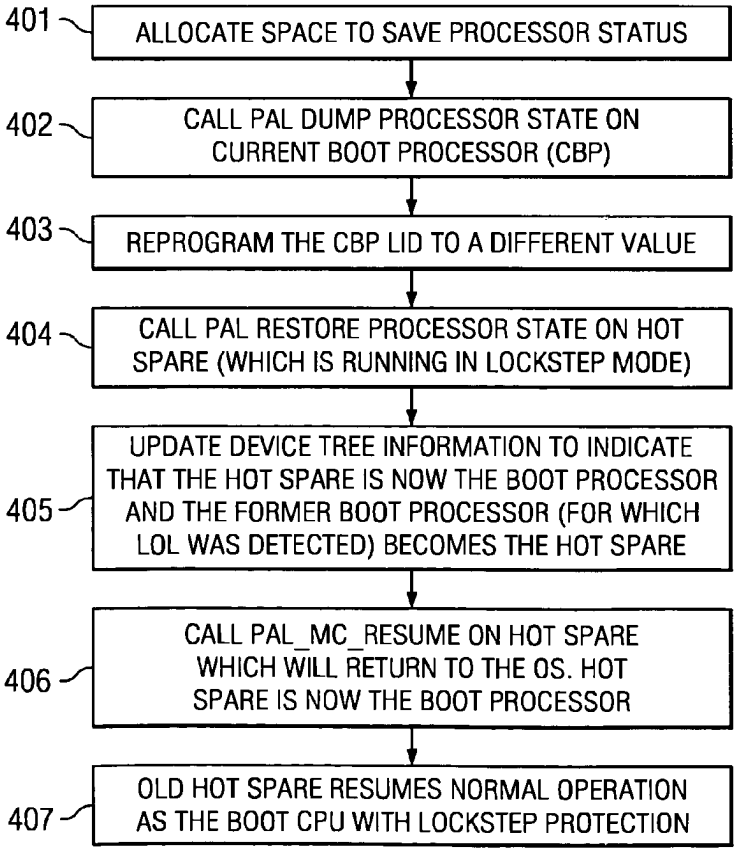

401
ALLOCATE SPACE TO SAVE PROCESSOR STATUS
402
CALL PAL DUMP PROCESSOR STATE ON CURRENT BOOT PROCESSOR (CBP)
403
REPROGRAM THE CBP LID TO A DIFFERENT VALUE
404
CALL PAL RESTORE PROCESSOR STATE ON HOT SPARE (WHICH IS RUNNING IN LOCKSTEP MODE)
405
UPDATE DEVICE TREE INFORMATION TO INDICATE THAT THE HOT SPARE IS NOW THE BOOT PROCESSOR AND THE FORMER BOOT PROCESSOR (FOR WHICH LOL WAS DETECTED) BECOMES THE HOT SPARE
406
CALL PAL_MC_RESUME ON HOT SPARE WHICH WILL RETURN TO THE OS. HOT SPARE IS NOW THE BOOT PROCESSOR
407
OLD HOT SPARE RESUMES NORMAL OPERATION AS THE BOOT CPU WITH LOCKSTEP PROTECTION

FIG. 5

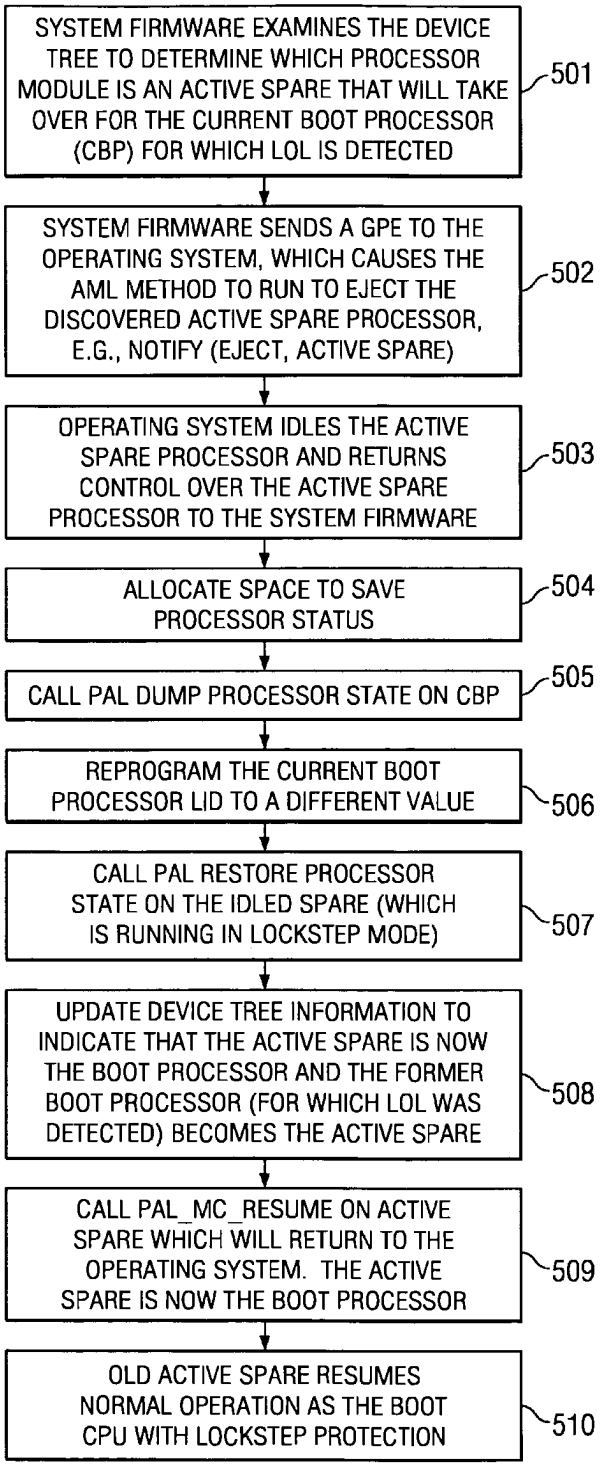
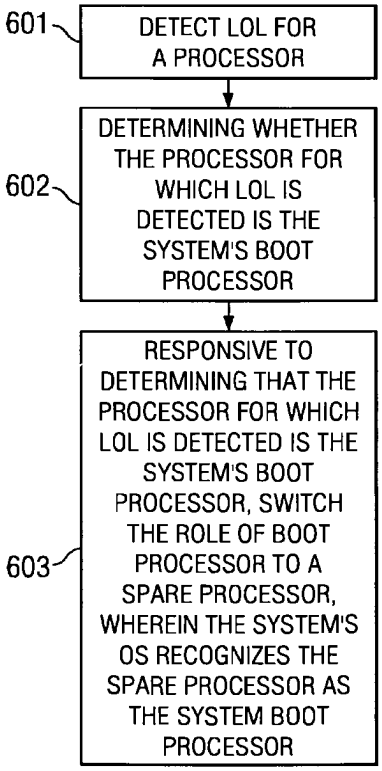

SYSTEM FIRMWARE EXAMINES THE DEVICE TREE TO DETERMINE WHICH PROCESSOR MODULE IS AN ACTIVE SPARE THAT WILL TAKE OVER FOR THE CURRENT BOOT PROCESSOR (CBP) FOR WHICH LOL IS DETECTED
501
SYSTEM FIRMWARE SENDS A GPE TO THE OPERATING SYSTEM, WHICH CAUSES THE AML METHOD TO RUN TO EJECT THE DISCOVERED ACTIVE SPARE PROCESSOR, E.G., NOTIFY (EJECT, ACTIVE SPARE)
502
OPERATING SYSTEM IDLES THE ACTIVE SPARE PROCESSOR AND RETURNS CONTROL OVER THE ACTIVE SPARE PROCESSOR TO THE SYSTEM FIRMWARE
503
ALLOCATE SPACE TO SAVE PROCESSOR STATUS
504
CALL PAL DUMP PROCESSOR STATE ON CBP
505
REPROGRAM THE CURRENT BOOT PROCESSOR LID TO A DIFFERENT VALUE
506
CALL PAL RESTORE PROCESSOR STATE ON THE IDLED SPARE (WHICH IS RUNNING IN LOCKSTEP MODE)
507
UPDATE DEVICE TREE INFORMATION TO INDICATE THAT THE ACTIVE SPARE IS NOW THE BOOT PROCESSOR AND THE FORMER BOOT PROCESSOR (FOR WHICH LOL WAS DETECTED) BECOMES THE ACTIVE SPARE
508
CALL PAL_MC_RESUME ON ACTIVE SPARE WHICH WILL RETURN TO THE OPERATING SYSTEM. THE ACTIVE SPARE IS NOW THE BOOT PROCESSOR
509
OLD ACTIVE SPARE RESUMES NORMAL OPERATION AS THE BOOT CPU WITH LOCKSTEP PROTECTION
510

FIG. 6


601
DETECT LOL FOR A PROCESSOR
602
DETERMINING WHETHER THE PROCESSOR FOR WHICH LOL IS DETECTED IS THE SYSTEM'S BOOT PROCESSOR
603
RESPONSIVE TO DETERMINING THAT THE PROCESSOR FOR WHICH LOL IS DETECTED IS THE SYSTEM'S BOOT PROCESSOR, SWITCH THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR, WHEREIN THE SYSTEM'S OS RECOGNIZES THE SPARE PROCESSOR AS THE SYSTEM BOOT PROCESSOR

SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

CROSS-REFERENCE TO RELATED APPLICATIONS

The present application is related to the following concurrently filed and commonly assigned U.S. Patent Applications: U.S. patent application Ser. No. 10/973,077 (U.S. Published Patent Application 20060107112) titled "SYSTEM AND METHOD FOR ESTABLISHING A SPARE PROCESSOR FOR RECOVERING FROM LOSS OF LOCKSTEP IN A BOOT PROCESSOR"; U.S. patent application Ser. No. 10/973,004 (U.S. Published Patent Application 20060107117, now U.S. Pat. No. 7,308,566) titled "SYSTEM AND METHOD FOR CONFIGURING LOCKSTEP MODE OF A PROCESSOR MODULE"; U.S. patent application Ser. No, 10/972,835 (U.S. Published Patent Application 20060107114) titled "SYSTEM AND METHOD FOR USING INFORMATION RELATING TO A DETECTED LOSS OF LOCKSTEP FOR DETERMINING A RESPONSIVE ACTION"; U.S. patent application Ser. No. 10/972,588 (U.S. Published Patent Application 20060090064) titled "SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR"; U.S. patent application Ser. No. 10/973,003 (U.S. Published Patent Application 20060107116) titled "SYSTEM AND METHOD FOR REESTABLISHING LOCKSTEP FOR A PROCESSOR MODULE FOR WHICH LOSS OF LOCKSThP IS DETECTED"; U.S. patent application Ser. No. 10/972,888 (U.S. Published Patent Application 20060107115) titled "SYSTEM AND METHOD FOR SYSTEM FIRMWARE CAUSING AN OPERATING SYSTEM TO IDLE A PROCESSOR"; U.S. patent application Ser. No. 10/973,075 (U.S. Published Patent Application 20060107111) titled "SYSTEM AND METHOD FOR REINTRODUCING A PROCESSOR MODULE TO AN OPERATING SYSTEM AFTER LOCKSTEP RECOVERY"; U.S. patent application Ser. No. 10/972,796 (U.S. Published Patent Application 20060107106) titled "SYSTEM AND METHOD FOR MAINTAINING IN A MULTI-PROCESSOR SYSTEM A SPARE PROCESSOR THAT IS IN LOCKSTEP FOR USE IN RECOVERING FROM LOSS OF LOCKSTEP FOR ANOTHER PROCESSOR", the disclosures of which arc hereby incorporated herein by reference.

DESCRIPTION OF RELATED ART

Silent Data Corruption ("SDC") is a difficult problem in the computing industry. In general, SDC refers to data that is corrupt, but which the system does not detect as being corrupt. SDCs primarily occur due to one of two factors: a) a broken hardware unit or b) a "cosmic" event that causes values to change somewhere in the system. Broken hardware means that a "trusted" piece of hardware is silently giving wrong answers. For example, the arithmetic unit in a processor is instructed to add 1+1 and it returns the incorrect answer 3 instead of the correct answer 2. An example of a cosmic event is when a charged particle (e.g., alpha particle or cosmic ray) strikes a region of a computing system and causes some bits to change value (e.g., from a 0 to a 1 or from a 1 to a 0).

Numerous techniques have been developed for detecting SDC to prevent the SDC from remaining "silent" or "undetected" within a system, as well as preventing such SDC from propagating through the system. Examples of these techniques include parity-based mechanisms and error correcting codes (ECCs) on buses and memory locations, as well as checksums and/or cyclic redundancy checks (CRC) over regions of memory. Parity-based mechanisms are often employed in processors, wherein a parity bit is associated with each block of data when it is stored. The parity bit is set to one or zero according to whether there is an odd or even number of ones in the data block. When the data block is read out of its storage location, the number of ones in the block is compared with the parity bit. A discrepancy between the values indicates that the data block has been corrupted. ECCs are parity-based mechanisms that track additional information for each data block. The additional information allows the corrupted bit(s) to be identified and corrected.

Parity/ECC mechanisms have been employed extensively for caches, memories, and similar data storage arrays. In the remaining circuitry on a processor, such as data paths, control logic, execution logic, and registers (the "execution core"), it is more difficult to apply parity/ECC mechanisms for SDC detection. Thus, there is typically some unprotected area on a processor in which data corruption may occur and the parity/ECC mechanisms do not prevent the corrupted data from actually making it out onto the system bus. One approach to SDC detection in an execution core (or other unprotected area of the processor chip) is to employ "lockstep processing." Generally, in lockstep processing two processors are paired together, and the two processors perform exactly the same operations and the results are compared (e.g., with an XOR gate). If there is ever a discrepancy between the results of the lockstep processors, an error is signaled. The odds of two processors experiencing the exact same error at the exact same moment (e.g., due to a cosmic event occurring in both processors at exactly the same time or due to a mechanical failure occurring in each processor at exactly the same time) is nearly zero.

A pair of lockstep processors may, from time to time, lose their lockstep. "Loss of lockstep" (or "LOL") is used broadly herein to refer to any error in the pair of lockstep processors. One example of LOL is detection of data corruption (e.g., data cache error) in one of the processors by a parity-based mechanism and/or ECC mechanism. Another example of LOL is detection of the output of the paired processors not matching, which is referred to herein as a "lockstep mismatch." It should be recognized that in some cases the data in the cache of a processor may become corrupt (e.g., due to a cosmic event), which once detected (e.g., by a parity-based mechanism or ECC mechanism of the processor) results in LOL. Of course, unless such corrupt data is acted upon by the processor, the output of that processor will not fail to match the output of its paired processor and thus a "lockstep mismatch" will not occur. For example, suppose that a value of "1" is stored to first location of cache in each of a pair of lockstep processors and a value of "1" is also stored to a second location of cache in each of the pair of lockstep processors. Further suppose that a cosmic event occurs for a first one of the processors, resulting in the first location of its cache being changed from "1" to "0", and thus corrupted. This data corruption in the first processor is a LOL for the pair. An error detection mechanism of this first processor may detect the data corruption, thus detecting the LOL. If the processors are instructed to act on the data of their first cache locations, then a lockstep mismatch will occur as the output of each of the processors will not match. For instance, if the processors each add the data

stored to the first location of their respective cache with the data stored to the second location of their respective cache, the first processor (having the corrupt data) will output a result of "1" (0+1=1) while the second processor outputs a result of "2" (1+1=2), and thus their respective outputs will not match.

By employing such techniques as parity-based error detection mechanisms and output comparisons for lockstep paired processors, SDC detection can be enhanced such that practically no SDC occurring in a processor goes undetected (and thus such SDC does not remain "silent") but instead results in detection of LOL. However, the issue then becomes how best for the system to respond to detected LOL. The traditional response to detected LOL has been to crash the system to ensure that the detected error is not propagated through the system. That is, LOL in one pair of lockstep processors in a system halts processing of the system even if other processors that have not encountered an error are present in the system. However, with the increased desire for many systems to maintain high availability, crashing the system each time LOL is detected is not an attractive proposition. This is particularly unattractive for large systems having many processors because cosmic events typically occur more frequently as the processor count goes up, which would result in much more frequent system crashes in those large systems. High availability is a major desire for many customers having large, multi-processor systems, and thus having their system crash every few weeks is not an attractive option. Of course, permitting corrupt data to propagate through the system is also not a viable option.

Prior solutions attempting to resolve at least some detected SDCs without requiring the system to be crashed have been Operating System ("OS") centric. That is, in certain solutions the OS has been implemented in a manner to recover from a detected LOL without necessarily crashing the system. This OS-centric type of solution requires a lot of processor and platform specific knowledge to be embedded in the OS, and thus requires that the OS provider maintain the OS up-to-date as changes occur in later versions of the processors and platforms in which the OS is to be used. This is such a large burden that most commonly used OSs do not support lockstep recovery.

Certain solutions have attempted to recover from a LOL without involving the OS in such recovery procedure. For instance, in one technique upon LOL being detected, firmware is used to save the state of one of the processors in a lockstep pair (the processor that is considered "good") to memory, and then both processors of the pair are reset and reinitialized. Thereafter, the state is copied from the memory to each of the processors in the lockstep pair. This technique makes the processors unavailable for an amount of time without the OS having any knowledge regarding this unavailability, and if the amount of time required for recovery is too long, the system may crash. That is, typically, if a processor is unresponsive for X amount of time, the OS will assume that the processor is hung and will crashdump the system so that the problem can be diagnosed. Further, in the event that a processor in the pair cannot be reset and reinitialized (e.g., the processor has a physical problem and fails to pass its self-test), this technique results in crashing the system.

BRIEF SUMMARY OF THE INVENTION

According to one embodiment, a method comprises detecting loss of lockstep (LOL) for a processor in a multi-processor system. The method further comprises determining that the processor for which the LOL is detected is assigned the role of boot processor, and switching the role of boot processor to a spare processor without shutting down the system's operating system.

According to one embodiment, a method comprises system firmware determining that loss of lockstep (LOL) is detected for a lockstep pair of processors that are assigned the role of boot processor in a system. The method further comprises determining one of the lockstep pair of processors that is not the cause of the LOL, and copying the state of the determined one of the lockstep pair of processors that is not the cause of the LOL to a spare processor. The method further comprises switching the role of boot processor to the spare processor.

According to one embodiment, a system comprises a plurality of processor modules that each include a master processor and a slave processor that operate in lockstep. The system further comprises an operating system, and error detection logic operable to detect loss of lockstep (LOL) for at least one of the processor modules. The system further comprises system firmware operable, responsive to detection of LOL for a first of the processor modules, to determine whether the first processor module is assigned a role of system boot processor. If determined that the first processor module is assigned the role of system boot processor, the system firmware is further operable to cause the operating system to recognize another processor module as system boot processor without shutting down the operating system.

According to one embodiment, a system comprises means for detecting loss of lockstep (LOL) for a processor in a multi-processor system. The system further comprises means for determining whether the processor for which the LOL is detected is assigned the role of boot processor, and means for switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which the LOL is detected is assigned the role of boot processor.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** shows an example multi-processor system in which firmware utilizes a spare processor for recovering from LOL for the system's boot processor;

FIG. **2** shows a block diagram of one embodiment implemented for the IA-64 processor architecture;

FIG. **3** shows an example system having multi-processor cells in which an embodiment for switching the role of boot processor to a spare processor responsive to detected LOL in the boot processor may be employed;

FIG. **4** shows an exemplary operational flow diagram for switching the role of boot processor to a hot spare processor according to one embodiment;

FIG. **5** shows an exemplary operational flow diagram for switching the role of boot processor to an active spare processor according to one embodiment; and

FIG. **6** shows a more general operational flow diagram for certain embodiments.

DETAILED DESCRIPTION OF THE INVENTION

As described further herein and in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/973,076 (U.S. Published Patent Appplication 20060107107) titled "SYSTEM AND METHOD FOR PROVIDING FIRMWARE RECOVERABLE LOCKSTEP PROTECTION," the disclosure of which is incorporated herein by reference, certain techniques are provided for recovering from LOL detected for a boot processor in a multi-processor system, in

which a spare processor is utilized for such recovery. For instance, upon LOL being detected for the system's boot processor, the spare processor assumes the role of boot processor, and then recovery of lockstep for the original boot processor is attempted. Accordingly, for such lockstep recovery technique, a spare processor is used for attempting to recover from LOL detected for the boot processor. Exemplary techniques are described in concurrently flied and commonly assigned U.S. patent application Ser. No. 10/973,077 (U.S. Published Patent Application 20060107112) titled "SYSTEM AND METHOD FOR ESTABLISHING A SPARE PROCESSOR FOR RECOVERING FROM LOSS OF LOCKSTEP IN A BOOT PROCESSOR," the disclosure of which is incorporated herein by reference, for establishing a spare processor for the boot processor. Such exemplary techniques include techniques for establishing a hot spare processor that is held in reserve until such time as it is used to assume the role of boot processor responsive to a detected LOL in the boot processor. The exemplary techniques further include techniques for establishing an active spare processor that is not held in reserve, but is instead made available to the OS for executing instructions during normal system operation and is dynamically transformed into a hot spare (i.e., it is idled) such that it can assume the role of the boot processor responsive to detection of LOL in the boot processor.

Embodiments are provided herein for switching the role of boot processor to an established spare processor responsive to detection of LOL in the boot processor. For instance, exemplary actions by system firmware for switching the role of boot processor to an established spare processor and thus causing the OS to recognize the established spare processor as the system's boot processor are provided in accordance with certain embodiments hereof.

As described further in concurrently filed and commonly assigned U.S. patent application Ser. No. U.S. patent application Ser. No. 10/973,077 (U.S. Published patent application 20060107112) titled "SYSTEM AND METHOD FOR ESTABLISHING A SPARE PROCESSOR FOR RECOVERING FROM LOSS OF LOCKSTEP IN A BOOT PROCESSOR," in certain embodiments, information (referred to herein as a "device tree") is stored to a multi-processor system, wherein such information identifies the processor in the system that is the boot processor and identifies the processor that is the spare for the boot processor. The information is stored such that it is accessible by the system's firmware, and the firmware, upon detecting LOL for the boot processor, determines the spare processor from the device tree and assigns the spare processor the role of boot processor. The firmware then attempts to recover lockstep for the processor for which LOL was detected. If the recovery of lockstep is successful, the processor for which lockstep was recovered (i.e., the former boot processor) may be assigned the role of spare for the current boot processor. Thus, the multi-processor system may recover from LOL detected for the boot processor without requiring a crash of the system (at least for certain types of "recoverable" LOLs).

In some implementations described herein, the spare processor for the boot processor is a "hot" spare, in which it is idled and not made available to the OS for processing instructions during normal system operation. That is, the hot spare processor is held in reserve, idling until such time as it assumes the role of boot processor responsive to detected LOL for the boot processor. In other implementations, the spare processor is not held in reserve, but is instead an "active" processor that is made available to the OS for executing instructions during normal system operation. Upon LOL being detected for the boot processor, the designated spare processor is dynamically transformed into a hot spare (i.e., it is idled) such that it can assume the role of the boot processor. For instance, upon the system's firmware detecting a LOL for the boot processor, the firmware accesses the device tree and determines the designated spare processor, and then the firmware instructs the OS to idle the designated spare processor (i.e., stop scheduling instructions for the processor) to make the "active" spare a "hot" spare, which then assumes the role of boot processor.

Concurrently filed and commonly assigned U.S. patent application Ser. No. 10/973,076 (U.S. Published Patent Application 20060107107) titled "SYSTEM AND METHOD FOR PROVIDING FIRMWARE RECOVERABLE LOCKSTEP PROTECTION," discloses a technique in which lockstep can be recovered for processors (particularly, non-boot processors) without requiring use of a spare processor. Rather, upon LOL being detected for a lockstep pair of processors, the firmware instructs the OS to idle (or "eject") the lockstep pair of processors for which LOL was detected and then the firmware attempts to reestablish lockstep (e.g., by resetting the processors). If successful in reestablishing lockstep, the firmware triggers the OS to recognize the lockstep pair of processors as again being available for servicing instructions.

However, for various reasons, in certain system architectures problems arise in attempting to idle (or eject) the boot processor from the system. Thus, in certain embodiments, a spare processor is used for recovering from LOL for the system's boot processor. Again, embodiments are provided herein for switching the role of boot processor to an established spare processor responsive to detection of LOL for the boot processor. In certain embodiments, the spare processor is used for recovering from LOL only in the case in which LOL is detected for the boot processor. For instances in which LOL is detected for non-boot processors, the lockstep may be recovered without the use of a spare by using the recovery technique described in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/973,076 (U.S. Published Patent Application 20060107107 titled "SYSTEM AND METHOD FOR PROVIDING FIRMWARE RECOVERABLE LOCKSTEP PROTECTION," for example.

Turning to FIG. **1**, an example embodiment of a system **10** in which a spare processor is used in recovering from detected loss of lockstep (LOL) for a system boot processor is shown. System **10** includes OS **11**, as well as master processor **12**A and slave processor **12**B (collectively referred to as a lockstep processor pair **12**). In certain implementations the lockstep processor pair **12** may be implemented on a single silicon chip, which is referred to as a "dual core processor" in which master processor **12**A is a first core and slave processor **12**B is a second core. Further, lockstep processor pair **12** may be referred to as a processor or CPU "module" because it includes a plurality of processors (**12**A and **12**B) in such module. As described further herein, to certain portions of system **10**, such as its OS **11**, the processor module **12** may appear as a single processor because slave processor **12**B is merely used for redundant processing for checking the output of master processor **12**A. In this example, lockstep processor pair **12** is designated as the boot processor for system **10**. As described further herein, in certain embodiments lockstep processor pair **12** is designated as the system boot processor by storing information to the system that specifies such designation.

Master processor **12**A includes cache **14**A, and slave processor **12**B includes cache **14**B. OS **11** and lockstep processor pair **12** are communicatively coupled to bus **16**. Typically, master processor **12**A and slave processor **12**B are coupled to

bus **16** via an interface that allows each of such processors to receive the same instructions to process, but such interface only communicates the output of master processor **12**A back onto bus **16**. The output of slave processor **12**B is used solely for checking the output of mater processor **12**A.

In this example, master processor **12**A includes error detect logic **13**A, and slave processor **12**B includes error detect logic **13**B. While shown as included in each of the processors **12**A and **12**B in this example, in certain embodiments the error detect logic **13**A and **13**B may be implemented external to processors **12**A and **12**B. Error detect logic **13**A and **13**B include logic for detecting errors, such as data cache errors, present in their respective processors **12**A and **12**B. Examples of error detect logic **13**A and **13**B include known parity-based mechanisms and ECC mechanisms. Error detect logic **13**C is also included, which may include an XOR (exclusive OR) gate, for detecting a lockstep mismatch between master processor **12**A and slave processor **12**B. As mentioned above, a lockstep mismatch refers to the output of master processor **12**A and slave processor **12**B failing to match. While shown as external to the lockstep processor pair **12** in this example, in certain embodiments error detect logic **13**C may be implemented on a common silicon chip with processors **12**A and **12**B.

Lockstep mismatch is one way of detecting a LOL between the master processor **12**A and slave processor **12**B. A detection of an error by either of error detect logic **13**A and **13**B also provides detection of LOL in the processors **12**A and **12**B. Because the detection of LOL by error detect logic **13**A and **13**B may occur before an actual lockstep mismatch occurs, the detection of LOL by error detect logic **13**A and **13**B may be referred to as a detection of a "precursor to lockstep mismatch". In other words, once an error (e.g., corrupt data) is detected by error detect logic **13**A or **13**B, such error may eventually propagate to a lockstep mismatch error that is detectable by error detect logic **13**C.

System **10** further includes a second lockstep processor pair **15** that is communicatively coupled to bus **16**. Lockstep processor pair **15** includes master processor **15**A and slave processor **15**B. Master processor **15**A includes cache **17**A, and slave processor **15**B includes cache **17**B. As with error detect logic **13**A-**13**C implemented for lockstep processor pair **12**, lockstep processor pair **15** has error detect logic **16**A-**16**C. While only two lockstep processor pairs **12** and **15** are shown for simplicity in the example of FIG. **1**, system **10** may include any number of such lockstep processor pairs. As one specific example, in one implementation system **10** includes **64** lockstep processor pairs, wherein the master processors of the pairs perform parallel processing for the system. In the specific example of FIG. **1**, lockstep processor pair **15** is designated as a spare for boot processor **12**, as described further herein.

In certain embodiments described herein, techniques are provided for switching the role of boot processor to the designated spare processor responsive to detection of LOL for the boot processor. Thus, the spare processor is used for recovery when LOL is detected for the system's boot processor. That is, the designated spare processor assumes the role of boot processor in the event that LOL is detected for the boot processor. In accordance with certain embodiments, upon LOL being detected for a boot processor, a hot spare processor (i.e., an idling processor that is available in the system) is transferred the role of boot processor, and then the old boot processor having LOL is reset to reestablish its lockstep. In other embodiments, a processor is designated as a spare processor for the boot processor, but such designated spare processor is not held in reserve (idled) during normal system operation. Rather, the designated spare is utilized as an active processing resource made available to OS **11** for processing instructions that OS **11** schedules for such "active" spare processor. Upon LOL being detected for the boot processor, the designated spare processor is idled (thereby, transforming such active processor into a hot spare), and is transferred the role of boot processor. Thereafter, the former boot processor for which the LOL was detected may be reset to reestablish its lockstep. Thus, in certain embodiments, a designated spare processor is not required to be held in reserve, but is instead an available resource for use within the system and is only transformed into an actual spare processor when needed for recovering from LOL for the boot processor.

Firmware **18** is also included in system **10**, which in this embodiment is invoked upon an error being detected by any of the error detect logics **13**A-**13**C and **16**A-**16**C. In certain embodiments, processors **12**A, **12**B, **15**A, and **15**B are processors from the Itanium Processor Family (IPF). IPF is a 64-bit processor architecture co-developed by Hewlett-Packard Company and Intel Corporation, which is based on Explicitly Parallel Instruction Computing (EPIC). IPF is a well-known family of processors. IPF includes processors such as those having the code names of MERCED, MCKINLEY, and MADISON. In addition to supporting a 64-bit processor bus and a set of 128 registers, the 64-bit design of IPF allows access to a very large memory (VLM) and exploits features in EPIC. While a specific example implementation of one embodiment is described below for the IPF architecture, embodiments described herein for establishing a spare processor for recovering from LOL detected for a boot processor are not limited in application to an IPF architecture, but may be applied as well to other architectures (e.g., 32-bit processor architectures, etc.).

Processor architecture generally comprises corresponding supporting firmware, such as firmware **18** of system **10**. For example, as described further below in conjunction with the specific example of FIG. **2**, the IPF processor architecture comprises such supporting firmware as Processor Abstraction Layer (PAL), System Abstraction Layer (SAL), and Extended Firmware Interface (EFI). Such supporting firmware may enable, for example, the OS to access a particular function implemented for the processor. For instance, the OS may query the PAL as to the size of the cache implemented for the processor, etc. Other well-known functions provided by the supporting firmware (SAL, EFI) include, for example: (a) performing I/O configuration accesses to discover and program the I/O Hardware (SAL_PCI_CONFIG_READ and SAL_PCI_CONFIG_WRITE); (b) retrieving error log data from the platform following a Machine Check Abort (MCA) event (SAL_GET_STATE_INFO); (c) accessing persistent store configuration data stored in non-volatile memory (EFI variable services: GetNextVariableName, GetVariable and SetVariable); and accessing the battery-backed real-time clock/calendar (EFI GetTime and SetTime). Accordingly, the supporting firmware, such as the PAL, is implemented to provide an interface to the processor(s) for accessing the functionality provided by such processor(s). Each of those interfaces provide standard, published procedure calls that are supported.

In the example system **10** of FIG. **1**, firmware **18** utilizes the above-mentioned hot spare technique for recovering from LOL for a boot processor. That is, processor **15** is designated as a hot spare for boot processor **12**, and such processor **15** is held as an idle, spare processor and used in the manner described further herein for assuming the role of boot processor in the event that a LOL is detected for boot processor **12**. Of course, as described further herein, the above-mentioned

technique in which a designated spare is not held in reserve but is instead dynamically transformed into a spare processor may be implemented by firmware **18** in alternative embodiments. As described further herein, in certain embodiments, a device tree is stored to a data storage device (e.g., SRAM) such that it is accessible to the firmware **18**, wherein the device tree includes information identifying processor **12** as the system's boot processor and identifying processor **15** as the spare. The information specifying the boot processor and/or spare may be received as input from a user in certain embodiments. For instance, in certain embodiments a user interface is provided that enables a system administrator to designate the boot processor and/or spare processor in the multi-processor system. In other embodiments, the firmware **18** selects the boot processor and/or spare.

Suppose that during the system boot-up procedure, master processor **12**A assumes the role of boot processor, as shown in the example of FIG. **1**. In this example embodiment, lockstep processor pair **15** is held as a hot spare for recovering from a LOL that may be detected for the boot processor **12**A. Of course, additional lockstep processor pairs may be included in system **10** (not specifically shown in the example of FIG. **1**), and those additional lockstep processor pairs, which are not the system boot processor, may recover from LOL in a different manner that does not utilize a hot spare processor. Thus, in accordance with certain embodiments hereof, hot spare processor pair **15** is not needed for recovering from LOL detected for any non-boot processor, but may instead be used for recovery of LOL only if such LOL is detected for the boot processor (processor **12**A in this example).

Firmware **18** is included in this example, and upon detection of LOL by any of error detect logics **13**A-**13**C, firmware **18** determines, in operational block **101**, whether the detected LOL is a recoverable LOL. That is, firmware **18** determines in block **101** whether the detected LOL is of a type from which the firmware can recover lockstep for the lockstep processor pair **12** without crashing the system. As described further herein, lockstep is recoverable for certain detected LOLs (which may be referred to as "recoverable LOLs"), while lockstep is not recoverable for other detected LOLs (which may be referred to as "non-recoverable LOLs"). If the lockstep is not recoverable from the detected LOL, then in the example of FIG. **1** firmware **18** crashes the system in block **102**.

If the lockstep is recoverable, then operation advances to block **103** whereat the firmware determines whether the processor for which the LOL was detected is the system's boot processor. Because the LOL is detected for the lockstep processor pair **12** in this example, firmware **18** determines whether master processor **12**A is the system's boot processor. This can be determined, for example, by accessing the device tree **203** of FIG. **2** (described further below). The device tree may, in certain implementations, include a field for each processor indicating whether such processor is the system's boot processor. This field in the device tree may be set by the firmware during the boot-up process to identify the corresponding processor that is used as the system's boot processor or it may be pre-set by a system administrator (e.g., via a user interface), as examples. If determined in block **103** that the master processor **12**A is not the system's boot processor, then operation advances in this example to block **104**, and a lockstep recovery process is performed that does not utilize hot spare **15**.

In the specific example shown in FIG. **1**, if the lockstep is determined to be recoverable in block **101** and the processor for which the LOL is detected is determined in block **103** to not be the system boot processor, firmware **18** cooperates with OS **11** via standard OS methods to recover the lockstep. For instance, in the example embodiment of FIG. **1**, Advanced Configuration and Power Interface (ACPI) methods are used by firmware **18** to cooperate with OS **11**. Accordingly, no processor or platform specific knowledge is required to be embedded in OS **11**, but instead any ACPI-compatible OS may be used, including without limitation HP-UX and Open VMS operating systems. Example techniques that may be used for recovering from LOL for non-boot processors of a system are described further in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/973,076 titled "SYSTEM AND METHOD FOR PROVIDING FIRMWARE RECOVERABLE LOCKSTEP PROTECTION," the disclosure of which is incorporated herein by reference.

In the example embodiment of FIG. **1**, if determined in block **103** that the processor for which the LOL is detected is not the system boot processor, firmware **18** triggers OS **11** to idle the master processor **12**A in operational block **104**. In this embodiment, firmware **18** utilizes an ACPI method **105** to "eject" master processor **12**A, thereby triggering OS **11** to idle the master processor **12**A (i.e., stop scheduling tasks for the processor). Of course, by idling master processor **12**A, slave processor **12**B will in turn be idled. Thus, idling master processor **12**A results in idling the lockstep processor pair **12**. In this example embodiment, OS **11** is not aware of the presence of slave processor **12**B, but is instead aware of master processor **12**A. The interface of lockstep processor pair **12** to bus **16** manages copying to slave processor **12**B the instructions that are directed by OS **11** to master processor **12**A. Thus, firmware **18** need not direct OS **11** to eject slave processor **12**B, as OS **11** is not aware of such slave processor **12**B in this example implementation. Again, by idling master processor **12**A, slave processor **12**B is also idled as it merely receives copies of the instructions directed to master processor **12**A. Of course, if in a given implementation OS **11** is aware of slave **12**B, firmware **18** may be implemented to also direct OS **11** to idle such slave processor **12**B in a manner similar to that described for idling master processor **12**A.

Firmware **18** then attempts to recover lockstep for the lockstep processor pair **12** in block **106**. For instance, firmware **18** resets the processor pair **12**. During such reset of processor pair **12**, system **10** can continue to operate on its remaining available processors. Once the processor pair **12** is reset and lockstep is recovered, firmware **18** reintroduces master processor **12**A to OS **11** in operational block **107**. In this embodiment, firmware **18** updates the ACPI device table information for master processor **12**A to indicate that such master processor **12**A is "present, functioning and enabled." As discussed in the ACPI 2.0 specification for the _STA status method of a device, the _STA (status) object returns the status of a device, which can be one of the following: enabled, disabled, or removed. In this respect, in the result code (bitmap) bit **0** is set if the device is present; bit **1** is set if the device is enabled and decoding its resources; bit **2** is set if the device should be shown in the UI; bit **3** is set if the device is functioning properly (cleared if the device failed its diagnostics); bit **4** is set if the battery is present; and bits **5**-**31** are reserved. A device can only decode its hardware resources if both bits **0** and **1** are set. If the device is not present (bit **0** cleared) or not enabled (bit **1** cleared), then the device must not decode its resources. Bits **0**, **1** and **3** are the "present, enabled and functioning" bits mentioned above. Firmware **18** utilizes an ACPI method **108** to trigger OS **11** to "check for" master processor **12**A, thereby reintroducing the master processor **12**A to OS **11**. As a result of checking for master processor **12**A, OS **11**

will recognize that such master processor **12**A is again available and will thus begin scheduling tasks for master processor **12**A once again.

In view of the above, a recovery technique is provided that does not require utilizing a spare processor for recovering from LOL detected for non-boot processors. For various reasons, in certain system architectures problems arise in attempting to idle (or eject) the boot processor from the system. Thus, in certain embodiments, a hot spare processor is used for recovering from LOL for the system's boot processor. For instance, in the example of FIG. **1**, if determined in block **103** that the master processor **12**A is the system's boot processor, operation advances to block **109** whereat the state of the "good" processor in the lockstep processor pair **12** is copied over to each of processors **15**A and **15**B in the spare lockstep processor pair **15**. Exemplary techniques that may be utilized by firmware **18** for switching the role of boot processor to the hot spare are described further herein below.

Thus, operational block **109** essentially makes the spare processor pair **15** the system's boot processor, and then in certain embodiments firmware **18** resets the lockstep processor pair **12** and reestablishes its lockstep in block **110**. Once lockstep is reestablished for lockstep processor pair **12**, that pair is held in operational block **111** as a hot spare for the new boot processor pair **15**. That is, firmware **18** updates the device tree **203** of FIG. **2** to reflect that lockstep processor pair **12** is a hot spare for the boot processor pair **15**. Thus, should a recoverable LOL be detected for the new boot processor **15**, the above process may be used by firmware **18** to make the spare lockstep processor pair **12** the boot processor and then recover lockstep for pair **15**. An example technique for resetting the lockstep processor pair **12** and reestablishing its lockstep in which the lockstep processor pair **12** is then held as a hot spare is described further in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/972,796 (U.S. Published Patent Application 20060107106) titled "SYSTEM AND METHOD FOR MAINTAINING IN A MULTI-PROCESSOR SYSTEM A SPARE PROCESSOR THAT IS IN LOCKSTEP FOR USE IN RECOVERING FROM LOSS OF LOCKSTEP FOR ANOTHER PROCESSOR", the disclosure of which is hereby incorporated herein by reference.

FIG. **2** shows a block diagram of one embodiment of the above system **10**, which is implemented for the IPF processor architecture and is labeled as system $\mathbf{10}_A$. The quintessential model of the traditional IPF architecture is given in the *Intel IA-64 Architecture Software Developer's Manual, Volume 2: IA-64 System Architecture*, in section 11.1 *Firmware Model*, the disclosure of which is hereby incorporated herein by reference. Accordingly, in this example embodiment of system $\mathbf{10}_A$, firmware **18**, labeled as firmware $\mathbf{18}_A$, includes processor abstraction layer (PAL) **201** and platform/system abstraction layer (SAL) **202**. In general, PAL **201** is firmware provided by Intel for its processors, and SAL **202** is developed by an original equipment manufacturer (OEM) for the specific system/platform in which the processors are to be employed. PAL **201**, SAL **202**, as well as an extended firmware interface (EFI) layer (not shown), together provide, among other things, the processor and system initialization for an OS boot in an IPF system.

The boot-up process of a traditional IPF system, for example, proceeds as follows: When the system is first powered on, there are some sanity checks (e.g., power on self-test) that are performed by microprocessors included in the system platform, which are not the main system processors that run applications. After those checks have passed, power and clocks are given to a boot processor (which may, for example, be master processor **12**A). The boot processor begins executing code out of the system's Read-Only Memory (ROM) (not specifically shown in FIG. **2**). The code that executes is the PAL **201**, which gets control of system **10**. PAL **201** executes to acquire all of the processors in system $\mathbf{10}_A$ (recall that there may be many lockstep processor pairs, such as pairs **12** and **15**) such that the processors begin executing concurrently through the same firmware.

After it has performed its duty of initializing the processor(s), PAL **201** passes control of system $\mathbf{10}_A$ to SAL **202**. It is the responsibility of SAL **202** to discover what hardware is present on the system platform, and initialize it to make it available for the OS **11**. When main memory is initialized and functional, the firmware $\mathbf{18}_A$ is copied into the main memory. Then, control is passed to EFI (not shown), which is responsible for activating boot devices, which typically includes the disk. The EFI reads the disk to load a program into memory, typically referred to as an operating system loader. The EFI loads the OS loader into memory, and then passes it control of system $\mathbf{10}_A$ by branching the boot processor into the entry point of such OS loader program.

The OS loader program then uses the standard firmware interfaces to discover and initialize system $\mathbf{10}_A$ further for control. One of the things that the OS loader typically has to do in a multi-processor system is to retrieve control of the other processors (those processors other than the boot processor). For instance, at this point in a multi-processor system, the other processors may be executing in do-nothing loops. In an ACPI-compatible system, OS **11** makes ACPI calls to parse the ACPI tables to discover the other processors of a multi-processor system in a manner as is well-known in the art. Then OS **11** uses the firmware interfaces to cause those discovered processors to branch into the operating system code. At that point, OS **11** controls all of the processors and the firmware $\mathbf{18}_A$ is no longer in control of system $\mathbf{10}_A$.

As OS **11** is initializing, it has to discover from the firmware $\mathbf{18}_A$ what hardware is present at boot time. And in the ACPI standards, it also discovers what hardware is present or added or removed at run-time. Further, the supporting firmware (PAL, SAL, and EFI) are also used during system runtime to support the processor. For example, OS **11** may access a particular function of master processor **12**A via the supporting firmware $\mathbf{18}_A$, such as querying PAL **201** for the number, size, etc., of the processor's cache **14**A. Some other well-known firmware functions that OS **11** may employ during runtime include: (a) PAL **201** may be invoked to configure or change processor features such as disabling transaction queuing (PAL_BUS_SET_FEATURES); (b) PAL **201** may be invoked to flush processor caches (PAL_CACHE_FLUSH); (c) SAL **202** may be invoked to retrieve error logs following a system error (SAL_GET_STATE_INFO, SAL_CLEAR_STATE_INFO); (d) SAL **202** may be invoked as part of hot-plug sequences in which new I/O cards are installed into the hardware (SAL_PCI_CONFIG_READ, SAL_PCI_CONFIG_WRIT); (e) EFI may be invoked to change the boot device path for the next time the system reboots (SetVariable); (f) EFI may be invoked to change the clock/calendar hardware settings; and (g) EFI may be invoked to shutdown the system (ResetSystem).

A "device tree" is provided, which is shown as device tree **203** in this example. Device tree **203** is stored in SRAM (Scratch RAM) on the cell, which is RAM that is reinitialized. Firmware **18**A builds the device tree **203** as it discovers what hardware is installed in the system. Firmware then converts this information to the ACPI tables format and presents it to OS **11** so that OS **11** can know what is installed in the system. The ACPI device tables (not shown) are only consumed by

OS **11** at boot time, so they are never updated as things change. For OS **11** to find the current status, it calls an ACPI "method" to discover the "current status". The _STA method described above is an example of such an ACPI method. When _STA is called, the AML can look for properties on the device specified in the firmware device tree and convert that into the Result Code bitmap described above. So, if lockstep has been lost on a processor, firmware **15**A will set the device tree property that indicates loss of lockstep, then when OS **11** calls _STA for that device, the "lockstep lost" property directs the AML code to return to "0" in the "functioning properly" bit so that OS **11** can know there is a problem with that processor.

If a lost lockstep is recovered in accordance with the recovery technique described herein, firmware **18**A can indicate that lockstep has been recovered in the device tree **203**. Then when _STA is called on that device responsive to the OS receiving the "check for device" ACPI method, the present, enabled and functioning bits will all be set and OS **11** will know the CPU is safe to use.

Further, device tree **203** includes information designating which processor is the boot processor, as well as a spare processor for the boot processor. A simple example of device tree **203** is shown below in Table 1:

TABLE 1

Device	Status	Lockstep Enabled	Boot Processor?	Spare for Boot Processor?
Processor A	Present, Enabled, and Functioning	Yes	Yes	No
Processor B	N/A	Yes	No	Yes
Processor C	Present, Enabled, and Functioning	Yes	No	No
...	...	...	...	...

In the above example of Table 1, 3 processors "Processor A," "Processor B," and "Processor C" are identified in the device tree as being included in the multi-processor system. The status of each of processors Processor A and Processor B is identified as "Present, Enabled, and Functioning." In this example Processor B is a hot spare, so it is held in reserve and does not show up in the static ACPI tables that are built when the system boots. Accordingly the OS does not know about Processor B and does not have a device handle to call the _STA method on. As described above, the _STA method is a known ACPI method for checking the status of the corresponding device. Therefore, the status of Processor B is designated N/A (not applicable) in this example because the _STA method will not be called for this processor as long as it is the hot spare. Of course, upon such spare assuming the role of boot processor, its status is updated to "Present, Enabled, and Functioning." In other embodiments in which Processor B is an active spare, the OS will be aware of such processor and accordingly its status may be also indicated as "Present, Enabled, and Functioning."

Further, lockstep is identified in this example as enabled for these processors. In certain embodiments, the system administrator can designate, via user interface **20**, whether lockstep mode is to be enabled for the system's processors. Further, Processor A is designated as the system's boot processor in the example device tree shown in Table 1, and Processor B is designated as the spare processor. Thus, upon detection of LOL for one of the processors in the system, the firmware can access this device tree to determine if the processor for which LOL is detected is the system's boot processor; and if determined that the processor for which LOL is detected is the system's boot processor, the firmware can further determine from this device tree the designated spare processor to use in the lockstep recovery process.

In the example of FIG. **2**, a user interface **20** is included, which allows a user (e.g., system administrator) to designate the system's boot processor and spare processor. In certain embodiments, the user interface **20** further allows the user to designate other information, such as whether lockstep mode is enabled for the processors.

One example system in which firmware $\mathbf{18}_A$ of FIG. **2** may be employed is shown in FIG. **3**. FIG. **3** shows system **30** that includes multi-processor cells A-D, labeled $\mathbf{31}_A$-$\mathbf{31}_D$, respectively. Cell A $\mathbf{31}_A$ is shown in more detail, and should be understood that cells B-D $\mathbf{31}_B$-$\mathbf{31}_D$ have substantially the same architecture as that of cell A $\mathbf{31}_A$ in this example. The architecture of cell A $\mathbf{31}_A$ described hereafter corresponds to that of Hewlett-Packard's Superdome™ systems. As shown, multi-processor cell A $\mathbf{31}_A$ includes coherency controller **304** that is communicatively coupled to two buses, labeled **305** and **306**, respectively. A plurality of processor modules are included within cell A $\mathbf{31}_A$, shown as CPU **0**, CPU **1**, CPU **2** and CPU **3** (and labeled **300**-**303**, respectively). More particularly, a plurality of processors modules are communicatively coupled to each of the buses **305** and **306**. In this example, CPU **0** (**300**) and CPU **1** (**301**) are each communicatively coupled to the first bus **305**, and CPU **2** (**302**) and CPU **3** (**303**) are each communicatively coupled to the second bus **306**. It should be understood that each of CPU **0** (**300**), CPU **1** (**301**), CPU **2** (**302**), and CPU **3** (**303**) is in actuality a lockstep processor pair, such as the lockstep processor pair **12** of FIGS. **1**-**3**. Thus, for instance, CPU **0** (**300**) includes both a master processor and a slave processor, as with processors **12**A and **12**B included in the lockstep processor pair **12** of FIGS. **1**-**2**.

Coherency controller **304** decodes the address an interrupt is targeted towards and determines which bus (**305** or **306**) the interrupt packet should be delivered to. Each CPU on the destination bus sees the interrupt and compares the interrupt target address with its internal "LID" register to determine if it is the targeted CPU. If the match is correct, the CPU responds to the interrupt. If the match is not correct, the CPU ignores the interrupt packet.

As shown, the example firmware $\mathbf{18}_A$ described above with FIGS. **1**-**2** may be employed to manage the recovery from detected LOLs for the processors of cells A-D ($\mathbf{31}_A$-$\mathbf{31}_D$). In this example architecture, when designating a spare for the boot processor, such spare is selected to be another processor communicatively coupled to the same bus as the boot processor. For instance, suppose that during the boot-up of system **30**, CPU **2** (**302**) is designated as the boot processor. Accordingly, firmware $\mathbf{18}^A$ will identify another processor that is communicatively coupled to bus **306** to designate as a spare for such boot processor, such as CPU **3** (**303**). During runtime if a LOL is detected for CPU **2** (**302**), then firmware $\mathbf{18}_A$ may operate according to the process described above in FIG. **1** to switch the designation of boot processor to a designated spare, CPU **3** (**303**). In performing this switch the LID register of the CPU that lost lockstep is copied into the LID register of the spare CPU and the spare CPU pair will then assume the ID of the failed CPU and start responding to interrupts directed to that ID. The LID of the failed CPU is quickly re-written to a different value so that it will no longer respond to interrupts directed to that ID. CPU **2** (**302**) can then be reset to recover its lockstep, and such CPU **2** (**302**) may then be designated as a spare for the new boot processor CPU **3** (**303**).

It should be recognized that in the example system of FIG. **3**, if a hot spare were used for recovering from LOL for all

processors, rather than just for the boot processor, much of the potential processing resources would be required to be held in reserve as spare resources. For instance, a hot spare would be required on each of the buses of each cell. Because there are two processors coupled to each bus in this example (i.e., CPU **0** and CPU **1** coupled to bus **305**, and CPU **2** and CPU **3** coupled to bus **306**), reserving a hot spare processor for each bus would result in holding half of the total processing resources in reserve. This is an undesirably expensive and wasteful solution. Thus, according to the illustrated embodiment, a hot spare processor is reserved only for the boot processor. Thus, continuing with the above example in which CPU **2** is the boot processor and CPU **3** is held as a "hot" spare, both of CPU **0** and CPU **1** are active (i.e., neither are held in spare). Further, none of the processors of cells B-D need to be held in reserve as hot spares. In the event of a recoverable LOL occurring for any of non-boot processors of the cells, the lockstep recovery process of FIG. **1** (i.e., operational blocks **104-108**) can be utilized without requiring a spare for lockstep recovery for those non-boot processors.

While in the above example embodiment, a hot spare CPU is maintained for recovery from LOL for the system's boot processor, in an alternative embodiment a hot spare need not be held as such, but rather a processor can be dynamically made a "spare" when needed for recovery from LOL encountered for the system boot processor. That is, assuming as in the above example of FIG. **3** that CPU **2** is the system's boot processor, CPU **3** does not have to be held in reserve (idled) as a hot spare for CPU **2**, but instead CPU **3** can be active and dynamically idled and made a "spare" upon detection of LOL in CPU **2**. Accordingly, CPU **3** may be designated as a spare for the boot processor, but such CPU **3** need not be held in reserve during normal system operation. Instead, the CPU **3** may be dynamically (during system run-time) transformed into a spare processor (e.g., by causing the OS to idle such processor) in the event that a LOL is detected for the boot processor.

Thus, the resource of this spare CPU need not be wasted during normal runtime, but can be dynamically turned into a spare processor that is available for recovering from LOL encountered on CPU **2** in the manner described above. As an example of this embodiment, the firmware may send an "eject request" on a healthy processor pair (e.g., CPU **3**), and lie to the OS by indicating (e.g., in response to an ACPI _STA method for the healthy processor pair) that such processor pair is not functioning. This would cause the OS to eject that processor pair as though it had encountered a LOL. This processor would become the "hot spare" and be used to replace the boot processor. The system boot processor role would be assumed by the idled healthy processor pair, just as described above in the example in which the healthy CPU **3** is held idle for the system boot processor, CPU **2**. The only difference being that now the firmware had to take some initial action to turn the healthy processor pair into a "hot spare" by idling it and making it appear to the OS as though it was not functioning. Thus, the LIDs would be swapped. That is, the LID register of the boot processor that lost lockstep is copied into the LID register of the now spare processor pair (CPU **3** in this example), and the LID register of this spare processor pair would be copied into the LID register of the original boot processor. Lockstep is then reestablished on the original boot processor, and it can be reintroduced to the OS with the ID of the original non-boot processor that replaced it as the system boot processor. The processor that was formerly the boot processor may be designated, in device tree **203**, as the spare for the now boot processor.

Turning to FIG. **4**, an exemplary operational flow diagram for switching the role of boot processor to a hot spare processor according to one embodiment is shown. In this case, an LOL has been detected for the boot processor. In operational block **401**, the system firmware allocates space to save the status of the good processor of the lockstep pair of processors for which LOL was detected. Exemplary techniques for determining which of the lockstep pair of processors had the error are disclosed in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/972,835 (U.S. Published Patent Application 20060107114) titled "SYSTEM AND METHOD FOR USING INFORMATION RELATING TO A DETECTED LOSS OF LOCKSTEP FOR DETERMINING A RESPONSIVE ACTION," the disclosure of which is hereby incorporated herein by reference. Accordingly, an appropriate amount of data storage space for storing the cache of the "good" processor in the lockstep pair of processors is allocated in block **401**. For instance, an amount of SRAM may be allocated for storing the registers from the good processor.

In operational block **402**, the state of the good processor is stored to the allocated storage space. For instance, in this illustrative example, PAL is called to dump the processor state of the good processor of the current boot processor (CBP) module (for which LOL was detected) to the allocated storage space.

In operational block **403**, the LID of the current boot processor (for which LOL was detected) is reprogrammed to a different value. For instance, the current boot processor may be assigned a LID that is reserved just for this situation (i.e., for switching the role of boot processor). Accordingly, the boot processor for which LOL was detected will not respond to the interrupts addressed for its original LID because the boot processor's LID value is reprogrammed to a different value.

In operational block **404**, the state of the good processor that was stored in block **402** to the allocated storage space is stored to the hot spare processor. In this example, the PAL is called to "Restore Processor State" on the hot spare processor, wherein the PAL is provided a pointer to the address to which the good processor's state was stored in block **402**. Accordingly, all the values from that storage space will be read into the registers of the spare processor. As a result, the LID value of the spare processor is programmed with the value of the boot processor that was saved to the allocated storage space. Accordingly, the spare processor automatically, as a result of receiving this stored state, receives the Local ID corresponding to the boot processor for interrupt handling.

In operational block **405**, the system firmware updates its device tree information to reflect that the hot spare is assuming the role of boot processor. The system firmware further updates its device tree, in this example, to indicate that the former boot processor (for which LOL was detected) is assuming the role of hot spare for the new boot processor.

In operational block **406**, PAL_MC_RESUME (which is a standard PAL call for resuming normal execution after a machine check (MC)) is called for the hot spare (which is assuming the role of boot processor) to cause the OS to recognize the hot spare processor as available for processing instructions, wherein the OS recognizes the hot spare as the boot processor. Thus, the hot spare has now assumed the role of the boot processor, taking the place of the former boot processor for which LOL was detected. Accordingly, in operational block **407** the former hot spare (which has assumed the role of boot processor) resumes normal operation as the system's boot processor with its lockstep protection enabled. Thus, in this exemplary embodiment, upon

detection of LOL for a boot processor, the role of boot processor is switched to a hot spare that has lockstep protection, wherein the system effectively recovers lockstep for its boot processor without requiring a system shut down (i.e., without shutting down the OS). In certain implementations, the firmware may go on to take action to attempt to reestablish lockstep for the former boot processor (e.g., by resetting the former boot processor), and if successful in reestablishing its lockstep, such former boot processor may be held in reserve as a hot spare processor for the new boot processor in the event that LOL is detected for this new boot processor.

Turning to FIG. **5**, an exemplary operational flow diagram for switching the role of boot processor to an active spare processor according to one embodiment is shown. Again, an LOL has been detected for the boot processor. In operational block **501**, the system firmware, responsive to such detection of LOL for the boot processor, examines the device tree to determine which processor module is assigned the role of active spare. In operational block **502** the system firmware (e.g., SAL) asserts a General Purpose Event Interrupt (GPE) by writing to the appropriate GPE register in the ACPI register space. This causes the OS to execute its interrupt handler, which will run. an AML Method associated with this particular interrupt. In this example, this AML method that is executed responsive to the interrupt generated by the GPE includes an AML instruction called "Notify". This "Notify" operation indicates the determined active spare processor object and that an Eject is being requested as arguments. Accordingly, an AML method is used to request the OS to eject the active spare that was determined in block **501**. In response, the OS idles the active spare processor and returns control over the active spare processor to the system firmware, in block **503**.

In operational block **504**, the system firmware allocates space to save the status of the good processor of the lockstep pair of processors for which LOL was detected (i.e., the lockstep pair of processors of the CBP). Exemplary techniques for determining which of the lockstep pair of processors had the error are disclosed in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/972,835 (U.S. Published Patent Application No. 20060107114) titled "SYSTEM AND METHOD FOR USING INFORMATION RELATING TO A DETECTED LOSS OF LOCKSTEP FOR DETERMINING A RESPONSIVE ACTION," the disclosure of which is hereby incorporated herein by reference. Accordingly, an appropriate amount of data storage space for storing the cache of the "good" processor in the lockstep pair of processors is allocated in block **504**. For instance, an amount of SRAM may be allocated for storing the registers from the good processor.

In operational block **505**, the state of the good processor is stored to the allocated storage space. For instance, in this illustrative example, PAL is called to dump the processor state of the good processor of the current boot processor (CBP) module (for which LOL was detected) to the allocated storage space.

In operational block **506**, the LID of the current boot processor (for which LOL was detected) is reprogrammed to a different value. For instance, the current boot processor may be assigned a LID that is reserved just for this situation (i.e., for switching the role of boot processor). Accordingly, the boot processor for which LOL was detected will not respond to the interrupts addressed for its original LID because the boot processor's LID value is reprogrammed to a different value.

In operational block **507**, the state of the good processor that was stored in block **505** to the allocated storage space is stored to the idled spare processor. In this example, the PAL is called to "Restore Processor State" on the active spare processor, wherein the PAL is provided a pointer to the address to which the good processor's state was stored in block **505**. Accordingly, all the values from that storage space will be read into the registers of the spare processor. As a result, the LID value of the spare processor is programmed with the value of the boot processor that was saved to the allocated storage space. Accordingly, the spare processor automatically, as a result of receiving this stored state, receives the Local ID (LID) corresponding to the boot processor for interrupt handling.

In operational block **508**, the system firmware updates its device tree information to reflect that the active spare is assuming the role of boot processor. The system firmware further updates its device tree, in this example, to indicate that the former boot processor (for which LOL was detected) is assuming the role of active spare for the new boot processor.

In operational block **509**, PAL_MC_RESUME (which is a standard PAL call for resuming normal execution after a machine check (MC)) is called for the active spare (which is assuming the role of boot processor) to cause the OS to recognize the active spare processor as available for processing instructions, wherein the OS recognizes the active spare as the boot processor. Thus, the active spare has now assumed the role of the boot processor, taking the place of the former boot processor for which LOL was detected. Accordingly, in operational block **510** the former active spare (which has assumed the role of boot processor) resumes normal operation as the system's boot processor with its lockstep protection enabled. Thus, in this exemplary embodiment, upon detection of LOL for a boot processor, an active spare processor is idled, and the role of boot processor is switched to such idled spare that has lockstep protection, wherein the system effectively recovers lockstep for its boot processor without requiring a system shut down (i.e., without shutting down the OS). In certain implementations, the firmware may go on to take action to attempt to reestablish lockstep for the former boot processor (e.g., by resetting the former boot processor), and if successful in reestablishing its lockstep, such former boot processor may be reintroduced to the OS as an active spare processor for the new boot processor in the event that LOL is detected for this new boot processor.

The exemplary procedures of FIGS. **4** and **5** switch the role of boot processor to a spare processor in accordance with certain embodiments. While exemplary operational flows are provided in FIGS. **4** and **5**, embodiments hereof are not limited to those illustrative examples. For instance, FIG. **6** shows a more general operational flow diagram for certain embodiments. In operational block **601**, LOL is detected for a processor in a multi-processor system. In operational block **602**, the system (e.g., system firmware) determines whether the processor for which LOL is detected is the system's boot processor. As described above, this is determined in certain embodiments by accessing the node of the device tree corresponding to the processor for which LOL is detected and determining if the information in that node of the device tree indicates that the processor is the system's boot processor.

In operational block **603**, responsive to determining that the processor for which LOL is detected is the system's boot processor, the system's firmware switches the role of boot processor to a spare processor, which may be either a hot spare or an active spare, as described above. Accordingly, the system's OS recognizes the spare processor as the system boot processor. In view of the above, embodiments hereof enable the role of boot processor to be dynamically switched, during system runtime, to a spare processor responsive to a

detected LOL for the boot processor, thus enabling recovery from the LOL without requiring that the system be crashed or re-booted.

What is claimed is:

1. A method comprising:

detecting loss of lockstep (LOL) for a processor in a multi-processor system;

determining that the processor for which said LOL is detected is assigned a role of boot processor;

changing a system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected, to a different value, which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor; and

changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to a spare processor without shutting down the system's operating system.

2. The method of claim **1** wherein the spare processor is a hot spare that is held in idle until switched the role of boot processor.

3. The method of claim **1** wherein the spare processor is an active spare.

4. The method of claim **3** wherein said switching further comprises:

causing the system's operating system to idle the active spare.

5. The method of claim **1** wherein the spare processor is a processor module that includes a master and slave processors that operate in lockstep.

6. The method of claim **1** wherein said determining that the processor for which said LOL is detected is assigned the role of boot processor comprises:

accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor.

7. The method of claim **1** further comprising:

determining the processor that is assigned the role of spare processor.

8. The method of claim **7** wherein said determining the processor that is assigned the role of spare processor comprises:

accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of spare processor.

9. The method of claim **1** wherein said determining step is performed by system firmware.

10. The method of claim **1** wherein said switching step is performed by system firmware.

11. The method of claim **1** wherein the processor for which said LOL is detected includes a lockstep pair of processors, and wherein said switching comprises:

determining one of the lockstep pair of processors that is not the cause of said LOL; and

copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to said spare processor.

12. The method of claim **11** wherein said spare processor includes a lockstep pair of processors, and wherein said copying comprises:

copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to each of said lockstep pair of processors of said spare processor.

13. A method comprising:

system firmware determining that loss of lockstep (LOL) is detected for a lockstep pair of processors that are assigned the role of boot processor in a system;

determining one of said lockstep pair of processors that is not the cause of said LOL;

copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to a spare processor;

changing a system identifier, used by the system's operating system to recognize the boot processor, of said lockstep pair of processors that are assigned the role of boot processor, to a different value, which causes the s system's operating system to not recognize the lock step pair of processors for which said LOL is detected as the processor; and

changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to said spare processor.

14. The method of claim **13** wherein the spare processor is a hot spare that is held in idle until switched the role of boot processor.

15. The method of claim **13** wherein the spare processor is an active spare.

16. The method of claim **15** further comprising:

causing the system's operating system to idle the active spare.

17. The method of claim **13** wherein said switching causes the system's operating system to recognize said spare processor as said boot processor without shutting down said operating system.

18. The method of claim **13** wherein said determining step is performed by system firmware.

19. The method of claim **13** wherein said copying step is performed by system firmware.

20. The method of claim **13** wherein said switching step is performed by system firmware.

21. A system comprising:

a plurality of processor modules that each include a master processor and a slave processor that operate in lockstep;

an operating system;

error detection logic operable to detect loss of lockstep (LOL) for at least one of said processor modules; and

system firmware operable, responsive to the detection of LOL for a first of said processor modules, to determine whether said first processor module is assigned a role of system boot processor;

wherein if determined that said first processor module is assigned the role of system boot processor, said system firmware is further operable to cause said operating system to recognize another processor module as system boot processor without shutting down said operating system.

22. The system of claim **21** wherein the another processor module is a processor module assigned a role of spare processor.

23. The system of claim **21** wherein the another processor module is held in idle until said system firmware causes said operating system to recognize said another processor module as said system boot processor.

24. The system of claim **21** wherein the system firmware causes the operating system to idle the another processor module before causing said operating system to recognize said another processor module as said system boot processor.

25. The system of claim 21 further comprising:

non-volatile data storage storing information that assigns the role of system boot processor to one of said plurality of processor modules.

26. A system comprising:

system firmware means for detecting loss of lockstep (LOL) for a processor in a multi-processor system;

means for determining whether the processor for which said LOL is detected is assigned the role of boot processor; and

system firmware means for switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which said LOL is detected is assigned the role of boot processor.

27. The system of claim 26 wherein said processor in said multi-processor system is an Itanium Processor Family (IPF) processor.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 7,624,302 B2
APPLICATION NO. : 10/972588
DATED : November 24, 2009
INVENTOR(S) : Scott L. Michaelis et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

In column 20, line 13, in Claim 13, after "the" delete "s".

In column 20, line 16, in Claim 13, before "processor;" insert -- boot --.

In column 21, line 8, in Claim 26, before "means" insert -- system firmware --.

Signed and Sealed this

Sixteenth Day of February, 2010

David J. Kappos

David J. Kappos
Director of the United States Patent and Trademark Office

UNIX Administration

A Comprehensive Sourcebook for Effective Systems and Network Management

© 2002 by CRC Press LLC

4

System Startup and Shutdown

4.1 Introductory Notes

UNIX systems run continuously under normal circumstances. Shutting down and powering-off a UNIX system should be done rarely, usually only when a hardware upgrade is being performed or a system is being allocated, or occasionally when another action requiring a system shutdown is performed. In real life, system shutdown is more frequent, because unpredictable situations always occur.

Power-cycling a UNIX system is not the only way the system can be shut down. Rebooting is also a familiar task for any UNIX administrator; UNIX administrators know well how system rebooting can be healthy for overall system maintenance.

Nevertheless, keeping the UNIX system running is the most visible task of a system administrator. If the system crashes, everyone will complain, your phone will ring constantly, and you will find yourself anxiously trying to fix the problem and bring the system back into production. Quickly you will learn how important the system you are in charge of really is, and how many users depend on it. Even more important, you will learn how crucial a smooth, fast, proper system startup can be.

This chapter covers topics related to normal UNIX system startup and shutdown procedures. Invoking a system startup and shutdown is quite simple; the main requirement is to be the superuser on the system (an easy task for an administrator). On the other hand, making the system behave correctly, especially during startup, requires a great deal of knowledge and administrative skill. Proper system startup is supposed to customize and set the myriad of existing system configuration files that will control each portion of the UNIX system. Some of these files include system-related configuration data, but there are also site-added applications; the bottom line is that the system should be fully operational after any system startup.

Given the complexity of properly configuring the system startup, this chapter could easily be located at the end of the overall text, rather than at its beginning. However, discussing the administration of a running UNIX system without knowing how that system came to be running seems strange; it is as though we are talking about administering a nonexistent UNIX system. So this material remains in the beginning by design; it will focus on the topic of global system startup and shutdown, and we will return to individual startup and shutdown issues later, whenever it is appropriate in discussing specific UNIX topics.

© 2002 by CRC Press LLC

From an administrative standpoint, system shutdown is the simpler procedure; at the end of the procedure a system must terminate all running processes, dismount all filesystems, and stop any other system activity. System shutdown works even if we never touch the default shutdown procedure — or perhaps it is better to say it *mostly* works, because the author of this text has witnessed a UNIX system that could not be shut down from the command line, and the only choice was to power-cycle the system. Our administrative task is to provide a graceful system shutdown. Everything must be stopped in a regular way, or the administrator will have to use the brute force method of power-cycling. System startup, on the other hand, must be done properly or the system will never come up. Obviously, more attention should be paid to system startup, and we will spend much more time discussing the startup procedure than the shutdown process.

System startup is often referred to as ***system booting***. Although "booting" specifies only one phase in the overall system startup, the two terms are commonly interchanged, as you will see in this chapter. Strictly speaking, system startup has a broader meaning than system booting.

All UNIX systems must be shut down in a regular way before any further action can be taken. You should never directly power-off UNIX systems (such as DOS-based PCs); the shutdown procedure must be implemented, otherwise disk data integrity can be corrupted (a UNIX filesystem could be damaged). The corruption can range between a relatively benign loss of data to heavy filesystem damage, which in the worst case scenario can leave a system unbootable.

The two major UNIX platforms BSD and System V have different startup and shutdown procedures, with, of course, the main differences occurring in startup. Among existing commercial UNIX flavors, the System V approach is more common; it provides more flexibility and some other administrative advantages. However, the BSD approach is somewhat easier to understand, and we will start our discussion with the BSD startup/shutdown procedure. Once the startup/shutdown concept is well understood, it will be easy to continue with the System V procedure.

4.2 System Startup

The system startup procedure is a continuous process that a UNIX system goes through, from its initial hardware-determined stage until the final production-ready stage. However, this unique system journey passes through several distinct phases, and each of these phases has its specific characteristics. The startup phases, listed in the order they occur, are:

- Bootstrap program execution
- Kernel execution
- *rc* system initialization
- Terminal line initialization

It is easier to understand the system startup procedure when the whole process is divided into several phases and each of the phases is analyzed separately, so this is the approach we will take. Although each of the listed phases is equally important for successful system startup, the ***system initialization*** phase requires the most administrative attention, so most of the following discussion will address this phase.

© 2002 by CRC Press LLC

In each of the startup phases, the system learns enough to execute the next phase. Each phase contributes a bit to the overall system startup. At the very beginning, the system does not know very much; at the very end, the system is ready for multi-user operations.

4.2.1 The Bootstrap Program

The origin of the word *boot* (as in, "to boot the system") is *bootstrapping*, which is the process of bringing a computer system to life and ready for use. ("Bootstrapping" is actually the "nerd word" for starting up a computer.) The computer system itself is only a collection of hardware resources (registers, arithmetic/logical unit, program counter, memories, etc.) capable of executing a sequence of instructions that make a program. The program, stored in the computer's memory (any kind of memory: ROM, RAM, external magnetic memory, etc.), defines the system's activity at every moment, including its first steps during the system startup.

An initial program, the ***bootstrap program***, must be stored in the fast nonvolatile memory directly accessible by a processor, or CPU (CPU stands for central processing unit and is another term for a processor). This portion of the computer memory is known as internal read-only memory (ROM). The execution of the bootstrap program is always automatically initiated when the system is powered-on or when a system hardware reset is applied. It is also initiated when the system is rebooted from the system console. Only the initial part of the bootstrap program, the part sufficient to bring the system into a workable state to deal with other memory types, must be stored in ROM. Once this level is achieved, the bootstrap program execution can be continued from another nonvolatile media such as a hard disk, a floppy disk, a tape, or a CD-ROM, or even through the network from a boot-server (in the case of diskless workstations). For UNIX systems, regular system booting is commonly executed from a hard disk, while first-time UNIX OS installation is performed from a CD-ROM (not long ago, a tape was used).

The system has to learn enough from the ROM to be able to access a disk to continue the bootstrap program, but it still assumes a simple flat data structure on the disk. A complex disk data organization such as the UNIX filesystem data structure is still too complicated for the system at this stage; more learning is needed to deal with a filesystem. That is why the rest of the bootstrap program is stored in a special part of the disk known as the *boot partition* (sometimes also known as the *boot segment*). The main characteristic of the boot partition is its easy access and flat data structure, so the system is able to continue with the bootstrap program execution, and further learning.

The ratio of the percentage of the bootstrap program stored in the ROM versus the disk boot partition varied through time. In the early days of UNIX when only low capacity, expensive ROM was available, the first part of the bootstrap program was reduced to the bare minimum size. Today, systems include high-density ROM sufficient to store quite sophisticated bootstrap programs; this makes boot partitions less important, although they are still a part of every system startup.

Once the bootstrap program is completely executed, the system is knowledgeable enough to continue with the kernel execution.

Traditionally UNIX presents the only OS running on underlying hardware; and traditionally this is a proprietary hardware for that UNIX flavor. This fact makes a booting process unique and quite straightforward. However, once PC hardware also became common in the UNIX arena, a more flexible booting, with UNIX as one of several choices, emerged as a preferable system characteristic. Linux is an example.

© 2002 by CRC Press LLC

On the Linux platform, the three most common booting mechanisms are:

- To boot Linux from the floppy, and leave hard drive for other OSs
- To use the ***Linux loader (LILO)***, the most common case
- To run ***Loadlin***, an MS-DOS program that boots Linux from within DOS

What is exceptional with LILO is the possibility of configuring this loader in different ways to match different needs; multiple-choice booting, including a non-UNIX startup, is also possible. The configured loader should then be installed in the boot sector of the first disk, known as MBR (master boot record). When the system is started, the PC BIOS transfers control to MBR and triggers a corresponding LILO booting. Linux provides an easy LILO configuration through its */etc/lilo.conf* configuration file, and the command ***lilo*** for its installation as MBR.

4.2.2 The Kernel Execution

The bootstrap program is responsible for loading the *UNIX kernel* into the system memory. The *kernel image*, originally named ***unix*** under System V, or ***vmunix*** under BSD, is intentionally located in the *root filesystem*, because the root filesystem is the first filesystem the system mounts to access data. Mounting is a UNIX-specific procedure that makes data on the disk accessible. (We discuss this issue in great detail in following chapters.) In the past, the kernel image was located in the root directory for easier access, but today, it usually resides in a separate subdirectory. We do not refer to "mounting" the kernel; we usually just say that the kernel image was loaded into the system memory and its execution was started.

The *kernel* manages all system hardware; all hardware drivers are part of the kernel, and the only OS access to the system hardware is through the kernel. Therefore the system hardware will be available upon the completion of this phase.

Once control passes to the *kernel*, it prepares itself to run the system by initializing its internal tables and completing the hardware diagnostics that are part of the boot process. The level of diagnostics implemented varies from one UNIX flavor to another. At the very end, the kernel verifies the integrity of the root filesystem and remounts it, and starts three programs that create three basic processes. Two of them, named ***kernel processes***, function wholly within the kernel in the kernel's privileged execution mode. They are actually portions of the kernel itself, only "dressed" like processes for scheduling reasons.

On BSD systems, the two processes are:

1. ***Swapper*** (process #0), responsible for the "*swapping*" — to schedule the transfer of whole processes between the main system memory and a mandatory swap partition on the primary disk when system resources are low
2. ***Pagedeamon*** (process #2), responsible for supporting the memory-management system regarding paging — a regular transfer of data in the pages between the main system memory and the disk

On System V systems, the processes are named differently: ***sched*** for the process #0, while the process #2 is replaced with *various memory handlers*.

The third process created by the system is the ***init*** process (process #1), which performs all administrative tasks during the system startup and shutdown. The ***init*** process is an extremely important process that enables the creation of all subsequent processes (in UNIX a process can be created only by another parent process). The ***init*** process has the PID=1, and it is the ancestor of all subsequent UNIX processes and the direct parent of each user's login shell.

© 2002 by CRC Press LLC

In the case of diskless workstations, the procedure is slightly different. Obviously, the kernel cannot be read from a nonexisting root filesystem; therefore, it must be downloaded from the network. Further kernel activities are adapted to the diskless environment.

The kernel is quite verbose and it prints messages on the console that report on the current execution status, total memory used and free, and some other information. However, the information reported varies among different UNIX flavors.

4.2.3 The Overall System Initialization

The ***init*** process does the rest of the work needed to bring the system into its final stage:

- Mounting the remaining local disk partitions
- Performing some filesystem cleanup
- Bringing on major UNIX subsystems (accounting, printing, etc.)
- Setting the system's name and time zone
- Starting the network
- Mounting remote filesystems
- Enabling user logins

4.2.3.1 rc Initialization Scripts

Most of the initialization activities are specified and carried out by means of the system ***rc initialization scripts*** stored in the ***/etc*** directory and its subdirectories. *Rc initialization scripts* are usually named in the way that they include the acronym **rc** as part of their names (as a prefix, a suffix, or in a fullpath name). **rc** stands for **run-command** and basically explains the purpose of the scripts. These mostly ***Bourne shell programs*** are organized differently on BSD and System V platforms, although their purpose is the same. As with any other script, rc initialization scripts are readable, so we can manage them in a very comprehensive way. Besides that, rc scripts are sufficiently verbose during execution, and this is a great help if the system hangs midway through the startup, or if there are any other related problems.

Main administrative activities are related to this phase. System site-related customization means editing the rc initialization scripts. Any system upgrade means to upgrade (or to add) rc initialization scripts; any startup modification means to do something with rc initialization scripts. The rest of this section exclusively addresses these issues. Afterward, a full picture of the necessary administration in this segment should be complete.

4.2.3.2 Terminal Line Initialization

The terminal line initialization is a part of the overall system initialization; however, the implemented initialization technique is quite different than that in the *rc system initialization*, which is sufficient reason to handle this topic separately. UNIX is extremely cautious with terminal line initialization — terminal lines are "gates" to the outside world. Users access the system via terminal lines, and the essence of UNIX existence is to serve users.

Once the initialization scripts have been executed, the system is fully operational, except for the fact that no one can log in to the system. In order to provide login via a particular terminal line, there must be a corresponding controlling process listening on it (usually the ***getty*** process, but the ***ttymon*** is used on the Solaris platform). At the final initialization phase, ***init*** spawns the ***getty*** processes to all indicated terminals and the startup procedure

© 2002 by CRC Press LLC

is completed. Today, users typically log in over a network using pseudo-terminals; however, the ***getty*** program is still doing its job.

Terminal line initialization is fully covered in Chapter 11.

4.2.4 System States

Once the initialization activities are completed, the UNIX system enters the ***multi-user mode***, and users may log in to the system. But ***init*** can also place the system in ***single-user mode*** instead of completing the initialization tasks required for multi-user mode. The single-user mode corresponds to a functionally reduced UNIX system. In single-user mode, a UNIX system looks very much like a personal computer. The single-user mode is primarily dedicated to administrative and maintenance activities that require complete control over the system. The user has all superuser privileges.

In some cases, the system will automatically enter single-user mode if there are any problems in the boot process that the system cannot handle on its own (for example, filesystem problems that **fsck** cannot fix), so the system administrator must resolve the problem. The ***init*** simply spawns the Bourne shell on the system's console and waits for it to terminate before continuing with the rest of the startup sequence. Entering <CTRL-D> or the exit command from the shell prompt can terminate the spawned single-user shell. Once this is done, the system may continue into multi-user mode.

Single-user mode represents a minimal system startup with no daemons running, so many UNIX facilities are disabled. Only the root filesystem is mounted (in the most common case) and a restricted number of commands are available (commands residing in the root filesystem). Under normal circumstances, other filesystems can be mounted by hand to access other commands.

Single-user mode can be a security problem for a system, because full control over the system is granted. On older UNIX systems, no password was required, but physical access to the system was required in the single-user mode. On some systems, a front panel lock with *normal* (*secure*) vs. *maintenance* (*service*) key positions enabled multi-user vs. single-user mode; the system protection was the key, and only authorized personnel could acquire the key. Modern UNIX systems usually require a root password to enter single-user mode. None of these approaches are perfect, and all of them have some disadvantage. A request for the root password could make difficulties under different circumstances, if the root password was forgotten.

While the BSD flavored system could be in one of three possible states — *off*, *single-user*, and *multi-user mode* — the System V platform explicitly defines a series of system states, called *run-levels* designated by a one-character name. System V run-levels are flavor dependent; an example is listed in the following table:

Run-Level	Name and Uses
0	Power-down state => safe to power-off the system
1	Administrative state
s or S	Single-user mode (on many systems same as 1)
2	Multi-user mode for stand-alone system
3	Multi-user mode for networked system, possibly sharing disks with other systems => via RFS, TCP/IP, and NFS, or some other protocol
4	Unused => can be user defined locally
5	Firmware state => for maintenance and running diagnostics, and for booting from an alternate not-root disk
6	Shutdown and reboot state => to reboot system from some running state (s, 2, 3, or 4); the system is taken down (to run-level 0) and then rebooted back

© 2002 by CRC Press LLC

To display the current system run-level, the following command is available:

```
$ who -r
. run-level   3   Mar   14   11:14   3   0   S
```

The system was taken to run-level 3, from run-level S, via run-level 0, on March 14, at 11:14. The leading dot is by default at the beginning of the line.

On the System V platform, movement between run-levels is managed by ***init***, and each run-level is controlled by its own set of initialization scripts.

4.2.5 The Outlook of a Startup Procedure

UNIX systems are configured to boot automatically when powered-on. If this is not possible, systems enter some form of the "ROM monitor mode" — a restricted ROM resident command interpreter that enables essential diagnostics, booting, and some other basic system activities. The ROM monitor mode is also the state that the system enters after being shut down; in that state, a system can be safely powered off. On some systems there is also a keystroke combination to enter this mode — for example on Sun Microsystems systems, the key (STOP-A) followed by the specific ROM monitor prompt "*OK*>."

The ROM monitor always provides the boot command, specified as "*b*" or "*boot*," among the other commands it provides. Certain options sufficient to control the system startup when problems are encountered (to boot the system from different media, into different modes, etc.) are also provided. The default booting media is the hard disk.

On old UNIX systems, manual booting from the ROM monitor was a two-stage procedure:

1. The boot command first loaded a boot program with a stand-alone shell (actually a mini-operating system).
2. A second command was then issued in a stand-alone shell to load UNIX kernel.

This two-step procedure looked like this:

```
>b
$$ unix
```

Different prompts specify two steps in the boot procedure. The technology available in the past limited the bootstrap program possibilities, which led to a more complicated startup procedure.

Today all UNIX flavors provide a relatively verbose system startup; a number of messages are directed to the console indicating the stage and status of the startup procedure. It is highly recommended that you monitor the system startup on the console. Otherwise, some trouble messages can remain undetected, which leads to a high probability for later surprises.

The startup sequences for two system user modes are presented in Figures 4.1 and 4.2. The UNIX system named "***atlas***" is running Solaris 2.x.; brief comments follow.

© 2002 by CRC Press LLC




```
---------- bootstrap program starts ----------
SPARCstation 20 (1 X 390Z50), Keyboard Present
ROM Rev. 2.19, 32 MB memory installed, Serial #7491530
Ethernet address 8:0:20:72:4f:ca, Host ID: 72724fca.

Rebooting with command:
Boot device: /iommu/sbus/espdma@f, 400000/esp@f, 800000/sd@3,0 File and args:
---------- bootstrap program ends, and kernel starts ----------
SunOS Release 5.4 Version generic [UNIX (R) System V Release 4.0]
Copyright (c) 1983-1984, Sun Microsystems, Inc.
---------- kernel ends, and rc initialization starts ----------
configuring network interfaces: le0.
Hostname: atlas
The system is coming up. Please wait.
checking ufs filesystems
/dev/rdsk/c0t2d0s6: is clean
/dev/rdsk/c0t3d0s7: is clean
/dev/rdsk/c0t2d0s0: is clean
Flushing routing table:
add net default: gateway 146.95.8.250
starting rpc services: rcpbind keyserv kerbd done.
Setting netmask of le0 to 255.255.255.0
Setting default interface for multicast: add net 244.0.0.0: gateway atlas.ph.hunter.cuny.edu
syslog service started.
Print services started.
volume management starting.
HTTP service starting.
The system is ready.
---------- rc initialization ends, and terminal line initialization starts ----------

atlas console login:
```

FIGURE 4.1
An illustration of a multiple-user startup sequence.

The Sun logo and first five lines are printed from the ***bootstrap*** program. These lines list basic system configuration and identification data, as well as the kind of boot device. The somewhat cryptic description of a boot device indicates an SCSI disk. The ***kernel*** prints only two identification lines that include the system version and release. Other lines are printed from initialization scripts invoked by the program ***init***. One of the lines indicates that the system was customized. The message that indicates the start of the HTTP service is not a part of a regular OS installation — obviously, this site has been customized to provide an Internet service. At the end, the login prompt is displayed upon the console initialization.

The startup procedure includes filesystem checking, one of the most important activities performed by the **fsck** utility (**fsck** is discussed in greater detail in Chapter 5). The filesystem verifications are different on BSD and System V platforms. BSD checks all filesystems on every boot; System V does not check filesystems if they were dismounted normally when the system last went down (the **fsstat** command is used for this purpose), and faster booting is enabled. Filesystem checking can result in the display of many messages depending on the current filesystem status. If more serious filesystem corruption is encountered, the system is left in single-user mode, and manual filesystem checking and repair by the administrator may be required.

A single-user startup sequence is much shorter, and it includes the *boot* and *kernel* lines. The next two lines about the network interface configuration and host's name are printed from corresponding initialization scripts involved in the system single-user startup. Finally, the console is activated and the user is informed of two possibilities:

© 2002 by CRC Press LLC

1. Enter the system in single-user mode by entering the root password
2. Or continue with multi-user startup by entering [Ctrl-D]

If [Ctrl-D] is entered, the system continues with the multi-user startup, as in the previous case.




```
---------- bootstrap program starts ----------
SPARCstation 20 (1 X 390Z50), Keyboard Present
ROM Rev. 2.19, 32 MB memory installed, Serial #7491530
Ethernet address 8:0:20:72:4f:ca, Host ID: 72724fca.

Rebooting with command: -s
Boot device: /iommu/sbus/espdma@f, 400000/esp@f, 800000/sd@3, 0 File and args: -s
---------- bootstrap program ends, and kernel starts ----------
SunOS Release 5.4 Version generic [UNIX (R) System V Release 4.0]
Copyright (c) 1983-1984, Sun Microsystems, Inc.
---------- kernel ends, and single-user rc initialization starts ----------
configuring network interfaces: le0.
Hostname: atlas

INIT: SINGLE USER MODE

Type Ctrl-d to proceed with normal startup,
(or give root password for system maintenance):

>>>>>>>>>>>>>>> Since Ctrl-D is entered <<<<<<<<<<<<<<<
---------- rc initialization continues ----------

INIT: New run level: 3
The system is coming up. Please wait.
checking ufs filesystems
/dev/rdsk/c0t2d0s6: is clean
/dev/rdsk/c0t3d0s7: is clean
/dev/rdsk/c0t2d0s0: is clean
Flushing routing table:
add net default: gateway 146.95.8.250
starting rpc services: rcpbind keyserv kerbd done.
Setting netmask of le0 to 255.255.255.0
Setting default interface for multicast: add net 244.0.0.0: gateway atlas.ph.hunter.cuny.edu
syslog service started.
Print services started.
volume management starting.
HTTP service starting.
The system is ready.
---------- rc initialization ends, and terminal line initialization starts ----------

atlas console login:
```

FIGURE 4.2
An illustration of a single-user startup sequence.

4.2.6 Initialization Scripts

Once the ***init*** process is born, the system startup is determined by a series of rc initialization scripts which define a detailed procedure to bring the system into the multi-user mode. This is the most common case, although other system modes (run-levels) are also possible. These files control all custom-defined and site-dependent items (there are multiple rc initialization scripts), and they are executed sequentially. Generally, rc initialization scripts represent Bourne shell script files, executable at any time and on any UNIX platform. (The Bourne shell is the default shell, and it is available at the very early system stage on every UNIX platform.) The rc initialization scripts do not differ from any other shell script, except at the time of their execution. (This, by the way, is why the prefix "rc" is used in their description, as well as in the name.) However, they can also be executed from the

© 2002 by CRC Press LLC

command line at any time, and administrators can make full use of this opportunity: on System V, individual function-specific initialization scripts are often used to stop and start specific UNIX functions during regular system production. On modern UNIX platforms, sometimes Korn shell rc initialization scripts are also included (for example, on the HP-UX platform) which indicated the early availability of the Korn shell.

Understanding rc initialization scripts is a vital part of system administration — this is the place for system customization. A system administrator must be familiar with these files, their locations and, in many cases, their contents. Only then is full control over the system startup possible, and quick corrective action can follow any problem encountered during system boot time. Each modification in the initialization scripts must be done very carefully with respect for the basic administrative rule: save original script files before making any changes. If this rule is not followed, various problems can ensue.

Despite the fact that rc initialization scripts on both UNIX platforms BSD and System V serve the same purpose, the mechanisms by which they are initiated and executed are quite different. These differences require great attention, knowledge, and skills from system administrators working in a heterogeneous environment, which is very common today. Today, the System V rc approach prevails — the System V organization of the rc initialization scripts offers more flexibility and other administrative advantages. We will discuss System V initialization in greater detail after a quick survey of the BSD-style initialization.

4.3 BSD Initialization

4.3.1 The BSD *rc* Scripts

Originally, the BSD initialization was controlled only by two rc initialization scripts: ***/etc/rc*** and ***/etc/rc.local***. A general system initialization was supported by the */etc/rc* script, while the */etc/rc.local* script referred to a local site, i.e., to issues that should be customized (probably a more appropriate script name would be "*rc.site*" to avoid any possible confusion toward the logical association with a "network-local relationship"). During system booting to the multi-user mode, ***init*** executed the *rc* script, which in turn executed the *rc.local* script. If a single-user boot was performed, scripts were only partially executed; the remaining parts were executed when the single-user shell was exited.

Having only two rc initialization scripts would lead one to believe that system maintenance was easy, but in fact the reality is quite the opposite. The work required for system initialization remained the same, regardless of how many rc scripts were involved, and huge rc script files were more difficult to manage and more vulnerable to corruption during modification. It could be very difficult to find an appropriate control sequence, items were often doubled, and so on.

SunOS introduced additional script files: ***/etc/rc.boot*** and ***/etc/rc.single***. The program ***init*** invokes first *rc.boot* script and from there *rc.single* (regardless of whether the system is booting to single vs. multi-user mode); then the */etc/rc* and */etc/rc.local* files follow.

4.3.2 BSD Initialization Sequence

For a clearer picture, the block diagram of the SunOS execution sequence is presented in Figure 4.3 (it is assumed the system is booting from the local disk). The SunOS organization made a clear distinction between single and multiple-user modes; it was immediately easier to follow any problems that developed in the system booting.

© 2002 by CRC Press LLC

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	PATENT NUMBER	GROUP ART UNIT	FILE WRAPPER LOCATION
10/972,588	7624302	2114	9200

OC000000081543932

Correspondence Address/Fee Address Change

The following fields have been set to Customer Number 56436 on 03/18/2016

- **Correspondence Address**
- **Maintenance Fee Address**

The address of record for Customer Number 56436 is:

56436
Hewlett Packard Enterprise
3404 E. Harmony Road
Mail Stop 79
Fort Collins, CO 80528

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	PATENT NUMBER	GROUP ART UNIT	FILE WRAPPER LOCATION
10/972,588	7624302	2114	9200

OC000000042094318

Correspondence Address/Fee Address Change

The following fields have been set to Customer Number 22879 on 06/14/2010

- **Correspondence Address**
- **Maintenance Fee Address**
- **Power of Attorney Address**

The address of record for Customer Number 22879 is:

22879
HEWLETT-PACKARD COMPANY
Intellectual Property Administration
3404 E. Harmony Road
Mail Stop 35
FORT COLLINS, CO 80528

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 7,624,302 B2
APPLICATION NO. : 10/972588
DATED : November 24, 2009
INVENTOR(S) : Scott L. Michaelis et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

In column 20, line 13, in Claim 13, after "the" delete "s".

In column 20, line 16, in Claim 13, before "processor;" insert -- boot --.

In column 21, line 8, in Claim 26, before "means" insert -- system firmware --.

Signed and Sealed this

Sixteenth Day of February, 2010

David J. Kappos

David J. Kappos
Director of the United States Patent and Trademark Office

IN THE UNITED STATES PATENT & TRADEMARK OFFICE

ATTORNEY DOCKET NUMBER: 200404973-1

IN RE APPLICATION OF: U.S. Patent No. 7,624,302

USPTO CONFIRMATION CODE: 1079

APPLICATION NO.: 10/972,588

FILED: Oct. 25, 2004

EXAMINER: DIEU MINH T LE

GROUP ART UNIT: 2114

FOR: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

37 CFR 1.322 & 37 CFR 1.323 REQUEST FOR CERTIFICATE OF CORRECTION

HONORABLE COMMISSIONER OF PATENTS & TRADEMARKS

Sir:

The following is a request for a certificate of correction in Serial Number 10/972,588, now Patent Number 7,624,302.

A certificate of correction under 35 USC 254 is respectfully requested in the above-identified patent.

☒ All errors were the fault of the USPTO, no fee required. In the event that a further fee is required, please charge the amount to Deposit Account No. 082025.

☐ All errors were the fault of the applicant and, accordingly, please charge **$100.00** to our Deposit Account No. 082025. In the event that a further fee is required, please charge the amount to the same Deposit Account.

☐ The errors were the fault of both the applicant and USPTO and, accordingly, please charge **$100.00** to our Deposit Account No. 082025. In the event that a further fee is required, please charge the amount to the same Deposit Account.

The exact locations where the errors appear in the patent and patent application are as follows:

In column 20, line 13, in Claim 13, after "the" delete "s".
(Amendments to Claims filed on Sep. 10, 2008, page 4, in claim 16, line 8)

In column 20, line 16, in Claim 13, before "processor;" insert - - boot - -.
(Amendments to Claims filed on Sep. 10, 2008, page 4, in claim 16, line 9)

In column 21, line 8, in Claim 26, before "means" insert - - system firmware - -.
(Amendments to Claims filed on Sep. 10, 2008, page 5, in claim 29, line 4)

The requested corrections are attached on Form PTO 1050.

Respectfully Submitted

1/7/2010
DATE

Name: Steven L. Webb
Registration No.: 44, 395

Attorney/Agent of Record

Hewlett-Packard Company
Intellectual Property Administration
3404 E. Harmony Road
Mail Stop 35
Fort Collins, CO 80528

PTO/SB/44 (04-05)
Approved for use through 04/30/2007. OMB 0651-0033
U.S. Patent and Trademark Office, U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.
(Also Form PTO-1050)

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 7,624,302

APPLICATION NO. : 10/972,588

ISSUE DATE : Nov. 24, 2009

INVENTOR(S) : Scott L. Michaelis et al.

It is certified that an error appears or errors appear in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

In column 20, line 13, in Claim 13, after "the" delete "s".

In column 20, line 16, in Claim 13, before "processor;" insert - - boot - -.

In column 21, line 8, in Claim 26, before "means" insert - - system firmware - -.

MAILING ADDRESS OF SENDER (Please do not use customer number below):

Hewlett-Packard Company
Intellectual Property Administration
3404 E. Harmony Road
Mail Stop 35
Fort Collins, CO 80528

This collection of information is required by 37 CFR 1.322, and 1.323, and 1.324. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C 122 and 37 CFR 1.14. This collection is estimated to take 1.0 hour to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Attention Certificate of Corrections Branch, Commissioner for Patent, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Electronic Acknowledgement Receipt

EFS ID:	6814910
Application Number:	10972588
International Application Number:	
Confirmation Number:	1079
Title of Invention:	SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR
First Named Inventor/Applicant Name:	Scott L. Michaelis
Customer Number:	22879
Filer:	Steven L. Webb/Marci Santillanes
Filer Authorized By:	Steven L. Webb
Attorney Docket Number:	200404973-1
Receipt Date:	14-JAN-2010
Filing Date:	25-OCT-2004
Time Stamp:	18:47:41
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Request for Certificate of Correction	200404973-1.pdf	499942 c70148e5f678803473a8ec708963c97677afcd3c	no	3

Warnings:	
Information:	

Total Files Size (in bytes):	499942

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	ISSUE DATE	PATENT NO.	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/972,588	11/24/2009	7624302	200404973-1	1079

22879 7590 11/04/2009

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
3404 E. Harmony Road
Mail Stop 35
FORT COLLINS, CO 80528

ISSUE NOTIFICATION

The projected patent number and issue date are specified above.

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)
(application filed on or after May 29, 2000)

The Patent Term Adjustment is 513 day(s). Any patent to issue from the above-identified application will include an indication of the adjustment on the front page.

If a Continued Prosecution Application (CPA) was filed in the above-identified application, the filing date that determines Patent Term Adjustment is the filing date of the most recent CPA.

Applicant will be able to obtain more detailed information by accessing the Patent Application Information Retrieval (PAIR) WEB site (http://pair.uspto.gov).

Any questions regarding the Patent Term Extension or Adjustment determination should be directed to the Office of Patent Legal Administration at (571)-272-7702. Questions relating to issue and publication fee payments should be directed to the Application Assistance Unit (AAU) of the Office of Data Management (ODM) at (571)-272-4200.

APPLICANT(s) (Please see PAIR WEB site http://pair.uspto.gov for additional applicants):

Scott L. Michaelis, Plano, TX;
Anurupa Rajkumari, Round Rock, TX;
William B. McHardy, Fort Collins, CO;

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/972,588	10/25/2004	Scott L. Michaelis	200404973-1	1079

22879 7590 10/20/2009

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
3404 E. Harmony Road
Mail Stop 35
FORT COLLINS, CO 80528

EXAMINER	
LE, DIEU MINH T	
ART UNIT	PAPER NUMBER
2114	

NOTIFICATION DATE	DELIVERY MODE
10/20/2009	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

JERRY.SHORMA@HP.COM
ipa.mail@hp.com
jessica.l.fusek@hp.com

UNITED STATES DEPARTMENT OF COMMERCE
U.S. Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450

APPLICATION NO./ CONTROL NO.	FILING DATE	FIRST NAMED INVENTOR / PATENT IN REEXAMINATION	ATTORNEY DOCKET NO.
10972588	10/25/04	MICHAELIS ET AL.	200404973-1

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
3404 E. Harmony Road
Mail Stop 35
FORT COLLINS, CO 80528

EXAMINER
Dieu-Minh Le

ART UNIT	PAPER
2114	20091014

DATE MAILED:

Please find below and/or attached an Office communication concerning this application or proceeding.

Commissioner for Patents

In response to the communication filed on 10/05/2009, The reference cited in this IDS has been considered by the Examiner.

/Dieu-Minh Le/
Primary Examiner, Art Unit 2114

PTO-90C (Rev.04-03)

RECEIVED
CENTRAL FAX CENTER
OCT 05 2009

PTO/SB/08b (07-09)
Approved for use through 07/31/2012. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 1

Complete if Known	
Application Number	10/972,588
Filing Date	October 25, 2009
First Named Inventor	Scott L. Michaelis,et.al
Art Unit	2114
Examiner Name	Dieu-Minh Le
Attorney Docket Number	200404973-1

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
/M.L./		Notice of Allowance, dated July 23, 2009, serial number 10/973,077, 20 pages	

Examiner Signature	/Dieu Minh Le/	Date Considered	10/14/2009

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

1 Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

APPLE INC
OCT 06 2009
PATENT & TRADEMARK OFFICE

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail Mail Stop ISSUE FEE
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (571)-273-2885

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

22879 7590 07/14/2009

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

10/06/2009 RMEBRAH1 00000174 082025 10972588

01 FC:1501 1510.00 DA
02 FC:1504 300.00 DA

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission
I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (571) 273-2885, on the date indicated below.

Marci Santillanes	(Depositor's name)
Marci Santillanes	(Signature)
10-06-09	(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/972,588	10/25/2004	Scott L. Michaelis	200404973-1	1079

TITLE OF INVENTION: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

APPLN. TYPE	SMALL ENTITY	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	$1510	$300	$0	$1810	10/14/2009

EXAMINER	ART UNIT	CLASS-SUBCLASS
LE, DIEU MINH T	2114	714-012000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).
☐ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.
☐ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. Use of a Customer Number is required.

2. For printing on the patent front page, list
(1) the names of up to 3 registered patent attorneys or agents OR, alternatively,
(2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.
1 ____
2 ____
3 ____

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE: Hewlett-Packard Development Company, L.P.
(B) RESIDENCE: (CITY and STATE OR COUNTRY): Houston, Texas

Please check the appropriate assignee category or categories (will not be printed on the patent): ☐ Individual ☒ Corporation or other private group entity ☐ Government

4a. The following fee(s) are submitted:
☒ Issue Fee
☒ Publication Fee (No small entity discount permitted)
☐ Advance Order - # of Copies ____

4b. Payment of Fee(s): (Please first reapply any previously paid issue fee shown above)
☐ A check is enclosed.
☐ Payment by credit card. Form PTO-2038 is attached.
☒ The Director is hereby authorized to charge the required fee(s), any deficiency, or credit any overpayment, to Deposit Account Number 08-2025 (enclose an extra copy of this form).

5. Change in Entity Status (from status indicated above)
☐ a. Applicant claims SMALL ENTITY status. See 37 CFR 1.27.
☐ b. Applicant is no longer claiming SMALL ENTITY status. See 37 CFR 1.27(g)(2).

NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

Authorized Signature ____ Date 10/5/09

Typed or printed name Steven L. Webb Registration No. 44,395

This collection of information is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, Virginia 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PAGE 1/2 * RCVD AT 10/6/2009 10:11:25 AM [Eastern Daylight Time] * SVR:USPTO-EFXRF-6/10 * DNIS:2732885 * CSID:970 898 2590 * DURATION (mm-ss):01-14

OCT 06 2009 PATENT & TRADEMARK OFFICE

COPY

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail Mail Stop ISSUE FEE
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (571)-273-2885

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

22879 7590 07/14/2009

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission
I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (571) 273-2885, on the date indicated below.

Marci Santillanes	(Depositor's name)
Marci Santillanes	(Signature)
10-06-09	(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/972,588	10/25/2004	Scott L. Michaelis	200404973-1	1079

TITLE OF INVENTION: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

APPLN. TYPE	SMALL ENTITY	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	$1510	$300	$0	$1810	10/14/2009

EXAMINER	ART UNIT	CLASS-SUBCLASS
LE, DIEU MINH T	2114	714-012000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).
- ☐ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.
- ☐ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. Use of a Customer Number is required.

2. For printing on the patent front page, list
(1) the names of up to 3 registered patent attorneys or agents OR, alternatively,
(2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.
1 _____
2 _____
3 _____

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE: Hewlett-Packard Development Company, L.P.
(B) RESIDENCE: (CITY and STATE OR COUNTRY): Houston, Texas

Please check the appropriate assignee category or categories (will not be printed on the patent): ☐ Individual ☒ Corporation or other private group entity ☐ Government

4a. The following fee(s) are submitted:
- ☒ Issue Fee
- ☒ Publication Fee (No small entity discount permitted)
- ☐ Advance Order - # of Copies _____

4b. Payment of Fee(s): (Please first reapply any previously paid Issue Fee shown above)
- ☐ A check is enclosed.
- ☐ Payment by credit card. Form PTO-2038 is attached.
- ☒ The Director is hereby authorized to charge the required fee(s), any deficiency, or credit any overpayment, to Deposit Account Number 08-2025 (enclose an extra copy of this form).

5. Change in Entity Status (from status indicated above)
- ☐ a. Applicant claims SMALL ENTITY status. See 37 CFR 1.27.
- ☐ b. Applicant is no longer claiming SMALL ENTITY status. See 37 CFR 1.27(g)(2).

NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

Authorized Signature _____ Date 10/5/09

Typed or printed name Steven L. Webb Registration No. 44,395

This collection of information is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, Virginia 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PAGE 2/2 * RCVD AT 10/6/2009 10:11:25 AM [Eastern Daylight Time] * SVR:USPTO-EFXRF-6/10 * DNIS:2732885 * CSID:970 898 2590 * DURATION (mm-ss):01-14

RECEIVED
CENTRAL FAX CENTER
OCT 05 2009

PTO/SB/08b (07-09)
Approved for use through 07/31/2012. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 1

Complete if Known	
Application Number	10/972,588
Filing Date	October 25, 2009
First Named Inventor	Scott L. Michaelis,et.al
Art Unit	2114
Examiner Name	Dieu-Minh Le
Attorney Docket Number	200404973-1

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		Notice of Allowance, dated July 23, 2009, serial number 10/973,077, 20 pages	

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

1 Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

RECEIVED
CENTRAL FAX CENTER
OCT 05 2009

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 200404973-1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): Scott L. Michaelis et al

Confirmation No.: 1079

Application No.: 10/972,588

Examiner: Dieu-Minh Le

Filing Date: Oct 25, 2004

Group Art Unit: 2114

Title: System And Method For Switching The Role Of Boot Processor To A Spare Processor Responsive To Detection Of Loss Of Lockstep In A Boot Processor

Mail Stop Amendment
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

INFORMATION DISCLOSURE STATEMENT

This Information Disclosure Statement is submitted:

☐ under 37 CFR 1.97 (b), or
(Within three months of filing national application; or date of entry of national application; or before mailing date of first action on the merits; whichever occurs last)

☐ under 37 CFR 1.97 (c), together with either a:
☐ Statement under 37 CFR 1.97 (e), or
☐ a $180 fee under 37 CFR 1.17 (p), or
(After the CFR 1.97 (b) time period, but before final action or notice of allowance, whichever occurs first)

☒ under 37 CFR 1.97 (d), together with a:
☒ Statement under 37 CFR 1.97 (e)(1) or (2), and
☒ a $180 fee set forth in 37 CFR 1.17 (p)
(Filed after final action, a notice of allowance, on or before payment of the issue fee)

Please charge to Deposit Account 08-2025 the sum of $ 180.00 . At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25.

Applicant(s) submit herewith form PTO/SB/08B - Information Disclosure Statement together with any required copies of patents, publications or other information of which applicant(s) are aware, which applicant(s) believe(s) may be material to the examination of this application and for which there may be a duty to disclose in accordance with 37 CFR 1.56.

☐ A concise explanation of the relevance of foreign language patents, foreign language publications and other foreign language information listed on PTO/SB/08B, as presently understood by the individual(s) designated in 37 CFR 1.56 (c) most knowledgeable about the content is given on the attached sheet, or where a foreign language patent is cited in a search report or other action by a foreign patent office in a counterpart foreign application, an english language version of the search report or action which indicates the degree of relevance found by the foreign office is listed on form PTO/SB/08B and is enclosed.

It is requested that the information disclosed herein be made of record in this application.

☐ I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, Alexandria, VA 22313-1450

Date of Deposit:

OR

☒ I hereby certify that this paper is being transmitted to the Patent and Trademark Office facsimile number (571) 273-8300

Date of facsimile: 10.05.09

Typed Name: Marci Santillanes

Signature: Marci Santillanes

Respectfully submitted,

Scott L. Michaelis et al

By _____

Steven L Webb

Attorney/Agent for Applicant(s)

Reg No.: 44,395

Date: 10/5/09

Telephone: 970 898 3451

Rev 10/08 (IDSCert)

10/06/2009 HMARZI1 00000010 082025 10972588

01 FC:1806 180.00 DA

RECEIVED
CENTRAL FAX CENTER
OCT 05 2009

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 200404973-1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): Scott L. Michaelis et al

Confirmation No.: 1079

Application No.: 10/972588

Examiner: Dieu-Minh Le

Filing Date: Oct 25, 2004

Group Art Unit: 2114

Title: System And Method For Switching The Role Of Boot Processor To A Spare Processor Responsive To Detection Of Loss Of Lockstep In A Boot Processor

Mail Stop Amendment
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

STATEMENT UNDER 37 CFR 1.97 (e)

The undersigned hereby certifies that either:

☐ Each item of information contained in the Information Disclosure Statement was cited for the first time in any communication from a foreign patent office in a counterpart foreign application, not more than three months prior to the filing of the statement, or

☒ No item of information contained in the Information Disclosure Statement

– was cited in a communication from a foreign patent office in a counterpart foreign application, and

– to the knowledge of the undersigned, after making reasonable inquiry, was known to any designated in 37 CFR 1.56 (c) more than three months prior to the filing of the Information Disclosure Statement.

☐ I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, Alexandria, VA 22313-1450

Date of Deposit:

OR

☒ I hereby certify that this paper is being transmitted to the Patent and Trademark Office facsimile number (571) 273-8300.

Date of facsimile: 10.05.09

Typed Name: Marci Santillanes

Signature: Marci Santillanes

Respectfully submitted,

Scott L. Michaelis et al

By _____

Steven L Webb

Attorney/Agent for Applicant(s)

Reg No. : 44,395

Date : 10/5/09

Telephone : 970 898 3451

Rev 10/00 (Stat197e)

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

NOTICE OF ALLOWANCE AND FEE(S) DUE

22879 7590 07/14/2009

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

EXAMINER	
LE, DIEU MINH T	
ART UNIT	PAPER NUMBER
2114	

DATE MAILED: 07/14/2009

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/972,588	10/25/2004	Scott L. Michaelis	200404973-1	1079

TITLE OF INVENTION: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

APPLN. TYPE	SMALL ENTITY	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	$1510	$300	$0	$1810	10/14/2009

THE APPLICATION IDENTIFIED ABOVE HAS BEEN EXAMINED AND IS ALLOWED FOR ISSUANCE AS A PATENT. PROSECUTION ON THE MERITS IS CLOSED. THIS NOTICE OF ALLOWANCE IS NOT A GRANT OF PATENT RIGHTS. THIS APPLICATION IS SUBJECT TO WITHDRAWAL FROM ISSUE AT THE INITIATIVE OF THE OFFICE OR UPON PETITION BY THE APPLICANT. SEE 37 CFR 1.313 AND MPEP 1308.

THE ISSUE FEE AND PUBLICATION FEE (IF REQUIRED) MUST BE PAID WITHIN THREE MONTHS FROM THE MAILING DATE OF THIS NOTICE OR THIS APPLICATION SHALL BE REGARDED AS ABANDONED. THIS STATUTORY PERIOD CANNOT BE EXTENDED. SEE 35 U.S.C. 151. THE ISSUE FEE DUE INDICATED ABOVE DOES NOT REFLECT A CREDIT FOR ANY PREVIOUSLY PAID ISSUE FEE IN THIS APPLICATION. IF AN ISSUE FEE HAS PREVIOUSLY BEEN PAID IN THIS APPLICATION (AS SHOWN ABOVE), THE RETURN OF PART B OF THIS FORM WILL BE CONSIDERED A REQUEST TO REAPPLY THE PREVIOUSLY PAID ISSUE FEE TOWARD THE ISSUE FEE NOW DUE.

HOW TO REPLY TO THIS NOTICE:

I. Review the SMALL ENTITY status shown above.

If the SMALL ENTITY is shown as YES, verify your current SMALL ENTITY status:

A. If the status is the same, pay the TOTAL FEE(S) DUE shown above.

B. If the status above is to be removed, check box 5b on Part B - Fee(s) Transmittal and pay the PUBLICATION FEE (if required) and twice the amount of the ISSUE FEE shown above, or

If the SMALL ENTITY is shown as NO:

A. Pay TOTAL FEE(S) DUE shown above, or

B. If applicant claimed SMALL ENTITY status before, or is now claiming SMALL ENTITY status, check box 5a on Part B - Fee(s) Transmittal and pay the PUBLICATION FEE (if required) and 1/2 the ISSUE FEE shown above.

II. PART B - FEE(S) TRANSMITTAL, or its equivalent, must be completed and returned to the United States Patent and Trademark Office (USPTO) with your ISSUE FEE and PUBLICATION FEE (if required). If you are charging the fee(s) to your deposit account, section "4b" of Part B - Fee(s) Transmittal should be completed and an extra copy of the form should be submitted. If an equivalent of Part B is filed, a request to reapply a previously paid issue fee must be clearly made, and delays in processing may occur due to the difficulty in recognizing the paper as an equivalent of Part B.

III. All communications regarding this application must give the application number. Please direct all communications prior to issuance to Mail Stop ISSUE FEE unless advised to the contrary.

IMPORTANT REMINDER: Utility patents issuing on applications filed on or after Dec. 12, 1980 may require payment of maintenance fees. It is patentee's responsibility to ensure timely payment of maintenance fees when due.

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail **Mail Stop ISSUE FEE**
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (571)-273-2885

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

22879 7590 07/14/2009

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (571) 273-2885, on the date indicated below.

(Depositor's name)
(Signature)
(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/972,588	10/25/2004	Scott L. Michaelis	200404973-1	1079

TITLE OF INVENTION: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

APPLN. TYPE	SMALL ENTITY	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	$1510	$300	$0	$1810	10/14/2009

EXAMINER	ART UNIT	CLASS-SUBCLASS
LE, DIEU MINH T	2114	714-012000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).

❑ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.

❑ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. **Use of a Customer Number is required.**

2. For printing on the patent front page, list

(1) the names of up to 3 registered patent attorneys or agents OR, alternatively,

(2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.

1 _____

2 _____

3 _____

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE (B) RESIDENCE: (CITY and STATE OR COUNTRY)

Please check the appropriate assignee category or categories (will not be printed on the patent) : ❑ Individual ❑ Corporation or other private group entity ❑ Government

4a. The following fee(s) are submitted:

❑ Issue Fee

❑ Publication Fee (No small entity discount permitted)

❑ Advance Order - # of Copies _____

4b. Payment of Fee(s): **(Please first reapply any previously paid issue fee shown above)**

❑ A check is enclosed.

❑ Payment by credit card. Form PTO-2038 is attached.

❑ The Director is hereby authorized to charge the required fee(s), any deficiency, or credit any overpayment, to Deposit Account Number _____ (enclose an extra copy of this form).

5. **Change in Entity Status** (from status indicated above)

❑ a. Applicant claims SMALL ENTITY status. See 37 CFR 1.27.

❑ b. Applicant is no longer claiming SMALL ENTITY status. See 37 CFR 1.27(g)(2).

NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

Authorized Signature _____ Date _____

Typed or printed name _____ Registration No. _____

This collection of information is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, Virginia 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PTOL-85 (Rev. 08/07) Approved for use through 08/31/2010. OMB 0651-0033 U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/972,588	10/25/2004	Scott L. Michaelis	200404973-1	1079

22879 7590 07/14/2009

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

EXAMINER	
LE, DIEU MINH T	
ART UNIT	PAPER NUMBER
2114	

DATE MAILED: 07/14/2009

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)

(application filed on or after May 29, 2000)

The Patent Term Adjustment to date is 529 day(s). If the issue fee is paid on the date that is three months after the mailing date of this notice and the patent issues on the Tuesday before the date that is 28 weeks (six and a half months) after the mailing date of this notice, the Patent Term Adjustment will be 529 day(s).

If a Continued Prosecution Application (CPA) was filed in the above-identified application, the filing date that determines Patent Term Adjustment is the filing date of the most recent CPA.

Applicant will be able to obtain more detailed information by accessing the Patent Application Information Retrieval (PAIR) WEB site (http://pair.uspto.gov).

Any questions regarding the Patent Term Extension or Adjustment determination should be directed to the Office of Patent Legal Administration at (571)-272-7702. Questions relating to issue and publication fee payments should be directed to the Customer Service Center of the Office of Patent Publication at 1-(888)-786-0101 or (571)-272-4200.

Notice of Allowability	Application No.	Applicant(s)	
	10/972,588	MICHAELIS ET AL.	
	Examiner	Art Unit	
	Dieu-Minh Le	2114	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to *the RCE filed 5/12/09*.

2. ☒ The allowed claim(s) is/are *1-12 and 16-30 [now as 1-27]*.

3. ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
   a) ☐ All b) ☐ Some* c) ☐ None of the:
      1. ☐ Certified copies of the priority documents have been received.
      2. ☐ Certified copies of the priority documents have been received in Application No. _____ .
      3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

   * Certified copies not received: _____.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application.
THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.

4. ☐ A SUBSTITUTE OATH OR DECLARATION must be submitted. Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL PATENT APPLICATION (PTO-152) which gives reason(s) why the oath or declaration is deficient.

5. ☐ CORRECTED DRAWINGS (as "replacement sheets") must be submitted.
   (a) ☐ including changes required by the Notice of Draftsperson's Patent Drawing Review (PTO-948) attached
      1) ☐ hereto or 2) ☐ to Paper No./Mail Date _____.
   (b) ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date _____.

   Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).

6. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1. ☐ Notice of References Cited (PTO-892)
2. ☐ Notice of Draftperson's Patent Drawing Review (PTO-948)
3. ☒ Information Disclosure Statements (PTO/SB/08), Paper No./Mail Date 5/12/09, 5/28/09, 5/29/09, 6/1/09, 6/2/09, and 6/3/09
4. ☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
5. ☐ Notice of Informal Patent Application
6. ☐ Interview Summary (PTO-413), Paper No./Mail Date _____ .
7. ☐ Examiner's Amendment/Comment
8. ☐ Examiner's Statement of Reasons for Allowance
9. ☐ Other _____.

/Dieu-Minh Le/
Primary Examiner, Art Unit 2114

PTO/SB/08b (03-09)
Approved for use through 04/30/2009. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 1

Complete if Known	
Application Number	10/972,588
Filing Date	10/25/2004
First Named Inventor	Scott L. Michaelis
Art Unit	2114
Examiner Name	Le, Dieu Minh T.
Attorney Docket Number	200404973-1

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		Notice of Allowance for Serial No. 10/972,835 dated 11-20-2008 in the United States	
		Notice of Allowance for Serial No. 10/972,835 dated 8-1-2008 in the United States	
		Office Action for Serial No. 10/972,835 dated 3-3-2008 in the United States	
		Office Action for Serial No. 10/972,835 dated 7-12-2007 in the United States	

Examiner Signature	/Dieu Minh Le/	Date Considered	05/29/2009

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

1 Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /M.L./

PTO/SB/08b (03-09)
Approved for use through 04/30/2009. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 1

Complete if Known	
Application Number	10/972,588
Filing Date	10/25/2004
First Named Inventor	Scott L. Michaelis
Art Unit	2114
Examiner Name	Le, Dieu Minh T.
Attorney Docket Number	200404973-1

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		Office Action for Serial No. 10/973,003 dated 3-5-2008 in the United States	
		Office Action for Serial No. 10/973,003 dated 9-12-2007 in the United States	

Examiner Signature	/Dieu Minh Le/	Date Considered	06/17/2009

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

1 Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /M.L./

PTO/SB/08b (03-09)
Approved for use through 04/30/2009. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 1

Complete if Known	
Application Number	10/972,588
Filing Date	10/25/2004
First Named Inventor	Scott L. Michaelis
Art Unit	2114
Examiner Name	Le, Dieu Minh T.
Attorney Docket Number	200404973-1

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		Notice of Allowance for Serial No. 10/973,076 dated 11-28-2008 in the United States	
		Notice of Allowance for Serial No. 10/973,076 dated 7-16-2008 in the United States	
		Notice of Allowance for Serial No. 10/973,076 dated 6-3-2008 in the United States	
		Final Office Action for Serial No. 10/973,076 dated 2-27-2008 in the United States	
		Office Action for Serial No. 10/973,076 dated 9-12-2007 in the United States	
		Notice of Allowance for Serial No. 10/972,888 dated 11-14-2007 in the United States	
		Office Action for Serial No. 10/972,888 dated 7-18-2007 in the United States	

Examiner Signature	/Dieu Minh Le/	Date Considered	06/17/2009

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

1 Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /M.L./

PTO/SB/08b (03-09)
Approved for use through 04/30/2009. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 1

Complete if Known	
Application Number	10/972,588
Filing Date	10/25/2004
First Named Inventor	Scott L. Michaelis
Art Unit	2114
Examiner Name	Le, Dieu Minh T.
Attorney Docket Number	200404973-1

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		Notice of Allowance for Serial No. 10/973,004 dated 8-1-2007 in the United States	
		Office Action for Serial No. 10/973,004 dated 4-10-2007 in the United States	
		Final Office Action for Serial No. 10/973,075 dated 5-12-2009 in the United States	
		Office Action for Serial No. 10/973,075 dated 10-29-2008 in the United State	
		Office Action for Serial No. 10/973,075 dated 3-12-2008 in the United States	
		Office Action for Serial No. 10/973,075 dated 7-2-2007 in the United States	
		Notice of Allowance for Serial No. 10/972,796 dated 12-19-2007 in the United States	
		Office Action for Serial No. 10/972,796 dated 7-12-2007 in the United States	

Examiner Signature	/Dieu Minh Le/	Date Considered	06/17/2009

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

1 Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /M.L./

PTO/SB/08b (03-09)
Approved for use through 04/30/2009. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 1

Complete if Known	
Application Number	10/972,588
Filing Date	10/25/2004
First Named Inventor	Scott L. Michaelis
Art Unit	2114
Examiner Name	Le, Dieu Minh T.
Attorney Docket Number	200404973-1

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
/M.L./		SHOICHI ARAKI, Real-Time Tracking of Multiple Moving Object Contours in a Moving Camera Image Sequence, Vol. E83-D IEICE Trans. Inf. & Syst, 7-7-2000, 9 pages	
no date		~~DONG KYU KIM, Active Object Tracking Using Image Mosaic Background, 4 pages~~	
/M.L./		JIANG-BIN ZHENG, Proceedings of the First International Conference on Machine Learning and Cybernetics, Beijing, 4-5 November 2002, 4 pages	✓
/M.L./		JONG RYUL KIM, Automatic Localization and Trackign of Moving Objects Using Adaptive Snake Algorithm, ICICS-PCM 2003, 15-18 December 2003, 5 pages, Singapore	
/M.L./		JIANHOA MENG, CVEPS-A Compressed Video Editing and Parsing System, 18-11-1996, Pages 1-11	

Examiner Signature	/Dieu Minh Le/	Date Considered	06/17/2009

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

1 Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

Issue Classification	Application/Control No.	Applicant(s)/Patent Under Reexamination
	10972588	MICHAELIS ET AL.
	Examiner	**Art Unit**
	Dieu-Minh Le	2114

ORIGINAL		INTERNATIONAL CLASSIFICATION	
CLASS	SUBCLASS	CLAIMED	NON-CLAIMED
714	12	G 0 6 F 11 / 00 (2006.01)	

CROSS REFERENCE(S)					
CLASS	SUBCLASS (ONE SUBCLASS PER BLOCK)				
714	11				

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original
1	1	14	17												
2	2	15	18												
3	3	16	19												
4	4	17	20												
5	5	18	21												
6	6	19	22												
7	7	20	23												
8	8	21	24												
9	9	22	25												
10	10	23	26												
11	11	24	27												
12	12	25	28												
	13	26	29												
	14	27	30												
	15														
13	16														

NONE		**Total Claims Allowed:**	
(Assistant Examiner)	(Date)	27	
/Dieu-Minh Le/ Primary Examiner.Art Unit 2114	07/02/2009	O.G. Print Claim(s)	O.G. Print Figure
(Primary Examiner)	(Date)	1	1

EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	315	spare adj processor	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/01 14:19
L2	229	(loss lose los$3 lost$3) same (lockstep lock-step "lock step")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/01 14:19
L3	95	(detect$3 track$3 monitor$3) same L2	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/01 14:19
L4	22	L3 and L1	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/01 14:19
L5	39197	(pair spare slave redundan$4 backup back-up "back up" second$4) adj2 processor	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/01 14:31
L6	46413	multi-processor "multi processor" (plural$3 adj processor)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/01 14:31
L7	6914	L6 and L5	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/01 14:31

L8	21	4 and 7	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/01 14:32
L9	33	(michaelis-scott$ or myer-sylvia$ or mchardy-william$).in.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/01 14:32
L10	33	L9	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/01 14:32
L11	869	(714/11).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/01 14:32
L12	421	(714/12).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/01 14:32
L13	763	(714/13).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/01 14:32
L14	738	(714/10).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/01 14:32
L15	117	(712/31).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/01 14:32

L16	2421	L11 L12 L13 L14 L15	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/01 14:32
L17	2637	(boot$3 initializ$5 power-up) adj processor	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/01 14:34
L18	17	8 and 17	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/01 15:04
L19	16	((operating adj system) OS) and 18	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/01 15:36
L20	160944	firmware	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/01 15:39
L21	16	19 and L20	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/01 15:39
L22	6	(11 12 13 14 15) and 3 and 6 and 17	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/01 15:40
L23	13	("714"/$.ccls.) and 3 and 6 and 17	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/01 15:40

L24	0	("711"/$.ccls.) and 3 and 6 and 17	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/01 15:41
S1	14	(("7085959") or ("20050172164") or ("5675794") or ("6438687") or ("6920581") or ("5915082") or ("6625749")).PN.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/26 17:54
S2	2	("7085959").PN.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/27 15:05
S3	730	(714/11).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/27 15:06
S4	337	(714/12).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 15:32
S5	606	(714/13).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/27 15:07
S6	36	("20020144177" | "20030070050" | "20040006722" | "20040019771" | "20040153857" | "5226152" | "5249188" | "5751932" | "5758058" | "5764660" | "6065135" | "6148348" | "6473869" | "6560682" | "6604177" | "6615366" | "6625749" | "6687851" | "6754787").PN.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/27 15:08

S7	44	("20020144177" | "20030070050" | "20040006722" | "20040019771" | "20040153857" | "6625749" | "6687851" | "6754787")".PN"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/27 15:09
S9	2222	lockstep lock-step	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 15:21
S11	25008	(loss adj S9) LOL	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 15:22
S12	51	(detect$3 or track$4) adj S11	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 15:23
S13	1547	(boot$4 or initializ$4) adj processor	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:41
S14	246	spare adj processor	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 15:27
S15	12	S12 and S13 and S14	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 15:30
S16	25	(michaelis-scott$ or myer-sylvia$ or mchardy-william$).in.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:14

S18	632	(714/10).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 15:33
S19	95	(712/31).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:08
S20	52536	(switch$4 chang$4 failover fail-over "fail over") same (spare slave redundan$4 backup back-up "back up" second$4) same processor	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:54
S21	81	S20 same S13	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:28
S22	168219	without same ("shut$4 down" stop$4 idle interrupt$4) same (OS "operating system")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:33
S23	6	S21 and S22	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:33
S24	20342	multi-processor "multi processor" and S11 and S13	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:45
S25	176	S24 and S20 and S22	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:46

S26	7	S25 and S14	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:53
S27	87	(recover$4 reestablish$3 restart$3) with S9	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:54
S28	31827	(pair spare slave redundan$4 backup back-up "back up" second$4) adj2 processor	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:55
S29	101834	firmware	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:56
S30	21	S27 and S28 and S29	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:57
S31	15153	trigger$3 same ("operating system" OS operating-system) same (idle stop$4 interrupt$3 halt$3)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 14:49
S32	207	(loss lose los$3 lost$3) same (lockstep lock-step "lock step")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 14:52
S33	86	(detect$3 track$3 monitor$3) same S32	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 14:56

S34	3	S33 same S31	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 14:56
S35	3	S32 same S31	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 15:02
S36	10	S32 and S31	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 15:02
S37	33339	(pair spare slave redundan$4 backup back-up "back up" second$4) adj2 processor	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 15:05
S38	2323	lockstep lock-step	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 15:06
S39	92	(recover$4 reestablish$3 restart$3) with S38	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 15:06
S40	114677	firmware	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 15:06
S41	11	S37 same S39 same S40	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 15:06

S42	764	(714/11).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 15:07
S43	352	(714/12).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 15:07
S44	659	(714/13).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 15:07
S45	658	(714/10).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 15:07
S46	96	(712/31).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 15:07
S47	2	(S42 S43 S44 S45 S46) and S31 and S33	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 15:07
S48	2	(S42 S43 S44 S45 S46) and S31 and S32 and S37 and S40	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 15:08
S49	40917	multi-processor "multi processor" (plural$3 adj processor)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 12:32

S50	1742	(boot$4 or initializ$4) adj processor	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 12:32
S51	94	S50 same S49	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 12:32
S52	16	(assign$4 giv$3 order$3 decid$3 mak$3) same role same S50	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 12:35
S53	2459	lockstep lock-step	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:00
S54	25133	(loss adj S53) LOL	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:00
S55	56	(detect$3 or track$4) adj S54	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:00
S56	13	S51 same S55	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:00
S57	0	S56 not S52	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:01

S58	803	(714/11).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:01
S59	370	(714/12).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:01
S60	693	(714/13).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:01
S61	682	(714/10).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:01
S62	104	(712/31).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:01
S63	2237	S58 S59 S60 S61 S62	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:01
S64	803	(714/11).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:02
S65	289	spare adj processor	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:02

S66	15	S55 and S50 and S65	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:02
S67	221	(loss lose los$3 lost$3) same (lockstep lock-step "lock step")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:02
S68	92	(detect$3 track$3 monitor$3) same S67	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:02
S69	3	S68 and S64 and S66	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:02
S70	803	(714/11).ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/04 13:46
S71	370	(714/12).ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/04 13:46
S72	693	(714/13).ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/04 13:46
S73	563	(714/10).ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/04 13:46
S74	15199	without same (shutt$3 turn-off power-off) same ((operating adj system) OS)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 14:09
S75	9	S52 and S74	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 14:10
S76	0	S75 not S56	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 14:12

C:\ Documents and Settings\ MLe\ My Documents\ EAST\ Workspaces\ rce310972588.wsp

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	10972588	MICHAELIS ET AL.
	Examiner	Art Unit
	Dieu-Minh Le	2114

SEARCHED			
Class	Subclass	Date	Examiner
714	10, 11, 12, 13	6/30/09	dml
712	31	6/30/09	dml

SEARCH NOTES		
Search Notes	Date	Examiner
EAST search all files updated	6/30/09	dml
Inventor name search	6/30/09	dml
NPL search files [IEEE, Google Scholar]	7/1/09	dml

INTERFERENCE SEARCH			
Class	Subclass	Date	Examiner
714	10-13	7/2/09	dml
712	31	7/2/09	dml
	See Interference search data file		

	/Dieu-Minh Le/ Primary Examiner.Art Unit 2114

EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	869	(714/11).ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2009/07/02 06:05
L2	421	(714/12).ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2009/07/02 06:05
L3	763	(714/13).ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2009/07/02 06:05
L4	619	(714/10).ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2009/07/02 06:05
L5	117	(712/31).ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2009/07/02 06:05
L6	38825	multi-processor "multi processor" (plural$3 adj processor)	US-PGPUB; USPAT; USOCR	OR	ON	2009/07/02 06:06
L7	33570	(pair spare slave redundan$4 backup back-up "back up" second$4) adj2 processor	US-PGPUB; USPAT; USOCR	OR	ON	2009/07/02 06:06
L8	1727	(boot$4 or initializ$4) adj processor	US-PGPUB; USPAT; USOCR	OR	ON	2009/07/02 06:07
L9	58002	(switch$4 chang$4 failover fail-over "fail over") same (spare slave redundan$4 backup back-up "back up" second$4) same processor	US-PGPUB; USPAT; USOCR	OR	ON	2009/07/02 06:07
L10	212	(loss lose los$3 lost$3) same (lockstep lock-step "lock step")	US-PGPUB; USPAT; USOCR	OR	ON	2009/07/02 06:08
L11	210	(detect$3 track$3 monitor$3) same 8	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/02 06:12
L12	50	11 and 6 and 7	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/02 06:12

L13	4	12 and 9 and 10 and 1	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/02 06:12
L14	4	12 and 9 and 10 and 2	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/02 06:12
L15	0	12 and 9 and 10 and 3	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/02 06:13
L16	0	12 and 9 and 10 and 4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/02 06:13
L17	2	12 and 9 and 10 and 5	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/02 06:13
L18	13	("714"/$.ccls.) and 12 and 9 and 10	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/02 06:13
L19	0	("711"/$.ccls.) and 12 and 9 and 10	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/02 06:13
L20	2	("712"/$.ccls.) and 12 and 9 and 10	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/07/02 06:14

7/ 2/ 09 6:14:21 AM
C:\ Documents and Settings\ MLe\ My Documents\ EAST\ Workspaces\ rce310972588.wsp

PTO/SB/08b (03-09)
Approved for use through 04/30/2009. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 1

Complete if Known	
Application Number	10/972,588
Filing Date	10/25/2004
First Named Inventor	Scott L. Michaelis
Art Unit	2114
Examiner Name	LE, DIEU MINH T.
Attorney Docket Number	200404973-1

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		Notice of Allowance dated 2-11-2009 for Serial Number 10/973,077 in the United States	
		Notice of Allowance dated 9-23-2008 for Serial Number 10/973,077 in the United States	
		Office Action dated 3-19-2008 for Serial Number 10/973,077 in the United States	
		Office Action dated 9-20-2007 for Serial Number 10/973,077 in the United States	

Examiner Signature	/Dieu Minh Le/	Date Considered	05/29/2009

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

1 Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /M.L./

PTO/SB/08b (03-09)
Approved for use through 04/30/2009. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 1

Complete if Known	
Application Number	10/972,588
Filing Date	10/25/2004
First Named Inventor	Scott L. Michaelis
Art Unit	2114
Examiner Name	Le, Dieu Minh T.
Attorney Docket Number	200404973-1

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		Final Office Action for Serial No. 10/972,835 dated 11-21-2007 in the United States	
		Office Action for Serial No. 10/973,075 dated 12-13-2007 in the United States	

Examiner Signature	/Dieu Minh Le/	Date Considered	06/17/2009

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

1 Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /M.L./

Web Images Video Maps News Shopping Gmail more ▼

Sign in

Google

detecting loss of lockstep multi-processor boot processor switching spare proces Search Advanced Search Preferences

Web Show options... Results 1 - 10 of about 601 for detecting loss of lockstep multi-processor boot processor switching spare processor. (0.58 seconds)

1. System and method for configuring lockstep mode of a processor ...

A system comprising: a **processor** module that supports **lockstep** mode of operation ; ... "System and Method for **Switching** the Role of **Boot Processor** to a **Spare Processor** Responsive to **Detection** of **Loss of Lockstep** in a **Boot Processor**". ... and method for terminating **lock-step** sequences in a **multiprocessor** system ...
www.patentstorm.us/patents/7308566/claims.html - Similar

2. System and method for configuring lockstep mode of a processor ...

Dec 11, 2007 ... A system comprises a **processor** module that supports **lockstep** mode of operation. ... in a **Multi-Processor** System a **Spare Processor** That is in **Lockstep** for Use ... and Method for **Switching** the Role of **Boot Processor** to a **Spare Processor** Responsive to **Detection** of **Loss of Lockstep** in a **Boot Processor**". ...
www.freepatentsonline.com/7308566.html - Similar
by SL Michaelis - 2007

3. Self-checked, lock step processor pairs - Patent 6233702

A **multiprocessor** system includes a number of sub-**processor** systems, ... system for a **multiple processor** telecommunications **switching** control, February, 1983, De et al. .... 4907228 is also an example of this pair-and-**spare** technique, ... the local router will subsequently **detect** the **loss** of keep-alive symbols, ...
www.freepatentsonline.com/6233702.html - Similar
by RW Horst - 2001 - Cited by 11 - Related articles
More results from www.freepatentsonline.com »

4. [PDF] Software Optimization Guide for the AMD Hammer Processor

File Format: PDF/Adobe Acrobat
Chapter 11, "**Multiprocessor** Considerations," discusses **processor**/core selection and related ..... Many compilers have an "assume no aliasing" optimization **switch**. ..... The application is also typically responsible for **detecting** the ..... counters in a **processor** and these counters are incremented in **lockstep**. ...
www.amd.com/us-en/assets/content_type/white.../40546.pdf - Similar

5. **Processor** Editorial Article - Product Releases

May 8, 2009 ... Marathon Technologies everRun VM **Lockstep** & everRun 2G .... Manager is connected between a network patch panel and a **switch** or router, ... Updated every four hours, a list of precompiled malware-**detection** rules lets the ClamAV ... With an Intel Xeon Nehalem 5500 Series **processor**, the Stirling 118m ...
www.**processor**.com/editorial/article.asp?article...asp - Cached - Similar

6. MD-11 Electronic Instrument System - Digital Avionics Systems ...

Should a failure occur, the Master Caution or Master Warning **switch**/light .... Monitoring within each DEU is capable of **detecting** a **loss** of DU ... Each **processor** board is capable of running at a sustained rate of 3.6 ... d) 16Kbytes **Boot** PROM (UVPROM) .... **lock-step** processing to implement software partitioning. ...
ieeexplore.ieee.org/iel2/1033/6983/00282148.pdf?arnumber... - Similar
by J Morgan - 1992 - Cited by 4 - Related articles

7. Unixware 7 Non stop Clusters

A CPU dies, but its twin, which has been running in **lockstep** all this time, ... the hardware identically, and the **spare** sits around waiting for the need. .... NUMA clusters work by extending memory across **multiple** machines. .... Well, gee, if **processor** 1 is done, then it is available for more work, isn't it? ...
aplawrence.com/UW/clusters.html - Cached - Similar

8. Error **detection**/correction and fault **detection**/recovery inventions ...

Jul 19, 2007 ... The number of **spare** servers is reduced for improving the .... 20070174689 - Computer platform embedded operating system backup **switching** handling method and system: ... 20070174687 - Systems and methods for maintaining **lock step** ..... 20070168760 - Saving state data in parallel in a **multi-processor** ...
www.freshpatents.com/Error-**detection**-correction-and-fault-**detection**-recovery-dt200707ntc714.php - Cached - Similar

9. [PDF] HP ROM-Based Setup Utility User Guide

File Format: PDF/Adobe Acrobat - View as HTML

Selecting the primary **boot** controller. •. Configuring online **spare** memory .... **Lockstep** Mode provides increased data protection by enabling ..... **Multi-Processor** Specification (MPS) Table Mode (on page 105) ..... ROM-based utilities generically **switch** to run in either Windows® or VT100 mode to provide maximum ...

h20000.www2.hp.com/bc/docs/support/.../c00191707.pdf - Similar

10. NEC Fault-Tolerant Linux Server

A stock Linux kernel provides too little error **detection** and recovery for ... Both modules run the same instructions in **lockstep**, checking each ... After pressing the power **switch**, located under a hinged plastic ... We compared this server to a **spare** generic server having a single Athlon XP 2100+ **processor** using ...

www.linuxjournal.com/article/6709 - Cached - Similar

Gooooooooooogle ▶
1 2 3 4 5 6 7 8 9 10 Next

detecting loss of lockstep multi-processor boot processor switching spare proces Search

Search within results - Language Tools - Search Help - Dissatisfied? Help us improve - Try Google Experimental

Google Home - Advertising Programs - Business Solutions - Privacy - About Google

IEEE Xplore DIGITAL LIBRARY
Home | Login | Logout | Access Information | Alerts | Purchase History | Cart | Sitemap | Help
Welcome United States Patent and Trademark Office
Celebrating 125 Years of Engineering the Future
IEEE
Search Session History
BROWSE SEARCH IEEE XPLORE GUIDE SUPPORT

Edit an existing query or compose a new query in the Search Query Display.

Select a search number (#) to:

- Add a query to the Search Query Display
- Combine search queries using AND, OR, or NOT
- Delete a search
- Run a search

Wed, 1 Jul 2009, 4:03:16 PM EST

Search Query Display

Run Search | Reset

Recent Search Queries		Results
#1	(((detecting monitoring tracking)<in>metadata) <and> ((loss of lockstep)<in>metadata))<and> ((boot processor) <in>metadata)	0
#2	(((switching changing swapping)<in>metadata) <and> ((boot processor)<in>metadata))<and> ((firmware)<in>metadata)	0
#3	(((spare processor)<in>metadata) <and> ((system identifier) <in>metadata))<and> ((shutting down os)<in>metadata)	0
#4	(((lol multi-processor)<in>metadata) <and> ((firmware) <in>metadata))<and> ((switching os)<in>metadata)	0
#5	(((system identifier)<in>metadata) <and> ((different os) <in>metadata))<and> ((lol boot processor)<in>metadata)	0

Clear Search History

Help Contact Us Privacy & Security IEEE.org
Indexed by Inspec
© Copyright 2009 IEEE – All Rights Reserved

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	10972588	MICHAELIS ET AL.
	Examiner	Art Unit
	Dieu-Minh Le	2114

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	02/28/2008	09/11/2008	02/01/2009	07/02/2009					
1	1	✓	=	=	=					
2	2	✓	=	=	=					
3	3	✓	=	=	=					
4	4	✓	=	=	=					
5	5	✓	=	=	=					
6	6	✓	=	=	=					
7	7	✓	=	=	=					
8	8	✓	=	=	=					
9	9	✓	=	=	=					
10	10	✓	=	=	=					
11	11	✓	=	=	=					
12	12	✓	=	=	=					
	13	✓	-	-	-					
	14	✓	-	-	-					
	15	✓	-	-	-					
13	16	✓	=	=	=					
14	17	✓	=	=	=					
15	18	✓	=	=	=					
16	19	✓	=	=	=					
17	20	✓	=	=	=					
18	21	✓	=	=	=					
19	22	✓	=	=	=					
20	23	✓	=	=	=					
21	24	=	=	=	=					
22	25	=	=	=	=					
23	26	=	=	=	=					
24	27	=	=	=	=					
25	28	=	=	=	=					
26	29	✓	=	=	=					
27	30	✓	=	=	=					

PTO/SB/08b (03-09)
Approved for use through 04/30/2009. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 1

Complete if Known	
Application Number	10/972,588
Filing Date	10/25/2004
First Named Inventor	Scott L. Michaelis
Art Unit	2114
Examiner Name	Le, Dieu Minh T.
Attorney Docket Number	200404973-1

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		Final Office Action for Serial No. 10/972,835 dated 11-21-2007 in the United States	
		Office Action for Serial No. 10/973,075 dated 12-13-2007 in the United States	

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

1 Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

Electronic Acknowledgement Receipt

EFS ID:	5450360
Application Number:	10972588
International Application Number:	
Confirmation Number:	1079
Title of Invention:	SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR
First Named Inventor/Applicant Name:	Scott L. Michaelis
Customer Number:	22879
Filer:	Robert Wayne Nelson/Kalyn Black
Filer Authorized By:	Robert Wayne Nelson
Attorney Docket Number:	200404973-1
Receipt Date:	03-JUN-2009
Filing Date:	25-OCT-2004
Time Stamp:	21:26:30
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	NPL Documents	10973075-OA-12-13-07.pdf	645056 8548bb285aa89be5b10bf0d092bde1dadaed7df5	no	18

Warnings:	
Information:	

2	NPL Documents	10972835-FOA-11-21-2007.pdf	652102 dbcff4b32fed5018b91c5721cba5ee329542d7ce	no	17
Warnings:					
Information:					
3	Transmittal Letter	E-IDS.pdf	560184 6a3b0761b6a5cdd833e1dc8794f799218b6af88b	no	1
Warnings:					
Information:					
4	Information Disclosure Statement (IDS) Filed (SB/08)	sb0008b-6.pdf	190983 3e166d1d0390c17d4284a3842f4c6c9dfdbcac79	no	1
Warnings:					
Information:					
This is not an USPTO supplied IDS fillable form					
		Total Files Size (in bytes):	2048325		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 200404973-1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): Scott L. Michaelis	**Confirmation No.:** 1079
Application No.: 10/972,588	**Examiner:** Le, Dieu Minh T.
Filing Date: 10/25/2004	**Group Art Unit:** 2114

Title: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

Mail Stop Amendment
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

INFORMATION DISCLOSURE STATEMENT

This Information Disclosure Statement is submitted:

☒ under 37 CFR 1.97 (b), or
(Within three months of filing national application; or date of entry of national application; or before mailing date of first action on the merits; whichever occurs last)

☐ under 37 CFR 1.97 (c), together with either a:
- ☐ Statement under 37 CFR 1.97 (e), or
- ☐ a $180 fee under 37 CFR 1.17 (p), or

(After the CFR 1.97 (b) time period, but before final action or notice of allowance, whichever occurs first)

☐ under 37 CFR 1.97 (d), together with a:
- ☐ Statement under 37 CFR 1.97 (e)(1) or (2), and
- ☐ a $180 fee set forth in 37 CFR 1.17 (p)

(Filed after final action, a notice of allowance, on or before payment of the issue fee)

Please charge to Deposit Account **08-2025** the sum of $ 0.00 . At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account **08-2025** pursuant to 37 CFR 1.25.

Applicant(s) submit herewith form PTO/SB/08B - Information Disclosure Statement together with any required copies of patents, publications or other information of which applicant(s) are aware, which applicant(s) believe(s) may be material to the examination of this application and for which there may be a duty to disclose in accordance with 37 CFR 1.56.

☐ A concise explanation of the relevance of foreign language patents, foreign language publications and other foreign language information listed on PTO/SB/08B, as presently understood by the individual(s) designated in 37 CFR 1.56 (c) most knowledgeable about the content is given on the attached sheet, or where a foreign language patent is cited in a search report or other action by a foreign patent office in a counterpart foreign application, an english language version of the search report or action which indicates the degree of relevance found by the foreign office is listed on form PTO/SB/08B and is enclosed.

It is requested that the information disclosed herein be made of record in this application.

Respectfully submitted,

Scott L. Michaelis

By: /Robert W. Nelson/

Robert W. Nelson, Esq.

Attorney/Agent for Applicant(s)

Reg No. : 37,898

Date : June 3, 2009

Telephone : (303) 298-9888

PTO/SB/08b (03-09)
Approved for use through 04/30/2009. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 1

Complete if Known	
Application Number	10/972,588
Filing Date	10/25/2004
First Named Inventor	Scott L. Michaelis
Art Unit	2114
Examiner Name	Le, Dieu Minh T.
Attorney Docket Number	200404973-1

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		Notice of Allowance for Serial No. 10/973,004 dated 8-1-2007 in the United States	
		Office Action for Serial No. 10/973,004 dated 4-10-2007 in the United States	
		Final Office Action for Serial No. 10/973,075 dated 5-12-2009 in the United States	
		Office Action for Serial No. 10/973,075 dated 10-29-2008 in the United State	
		Office Action for Serial No. 10/973,075 dated 3-12-2008 in the United States	
		Office Action for Serial No. 10/973,075 dated 7-2-2007 in the United States	
		Notice of Allowance for Serial No. 10/972,796 dated 12-19-2007 in the United States	
		Office Action for Serial No. 10/972,796 dated 7-12-2007 in the United States	

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

1 Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 200404973-1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s):	**Scott L. Michaelis**	**Confirmation No.:**	**1079**
Application No.:	**10/972,588**	**Examiner:**	**Le, Dieu Minh T.**
Filing Date:	**10/25/2004**	**Group Art Unit:**	**2114**

Title: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

Mail Stop Amendment
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

INFORMATION DISCLOSURE STATEMENT

This Information Disclosure Statement is submitted:

☒ under 37 CFR 1.97 (b), or
(Within three months of filing national application; or date of entry of national application; or before mailing date of first action on the merits; whichever occurs last)

☐ under 37 CFR 1.97 (c), together with either a:
- ☐ Statement under 37 CFR 1.97 (e), or
- ☐ a $180 fee under 37 CFR 1.17 (p), or

(After the CFR 1.97 (b) time period, but before final action or notice of allowance, whichever occurs first)

☐ under 37 CFR 1.97 (d), together with a:
- ☐ Statement under 37 CFR 1.97 (e)(1) or (2), and
- ☐ a $180 fee set forth in 37 CFR 1.17 (p)

(Filed after final action, a notice of allowance, on or before payment of the issue fee)

Please charge to Deposit Account **08-2025** the sum of $ 0.00 . At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account **08-2025** pursuant to 37 CFR 1.25.

Applicant(s) submit herewith form PTO/SB/08B - Information Disclosure Statement together with any required copies of patents, publications or other information of which applicant(s) are aware, which applicant(s) believe(s) may be material to the examination of this application and for which there may be a duty to disclose in accordance with 37 CFR 1.56.

☐ A concise explanation of the relevance of foreign language patents, foreign language publications and other foreign language information listed on PTO/SB/08B, as presently understood by the individual(s) designated in 37 CFR 1.56 (c) most knowledgeable about the content is given on the attached sheet, or where a foreign language patent is cited in a search report or other action by a foreign patent office in a counterpart foreign application, an english language version of the search report or action which indicates the degree of relevance found by the foreign office is listed on form PTO/SB/08B and is enclosed.

It is requested that the information disclosed herein be made of record in this application.

Respectfully submitted,

Scott L. Michaelis

By: /Robert W. Nelson/

Robert W. Nelson, Esq.

Attorney/Agent for Applicant(s)

Reg No. : 37,898

Date : June 2, 2009

Telephone : (303) 298-9888

Electronic Acknowledgement Receipt

EFS ID:	5433682
Application Number:	10972588
International Application Number:	
Confirmation Number:	1079
Title of Invention:	SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR
First Named Inventor/Applicant Name:	Scott L. Michaelis
Customer Number:	22879
Filer:	Robert Wayne Nelson/Kalyn Black
Filer Authorized By:	Robert Wayne Nelson
Attorney Docket Number:	200404973-1
Receipt Date:	02-JUN-2009
Filing Date:	25-OCT-2004
Time Stamp:	10:53:12
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	NPL Documents	10972796-OA-7-12-07.pdf	913945 8df7246704e1c499c4c302fa1312639234bd43a5	no	23

Warnings:	
Information:	

2	NPL Documents	10972796-NOA-12-19-07.pdf	393944 4d88be2151c0a787fec8c9743e653bc3c0324694	no	8
Warnings:					
Information:					
3	NPL Documents	10973075-FOA-5-12-09.pdf	1124143 3676b984375a0e719f6b2807fb21484f9907959d	no	28
Warnings:					
Information:					
4	NPL Documents	10973075-OA-7-2-07.pdf	746284 40981eb90cc6df82f77664306b7a69d1d0e45463	no	21
Warnings:					
Information:					
5	NPL Documents	10973075-OA-10-29-08.pdf	1015037 0b73192903357d5fa1ca57fd41ce5f8645751f93	no	25
Warnings:					
Information:					
6	NPL Documents	10973075-OA-12-13-07.pdf	645056 8548bb285aa89be5b10bf0d092bde1dadaed7df5	no	18
Warnings:					
Information:					
7	NPL Documents	10973004-NOA-8-1-07.pdf	493543 a450c3aabdb833d9a4e1af5fb345b8484e5e0533	no	12
Warnings:					
Information:					
8	NPL Documents	10973004-OA-4-10-07.pdf	494591 d888de82b3cfb9bc0956ca5109672f36293ff2d6	no	13
Warnings:					
Information:					
9	Information Disclosure Statement (IDS) Filed (SB/08)	sb0008b-4.pdf	192604 0006a50fea5499d9ff93fc683d5f9f318d7256ad	no	1
Warnings:					
Information:					
This is not an USPTO supplied IDS fillable form					
10	Transmittal Letter	E-IDS.pdf	554045 9b6a4e0d48806b74fd66b3802a57aad1fd772c0a	no	1

Warnings:	
Information:	
Total Files Size (in bytes):	6573192

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PTO/SB/08b (03-09)
Approved for use through 04/30/2009. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 1

Complete if Known	
Application Number	10/972,588
Filing Date	10/25/2004
First Named Inventor	Scott L. Michaelis
Art Unit	2114
Examiner Name	Le, Dieu Minh T.
Attorney Docket Number	200404973-1

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		SHOICHI ARAKI, Real-Time Tracking of Multiple Moving Object Contours in a Moving Camera Image Sequence, Vol. E83-D IEICE Trans. Inf. & Syst, 7-7-2000, 9 pages	
		DONG KYU KIM, Active Object Tracking Using Image Mosaic Background, 4 pages	
		JIANG-BIN ZHENG, Proceedings of the First International Conference on Machine Learning and Cybernetics, Beijing, 4-5 November 2002, 4 pages	
		JONG RYUL KIM, Automatic Localization and Trackign of Moving Objects Using Adaptive Snake Algorithm, ICICS-PCM 2003, 15-18 December 2003, 5 pages, Singapore	
		JIANHOA MENG, CVEPS-A Compressed Video Editing and Parsing System, 18-11-1996, Pages 1-11	

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

1 Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

Electronic Acknowledgement Receipt

EFS ID:	5440472
Application Number:	10972588
International Application Number:	
Confirmation Number:	1079
Title of Invention:	SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR
First Named Inventor/Applicant Name:	Scott L. Michaelis
Customer Number:	22879
Filer:	Robert Wayne Nelson/Kalyn Black
Filer Authorized By:	Robert Wayne Nelson
Attorney Docket Number:	200404973-1
Receipt Date:	02-JUN-2009
Filing Date:	25-OCT-2004
Time Stamp:	18:12:03
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Transmittal Letter	E-IDS.pdf	554045 9b6a4e0d48806b74fd66b3802a57aad1fd772c0a	no	1

Warnings:

Information:

2	Information Disclosure Statement (IDS) Filed (SB/08)	sb0008b-5.pdf	192377 d88564659260dae04beb9b9affd14c064ce2c94f	no	1
Warnings:					
Information:					
This is not an USPTO supplied IDS fillable form					
3	NPL Documents	ARAKI.pdf	665693 d120600a6183dda5502856570047b71d90a74c43	no	9
Warnings:					
Information:					
4	NPL Documents	DONG.pdf	341377 019107a21f6652392db9189d58c224663a6408ee	no	4
Warnings:					
Information:					
5	NPL Documents	JIANG-BIN.pdf	262550 2d9e113e16e6d7dc2117f9a36022acabf7ecfb13	no	4
Warnings:					
Information:					
6	NPL Documents	JIANHAO.pdf	891903 bc575f9ce6e3e09aaab6f5f719493ec251968ad9	no	11
Warnings:					
Information:					
7	NPL Documents	JONG.pdf	377312 5ad526bbd2aaab1ace0b05eea8f8a862732c42f6	no	5
Warnings:					
Information:					
			Total Files Size (in bytes):	3285257	

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 200404973-1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): Scott L. Michaelis	**Confirmation No.:** 1079
Application No.: 10/972,588	**Examiner:** Le, Dieu Minh T.
Filing Date: 10/25/2004	**Group Art Unit:** 2114

Title: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

Mail Stop Amendment
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

INFORMATION DISCLOSURE STATEMENT

This Information Disclosure Statement is submitted:

☒ under 37 CFR 1.97 (b), or
(Within three months of filing national application; or date of entry of national application; or before mailing date of first action on the merits; whichever occurs last)

☐ under 37 CFR 1.97 (c), together with either a:
- ☐ Statement under 37 CFR 1.97 (e), or
- ☐ a $180 fee under 37 CFR 1.17 (p), or

(After the CFR 1.97 (b) time period, but before final action or notice of allowance, whichever occurs first)

☐ under 37 CFR 1.97 (d), together with a:
- ☐ Statement under 37 CFR 1.97 (e)(1) or (2), and
- ☐ a $180 fee set forth in 37 CFR 1.17 (p)

(Filed after final action, a notice of allowance, on or before payment of the issue fee)

Please charge to Deposit Account **08-2025** the sum of $ 0.00 . At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account **08-2025** pursuant to 37 CFR 1.25.

Applicant(s) submit herewith form PTO/SB/08B - Information Disclosure Statement together with any required copies of patents, publications or other information of which applicant(s) are aware, which applicant(s) believe(s) may be material to the examination of this application and for which there may be a duty to disclose in accordance with 37 CFR 1.56.

☐ A concise explanation of the relevance of foreign language patents, foreign language publications and other foreign language information listed on PTO/SB/08B, as presently understood by the individual(s) designated in 37 CFR 1.56 (c) most knowledgeable about the content is given on the attached sheet, or where a foreign language patent is cited in a search report or other action by a foreign patent office in a counterpart foreign application, an english language version of the search report or action which indicates the degree of relevance found by the foreign office is listed on form PTO/SB/08B and is enclosed.

It is requested that the information disclosed herein be made of record in this application.

Respectfully submitted,

Scott L. Michaelis

By: /Robert W. Nelson/

Robert W. Nelson, Esq.

Attorney/Agent for Applicant(s)

Reg No. : 37,898

Date : June 2, 2009

Telephone : (303) 298-9888

PTO/SB/08b (03-09)
Approved for use through 04/30/2009. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 1

Complete if Known	
Application Number	10/972,588
Filing Date	10/25/2004
First Named Inventor	Scott L. Michaelis
Art Unit	2114
Examiner Name	Le, Dieu Minh T.
Attorney Docket Number	200404973-1

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		Notice of Allowance for Serial No. 10/973,076 dated 11-28-2008 in the United States	
		Notice of Allowance for Serial No. 10/973,076 dated 7-16-2008 in the United States	
		Notice of Allowance for Serial No. 10/973,076 dated 6-3-2008 in the United States	
		Final Office Action for Serial No. 10/973,076 dated 2-27-2008 in the United States	
		Office Action for Serial No. 10/973,076 dated 9-12-2007 in the United States	
		Notice of Allowance for Serial No. 10/972,888 dated 11-14-2007 in the United States	
		Office Action for Serial No. 10/972,888 dated 7-18-2007 in the United States	

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

1 Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 200404973-1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): Scott L. Michaelis	**Confirmation No.:** 1079
Application No.: 10/972,588	**Examiner:** Le, Dieu Minh T.
Filing Date: 10/25/2004	**Group Art Unit:** 2114

Title: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

Mail Stop Amendment
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

INFORMATION DISCLOSURE STATEMENT

This Information Disclosure Statement is submitted:

☒ under 37 CFR 1.97 (b), or
(Within three months of filing national application; or date of entry of national application; or before mailing date of first action on the merits; whichever occurs last)

☐ under 37 CFR 1.97 (c), together with either a:
- ☐ Statement under 37 CFR 1.97 (e), or
- ☐ a $180 fee under 37 CFR 1.17 (p), or

(After the CFR 1.97 (b) time period, but before final action or notice of allowance, whichever occurs first)

☐ under 37 CFR 1.97 (d), together with a:
- ☐ Statement under 37 CFR 1.97 (e)(1) or (2), and
- ☐ a $180 fee set forth in 37 CFR 1.17 (p)

(Filed after final action, a notice of allowance, on or before payment of the issue fee)

Please charge to Deposit Account **08-2025** the sum of $ 0.00 . At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account **08-2025** pursuant to 37 CFR 1.25.

Applicant(s) submit herewith form PTO/SB/08B - Information Disclosure Statement together with any required copies of patents, publications or other information of which applicant(s) are aware, which applicant(s) believe(s) may be material to the examination of this application and for which there may be a duty to disclose in accordance with 37 CFR 1.56.

☐ A concise explanation of the relevance of foreign language patents, foreign language publications and other foreign language information listed on PTO/SB/08B, as presently understood by the individual(s) designated in 37 CFR 1.56 (c) most knowledgeable about the content is given on the attached sheet, or where a foreign language patent is cited in a search report or other action by a foreign patent office in a counterpart foreign application, an english language version of the search report or action which indicates the degree of relevance found by the foreign office is listed on form PTO/SB/08B and is enclosed.

It is requested that the information disclosed herein be made of record in this application.

Respectfully submitted,

Scott L. Michaelis

By: /Robert W. Nelson/

Robert W. Nelson, Esq.

Attorney/Agent for Applicant(s)

Reg No. : 37,898

Date : June 1, 2009

Telephone : (303) 298-9888

Electronic Acknowledgement Receipt

EFS ID:	5426049
Application Number:	10972588
International Application Number:	
Confirmation Number:	1079
Title of Invention:	SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR
First Named Inventor/Applicant Name:	Scott L. Michaelis
Customer Number:	22879
Filer:	Robert Wayne Nelson/Kalyn Black
Filer Authorized By:	Robert Wayne Nelson
Attorney Docket Number:	200404973-1
Receipt Date:	01-JUN-2009
Filing Date:	25-OCT-2004
Time Stamp:	00:58:50
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Information Disclosure Statement (IDS) Filed (SB/08)	sb0008b-3.pdf	192644 cdb98c1114a5b5ef308010b498319c071194ddda	no	1

Warnings:	
Information:	

This is not an USPTO supplied IDS fillable form					
2	NPL Documents	10972888-NOA-11-14-07.pdf	343428 cc4d2287cce312a336dac7e46dbadd1a2c690d41	no	6
Warnings:					
Information:					
3	NPL Documents	10972888-OA7-18-07.pdf	757427 b845a9cc8aea7e7f74137f0c06a78694d291612e	no	20
Warnings:					
Information:					
4	NPL Documents	10973076-FOA-2-27-08.pdf	673034 c3414454ddcb4fdb1f15c94aa016213af1b26d6e	no	18
Warnings:					
Information:					
5	NPL Documents	10973076-NOA-6-3-08.pdf	1093613 a2d6b0b19a5d6ef731e857b195cde22c00765de6	no	26
Warnings:					
Information:					
6	NPL Documents	10973076-NOA-7-16-08.pdf	1096462 41fde393cc85463f280d18dc39da81177468ccc0	no	20
Warnings:					
Information:					
7	NPL Documents	10973076-NOA-11-28-08.pdf	1538148 65a401cdd16f234b9a802e527eda15bd9059c198	no	29
Warnings:					
Information:					
8	NPL Documents	10973076-OA-9-12-07.pdf	945952 ab199af1fc9eb369e9dea4ece830e8edd1d1d982	no	30
Warnings:					
Information:					
9	Transmittal Letter	E-IDS.pdf	547906 3ec08121c21b039bfb234ce86d5970e25f5fd6ab	no	1
Warnings:					
Information:					
			Total Files Size (in bytes):	7188614	

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PTO/SB/08b (03-09)
Approved for use through 04/30/2009. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 1

Complete if Known	
Application Number	10/972,588
Filing Date	10/25/2004
First Named Inventor	Scott L. Michaelis
Art Unit	2114
Examiner Name	Le, Dieu Minh T.
Attorney Docket Number	200404973-1

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		Office Action for Serial No. 10/973,003 dated 3-5-2008 in the United States	
		Office Action for Serial No. 10/973,003 dated 9-12-2007 in the United States	

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

1 Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

Electronic Acknowledgement Receipt

EFS ID:	5417501
Application Number:	10972588
International Application Number:	
Confirmation Number:	1079
Title of Invention:	SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR
First Named Inventor/Applicant Name:	Scott L. Michaelis
Customer Number:	22879
Filer:	Robert Wayne Nelson/Kalyn Black
Filer Authorized By:	Robert Wayne Nelson
Attorney Docket Number:	200404973-1
Receipt Date:	29-MAY-2009
Filing Date:	25-OCT-2004
Time Stamp:	08:35:50
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	NPL Documents	10973003_3_5_2008.pdf	636651 ecc20c500681c00ec53c5f029cd3ade46f6638fe	no	20

Warnings:	
Information:	

2	NPL Documents	10973003_9_12_2007.pdf	826362 16f7bdd13700389b43b8b2587d228e24298bd3cb	no	26
Warnings:					
Information:					
3	Transmittal Letter	E-IDS.pdf	541767 3d1fb64ac458f92e537e0e5e437d8c4395a2f83a	no	1
Warnings:					
Information:					
4	Information Disclosure Statement (IDS) Filed (SB/08)	sb0008b-2.pdf	189925 43948f4f398409d7e52e76d70f3e8e4e2d61bb90	no	1
Warnings:					
Information:					
This is not an USPTO supplied IDS fillable form					
		Total Files Size (in bytes):	2194705		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

<u>New Applications Under 35 U.S.C. 111</u>
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

<u>National Stage of an International Application under 35 U.S.C. 371</u>
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

<u>New International Application Filed with the USPTO as a Receiving Office</u>
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 200404973-1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s):	**Scott L. Michaelis**	**Confirmation No.:**	**1079**
Application No.:	**10/972,588**	**Examiner:**	**Le, Dieu Minh T.**
Filing Date:	**10/25/2004**	**Group Art Unit:**	**2114**

Title: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

Mail Stop Amendment
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

INFORMATION DISCLOSURE STATEMENT

This Information Disclosure Statement is submitted:

☒ under 37 CFR 1.97 (b), or
(Within three months of filing national application; or date of entry of national application; or before mailing date of first action on the merits; whichever occurs last)

☐ under 37 CFR 1.97 (c), together with either a:
- ☐ Statement under 37 CFR 1.97 (e), or
- ☐ a $180 fee under 37 CFR 1.17 (p), or

(After the CFR 1.97 (b) time period, but before final action or notice of allowance, whichever occurs first)

☐ under 37 CFR 1.97 (d), together with a:
- ☐ Statement under 37 CFR 1.97 (e)(1) or (2), and
- ☐ a $180 fee set forth in 37 CFR 1.17 (p)

(Filed after final action, a notice of allowance, on or before payment of the issue fee)

Please charge to Deposit Account **08-2025** the sum of $ 0.00 . At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account **08-2025** pursuant to 37 CFR 1.25.

Applicant(s) submit herewith form PTO/SB/08B - Information Disclosure Statement together with any required copies of patents, publications or other information of which applicant(s) are aware, which applicant(s) believe(s) may be material to the examination of this application and for which there may be a duty to disclose in accordance with 37 CFR 1.56.

☐ A concise explanation of the relevance of foreign language patents, foreign language publications and other foreign language information listed on PTO/SB/08B, as presently understood by the individual(s) designated in 37 CFR 1.56 (c) most knowledgeable about the content is given on the attached sheet, or where a foreign language patent is cited in a search report or other action by a foreign patent office in a counterpart foreign application, an english language version of the search report or action which indicates the degree of relevance found by the foreign office is listed on form PTO/SB/08B and is enclosed.

It is requested that the information disclosed herein be made of record in this application.

Respectfully submitted,

Scott L. Michaelis

By: /Robert W. Nelson/

Robert W. Nelson, Esq.

Attorney/Agent for Applicant(s)

Reg No. : 37,898

Date : May 29, 2009

Telephone : (303) 298-9888

PTO/SB/08b (03-09)
Approved for use through 04/30/2009. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 1

Complete if Known	
Application Number	10/972,588
Filing Date	10/25/2004
First Named Inventor	Scott L. Michaelis
Art Unit	2114
Examiner Name	Le, Dieu Minh T.
Attorney Docket Number	200404973-1

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		Notice of Allowance for Serial No. 10/972,835 dated 11-20-2008 in the United States	
		Notice of Allowance for Serial No. 10/972,835 dated 8-1-2008 in the United States	
		Office Action for Serial No. 10/972,835 dated 3-3-2008 in the United States	
		Office Action for Serial No. 10/972,835 dated 7-12-2007 in the United States	

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

1 Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

Electronic Acknowledgement Receipt

EFS ID:	5409696
Application Number:	10972588
International Application Number:	
Confirmation Number:	1079
Title of Invention:	SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR
First Named Inventor/Applicant Name:	Scott L. Michaelis
Customer Number:	22879
Filer:	Robert Wayne Nelson/Kalyn Black
Filer Authorized By:	Robert Wayne Nelson
Attorney Docket Number:	200404973-1
Receipt Date:	28-MAY-2009
Filing Date:	25-OCT-2004
Time Stamp:	09:14:01
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Transmittal Letter	E-IDS.pdf	535632 9874f45496f4b448c8a1c021430f814d55c3d76a	no	1

Warnings:	
Information:	

2	Information Disclosure Statement (IDS) Filed (SB/08)	sb0008b-1.pdf	190653 8a0e906e89192794de7a0eefb6622842be0d7a01	no	1
Warnings:					
Information:					
This is not an USPTO supplied IDS fillable form					
3	NPL Documents	10972835-NOA-8-1-2008.pdf	1240979 6182eab2b3051aa346aa5b53eb6da542b1ccf84e	no	25
Warnings:					
Information:					
4	NPL Documents	10972835-NOA-11-20-2008.pdf	930084 9b79b0497dbeeb0a4fa735fd4d88465afa5018cc	no	15
Warnings:					
Information:					
5	NPL Documents	10972835-OA-3-3-2008.pdf	943392 54f2d55316e1813f658637ccb44c210fc043663e	no	21
Warnings:					
Information:					
6	NPL Documents	10972835-OA-7-12-2007.pdf	738688 6859d5e094895028427b8cd924c6c7548a58fcd2	no	19
Warnings:					
Information:					
		Total Files Size (in bytes):	4579428		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 200404973-1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): Scott L. Michaelis	Confirmation No.: 1079
Application No.: 10/972,588	Examiner: Le, Dieu Minh T.
Filing Date: 10/25/2004	Group Art Unit: 2114

Title: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

Mail Stop Amendment
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

INFORMATION DISCLOSURE STATEMENT

This Information Disclosure Statement is submitted:

☒ under 37 CFR 1.97 (b), or
(Within three months of filing national application; or date of entry of national application; or before mailing date of first action on the merits; whichever occurs last)

☐ under 37 CFR 1.97 (c), together with either a:
- ☐ Statement under 37 CFR 1.97 (e), or
- ☐ a $180 fee under 37 CFR 1.17 (p), or

(After the CFR 1.97 (b) time period, but before final action or notice of allowance, whichever occurs first)

☐ under 37 CFR 1.97 (d), together with a:
- ☐ Statement under 37 CFR 1.97 (e)(1) or (2), and
- ☐ a $180 fee set forth in 37 CFR 1.17 (p)

(Filed after final action, a notice of allowance, on or before payment of the issue fee)

Please charge to Deposit Account **08-2025** the sum of $ 0.00 . At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account **08-2025** pursuant to 37 CFR 1.25.

Applicant(s) submit herewith form PTO/SB/08B - Information Disclosure Statement together with any required copies of patents, publications or other information of which applicant(s) are aware, which applicant(s) believe(s) may be material to the examination of this application and for which there may be a duty to disclose in accordance with 37 CFR 1.56.

☐ A concise explanation of the relevance of foreign language patents, foreign language publications and other foreign language information listed on PTO/SB/08B, as presently understood by the individual(s) designated in 37 CFR 1.56 (c) most knowledgeable about the content is given on the attached sheet, or where a foreign language patent is cited in a search report or other action by a foreign patent office in a counterpart foreign application, an english language version of the search report or action which indicates the degree of relevance found by the foreign office is listed on form PTO/SB/08B and is enclosed.

It is requested that the information disclosed herein be made of record in this application.

Respectfully submitted,

Scott L. Michaelis

By: /Robert W. Nelson/

Robert W. Nelson, Esq.

Attorney/Agent for Applicant(s)

Reg No. : 37,898

Date : May 28, 2009

Telephone : (303) 298-9888

PTO/SB/08a (01-09)
Approved for use through 02/28/2009. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 1

Complete if Known	
Application Number	10/972,588
Filing Date	10/25/2004
First Named Inventor	Scott L. Michaelis
Art Unit	2114
Examiner Name	Le, Dieu Minh T.
Attorney Docket Number	200404973-1

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number Number-Kind Code[2] *(if known)*	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US- 2004-0078651	02-21-2006	Kevin David Safford	
		US- 2006-0248384	11-02-2006	Kevin David Safford	
		US- 6,330,356	12-11-2001	SUNDARESWARAN	
		US- 6,141,770	10-31-2000	STEPHEN FUCHS	
		US- 6,263,452	07-17-2001	DOUGLAS E. JEWETT	
		US- 6,643,387	11-04-2003	Sriram Sethuraman	
		US- 6,735,253	05-11-2004	SHIH-FU CHANG	
		US- 6,983,018	01-03-2006	CHIH-LUNG	
		US- 7,042,943	05-09-2006	Barin Geoffry Haskell	
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3]-Number[4]-Kind Code[5] *(if known)*	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages Or Relevant Figures Appear	T[6]
		WO 01 45421	06-21-2001	Koninklljke		

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional). [2] See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3] Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4] For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5] Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [6] Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

(12) NACH DEM VERTRAG ÜBER DIE INTERNATIONALE ZUSAMMENARBEIT AUF DEM GEBIET DES PATENTWESENS (PCT) VERÖFFENTLICHTE INTERNATIONALE ANMELDUNG

(19) Weltorganisation für geistiges Eigentum
Internationales Büro

(43) Internationales Veröffentlichungsdatum
21. Juni 2001 (21.06.2001)

PCT

(10) Internationale Veröffentlichungsnummer
WO 01/45421 A1

(51) Internationale Patentklassifikation⁷: H04N 7/26

(21) Internationales Aktenzeichen: PCT/EP00/11653

(22) Internationales Anmeldedatum:
23. November 2000 (23.11.2000)

(25) Einreichungssprache: Deutsch

(26) Veröffentlichungssprache: Deutsch

(30) Angaben zur Priorität:
99204369.5 17. Dezember 1999 (17.12.1999) EP

(71) Anmelder *(für alle Bestimmungsstaaten mit Ausnahme von US)*: KONINKLIJKE PHILIPS ELECTRONICS N.V. [NL/NL]; Groenewoudseweg 1, NL-5621 BA Eindhoven (NL).

(72) Erfinder; und

(75) Erfinder/Anmelder *(nur für US)*: DE HAAN, Gerard [NL/NL]; Prof. Holstlaan 6, NL-5656 AA Eindhoven (NL). WITTEBROOD, Rimmert, B. [NL/NL]; Prof. Holstlaan 6, NL-5656 AA Eindhoven (NL). SCHUTTEN, Robert, J. [NL/NL]; Prof. Holstlaan 6, NL-5656 AA Eindhoven (NL).

(74) Anwalt: STEENBEEK, Leonardus, J.; Internationaal Octrooibureau B.V., Prof Holstlaan 6, NL-5656 AA Eindhoven (NL).

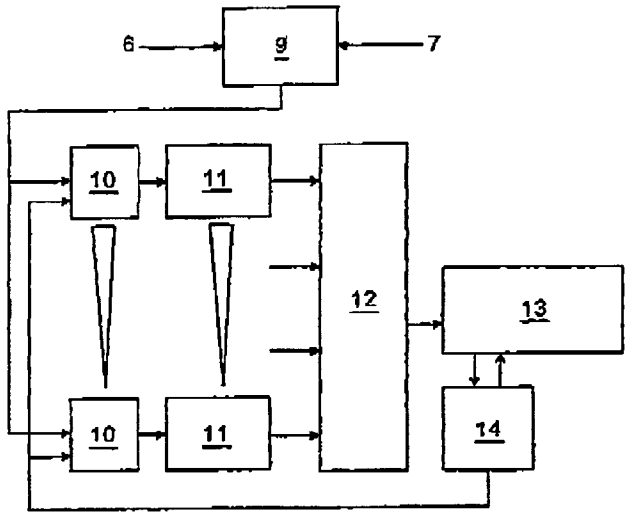
(81) Bestimmungsstaaten *(national)*: JP, KR, US.

(84) Bestimmungsstaaten *(regional)*: europäisches Patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR).

[Fortsetzung auf der nächsten Seite]

(54) Title: MOTION ESTIMATION IN VIDEO IMAGES

(54) Bezeichnung: BEWEGUNGSSCHÄTZUNG IN VIDEOBILDERN



(57) Abstract: The invention relates to an object-oriented motion estimation method for the interpolation between video image data sets. According to said method, two or more motion models are used to describe the displacement of image objects between a first and a second video image. The motion models are determined by sets of parameters from which displacement vectors for the image objects can be calculated. One of the motion models is required for taking into account the parts of the image that are static, i.e., stationary. The consistency of the allocation of the moveable image objects to the corresponding image parts in terms of time and space is taken into account when the image area is segmented.

[Fortsetzung auf der nächsten Seite]

WO 01/45421 A1

PAGE 42/62 * RCVD AT 6/21/2006 11:39:02 AM [Eastern Daylight Time] * SVR:USPTO-EFXRF-5/8 * DNIS:2738300 * CSID:6508575487 * DURATION (mm-ss):18-08

Veröffentlicht:
- *Mit internationalem Recherchenbericht.*
- *Vor Ablauf der für Änderungen der Ansprüche geltenden Frist; Veröffentlichung wird wiederholt, falls Änderungen eintreffen.*

Zur Erklärung der Zweibuchstaben-Codes, und der anderen Abkürzungen wird auf die Erklärungen ("Guidance Notes on Codes and Abbreviations") am Anfang jeder regulären Ausgabe der PCT-Gazette verwiesen.

(57) Zusammenfassung: Die Erfindung betrifft ein objekt-orientiertes Bewegungsschätzungsverfahren für die Interpolation zwischen Videobilddatensätzen. Es werden dabei zwei oder mehr Bewegungsmodelle verwendet, um die Verschiebung von Bildobjekten zwischen einem ersten und einem zweiten Videobild zu beschreiben. Die Bewegungsmodelle werden durch Parametersätze bestimmt, aus welchen sich Verschiebungsvektoren für die Bildobjekte berechnen lassen. Eines der Bewegungsmodelle wird benötigt, um diejenigen Bildteile zu berücksichtigen, die statisch, also unbeweglich sind. Bei der Segmentierung des Bildbereiches wird die zeitliche und räumliche Konsistenz der Zuordnung von beweglichen Bildobjekten zu den entsprechenden Bildteilen beachtet.

PAGE 43/62 * RCVD AT 6/21/2006 11:39:02 AM [Eastern Daylight Time] * SVR:USPTO-EFXRF-5/8 * DNIS:2738300 * CSID:6508575487 * DURATION (mm-ss):18-08

Bewegungsschätzung in Videobildern

Die Erfindung betrifft ein Verfahren zur Bewegungsschätzung in Videobilddaten, wobei zunächst ausgehend von einem ersten und einem zweiten Videobild Parametersätze von zwei oder mehr Bewegungsmodellen ermittelt werden und wobei dann den Bewegungsmodellen Bildobjekte zugeordnet werden. Die Erfindung betrifft ferner eine Vorrichtung zur Durchführung des Verfahrens, eine gemäß dem Verfahren arbeitende Vorrichtung zur Darstellung von Videobildern und ein Computerprogrammprodukt zur Bewegungsschätzung.

Fortschritte in der Multimedia-Technik haben zur Entwicklung einer Vielzahl von Videoformaten und Wiedergabestandards geführt. Diese unterscheiden sich unter anderem auch in der Bildrate, also in der Zahl der Einzelbilder pro Zeiteinheit. Soll eine Videosequenz auf einem PC- oder TV-Bildschirm wiedergegeben werden, so ist es erforderlich, eine Anpassung an die Bildrate des Wiedergabegeräts vorzunehmen. Hierfür geeignete Schnittstellen arbeiten mit Konversionsverfahren unterschiedlicher Komplexität. Die einfachste Methode besteht darin, bei der Wiedergabe je nach gewünschter Bildrate Einzelbilder der Videosequenz zu wiederholen oder auszulassen. Bei der Darstellung derartig behandelter Videodaten treten jedoch unerwünschte Artefakte auf. Es kommt je nach dem Verhältnis der zugrundeliegenden Bildraten zu unerwünschten Darstellungsfehlern. Die Wiedergabe erscheint stockend und ungleichmäßig, so daß die in der Videosequenz dargestellten Bewegungen unnatürlich wirken. Aufwendigere Verfahren nehmen eine Interpolation zwischen aufeinanderfolgenden Videobildern vor, wobei ein Algorithmus zur Bewegungsschätzung zum Einsatz kommt, welcher die Verschiebungen einzelner Bildelemente von einem Einzelbild zum nächsten erkennt und daraus Bilddaten erzeugt, die zeitlich zwischen den Einzelbildern der Videosequenz liegen. Die Anwendung derartiger Verfahren im Bereich der Heim-Endgeräte erfordert, daß die zugrundeliegenden Algorithmen eine qualitativ hochwertige Bildratenkonversion liefern und dabei nur einen geringen Rechenaufwand benötigen, da der Leistungsfähigkeit der digitalen Signalverarbeitungselektronik von Heimgeräten Grenzen gesetzt sind.

Verfahren zur Bewegungsschätzung der eingangs genannten Art eignen sich außer für die Bildratenkonversion auch für die Kodierung und Kompression bei der Übertragung von Videodaten sowie für die Tiefeschätzung in der 3D Bilddatenverarbeitung und für Disparitätsschätzungen bei Stereobildern.

Ein derartiges Verfahren schlägt die WO 99/16251 vor. Es handelt sich hierbei um ein leistungsfähiges, objektorientiertes Verfahren zur Bewegungsschätzung, bei dem zwei oder mehr Bewegungsmodelle verwendet werden, um die Verschiebung von Bildobjekten zwischen einem aktuellen und einem vorhergehenden Videobild zu beschreiben. Die Bewegungsmodelle werden durch Parametersätze bestimmt, aus welchen sich Verschiebungsvektoren berechnen lassen. Eines der Bewegungsmodelle wird benötigt, um diejenigen Bildteile zu berücksichtigen, die statisch sind. Der zugehörige Verschiebungsvektor ist also der Nullvektor. Die Parametersätze der übrigen Bewegungsmodelle werden ermittelt, indem die Anpassungsfehler der Bewegungsmodelle bei der Beschreibung der Verschiebung der Bildobjekte zwischen aufeinanderfolgenden Videobildern bewertet werden. Für die Interpolation ist es dann erforderlich, die Bilddaten zu segmentieren und dabei einzelnen Bildobjekten geeignete Bewegungsmodelle zuzuordnen. Das Ergebnis der Segmentierung sind separate Objekte, also Bildteile die vom vorhergehenden zum aktuellen Videobild eine ähnliche oder vergleichbare Verschiebung ausführen.

Das vorbekannte Verfahren zur Bewegungsschätzung stellt eine leistungsfähige Alternative zu den ansonsten üblichen blockorientierten Verfahren dar, da die Zahl der unabhängig voneinander beweglichen Objekte in normalen Videosequenzen gering ist und deswegen auch nur eine entsprechend kleine Zahl von Bewegungsmodellen verarbeitet werden muß. Daraus resultiert eine geringe Zahl von Rechenoperationen, was das Verfahren auch im Heimbereich universell einsetzbar macht.

Die grundlegende Aufgabe der vorliegenden Erfindung ist es, das vorbekannte Bewegungsschätzungsverfahren weiter zu verbessern und dabei gleichzeitig die Komplexität weiter zu reduzieren.

Bei dem Verfahren zur Bewegungsschätzung ist die Segmentierung des Videobildes der schwierigste Schritt. Ziel ist es, jeder Gruppe von Bildpunkten eindeutig ein Bewegungsmodell zuzuordnen. Bei dem vorbekannten Verfahren wird dies dadurch erreicht, daß jedem der typischerweise acht mal acht Bildpunkte großen Blöcke dasjenige

Bewegungsmodell zugeordnet wird, das die Verschiebung im jeweiligen Bildbereich am besten beschreibt. Hierfür wird ein Anpassungsfehler bestimmt, welcher als Summe der absoluten Differenzen der Intensitäten von gemäß dem entsprechenden Bewegungsmodell relativ zueinander verschobenen Bildpunkten berechnet wird. Jedem Block wird dann dasjenige Bewegungsmodell zugeordnet, bei dem der Anpassungsfehler am kleinsten ist.

In der Praxis hat es sich gezeigt, daß es bei dieser Art der Segmentierung, insbesondere wenn eine große Zahl von individuell beweglichen Objekten in der Videosequenz dargestellt werden, zu Zuordnungsfehlern und unerwünschten Bildartefakten kommt.

Gemäß der zugrundeliegenden Aufgabenstellung wird die Qualität der Segmentierung bei einem Verfahren der eingangs genannten Art dadurch verbessert, daß die Bildobjekte aus einer Mehrzahl von Blöcken zusammengesetzt werden, in die der gesamte Bildbereich unterteilt wird, und daß für die Zuordnung der Bewegungsmodelle zu den Blöcken die Übereinstimmung der Bewegungsmodelle mit der Verschiebung innerhalb der jeweiligen Blöcke sowie die räumliche und/oder die zeitliche Konsistenz der Zuordnung bewertet werden.

Erfindungsgemäß werden die Blöcke nicht unbedingt denjenigen Bewegungsmodellen zugeordnet, für die sich der geringste Anpassungsfehler ergibt. Zusätzlich wird die räumliche Konsistenz der Zuordnung bewertet, indem die Bewegung in räumlich benachbarten Bildbereichen berücksichtigt wird. Typische Videosequenzen bestehen aus einer großen Zahl von Einzelbildern. Das Verfahren zur Bewegungsschätzung wird dabei auf Paare von jeweils aufeinanderfolgenden Bildern angewandt. Gemäß der Erfindung werden die einzelnen Bildpaare nicht mehr unabhängig voneinander behandelt, sondern es wird bei der Zuordnung der Bewegungsmodelle zu den Blöcken bewertet, welche Zuordnung bei früheren Bildpaaren der Videosequenz getroffen wurde. Auf diese Weise wird eine verbesserte räumliche und zeitliche Konsistenz bei der Segmentierung erreicht. Insgesamt ergibt sich somit ein rekursives Segmentierungsverfahren, mit welchem eine Interpolation mit deutlich verbesserter Bildqualität ermöglicht wird. Bei wiederholter Anwendung des Verfahrens auf eine Videosequenz ist eine rasche Konvergenz bei der Bestimmung der einzelnen Bildobjekte festzustellen.

Blöcken im Sinne der vorliegenden Erfindung sind, wie oben erwähnt, typischerweise rechteckige oder quadratische Anordnungen von mehreren Bildpunkten. Im Prinzip können die Blöcke aber auch aus jeweils nur einem einzelnen Bildpunkt bestehen.

Die Zahl der Rechenoperationen hängt bei den Verfahren zur Bewegungsschätzung gemäß der Erfindung stark von der Zahl der Parameter, durch welche

die Bewegungsmodelle bestimmt werden, ab. Ist die Zahl der Parameter sehr gering, sind die Bewegungsmodelle also entsprechend einfach, können unter Umständen komplexere Bewegungen nicht beschrieben werden, ohne die Zahl der Bewegungsmodelle zu erhöhen. Dadurch wird jedoch die Zahl der Rechenoperationen wiederum erhöht und außerdem müssen einzelne Bildobjekte in separate Unterobjekte mit jeweils unterschiedlichen Bewegungsmodellen aufgeteilt werden. Dies führt leicht zu Inkonsistenzen und unerwünschten Bildartefakten. Vorteilhafterweise sind die Verfahren zur Bewegungsschätzung gemäß der Erfindung dadurch zu verbessern, daß die Parametersätze der zwei oder mehr Bewegungsmodelle aus jeweils zwei bis vier Parametern bestehen und der Parametersatz eines zusätzlichen Bewegungsmodells aus sechs oder mehr Parametern besteht.

Die Bestimmung von zwei bis vier Parametern ist mit nur geringem Rechenaufwand verbunden. Durch derartig einfache Bewegungsmodelle lassen sich grundlegende Bewegungstypen wie Schwenks, Drehungen und Zoom problemlos beschreiben. Falls im Bildbereich komplexere Bewegungen auftreten, können diese durch Hinzufügen nur eines zusätzlichen Bewegungsmodells mit sechs oder mehr Parametern beschrieben werden, wobei der benötigte Rechenaufwand nur geringfügig ansteigt.

Es ist dabei zweckmäßig, den Parametersatz des zusätzlichen Bewegungsmodelles bei der Zuordnung der Bewegungsmodelle zu den Bildobjekten zu bestimmen.

Wie oben beschrieben, können die Parametersätze der Bewegungsmodelle durch Minimierung eines Anpassungsfehlers, der in ausgewählten Bildbereichen berechnet wird, bestimmt werden. Hochdimensionale Minimierungsverfahren, wie Sie für die Bestimmung von sechs oder mehr Parameter benötigt würden, sind jedoch extrem rechenzeitintensiv. Wesentlich effektiver ist es, die sechs oder mehr Parameter des zusätzlichen Bewegungsmodells erst bei der Segmentierung des Bildbereiches zu ermitteln. In diesem Fall ist das zusätzliche Bewegungsmodell als globales Bewegungsmodell zu verstehen, welches die übrigen Bewegungsmodelle im gesamten Bildbereich ergänzt und dabei lokale Abweichungen der Bewegungsmodelle von der wahren Bewegung der Bildobjekte ausgleicht.

Um Rechenzeit zu sparen, ist es weiter vorteilhaft, den Parametersatz des zusätzlichen Bewegungsmodell ausgehend von den Parametern bei früheren Videobilddaten zur verfeinern. Auf diese Weise wird die Bestimmung der Parameter des komplexen Bewegungsmodells rekursiv verbessert. Experimente haben gezeigt, daß durch das zusätzliche Bewegungsmodell eine deutliche Verbesserung bei der Qualität der Interpolation

erreicht wird, wobei aufgrund der raschen Konvergenz bei der rekursiven Parameterbestimmung der Rechenaufwand nur geringfügig zunimmt.

Für die Durchführung des erfindungsgemäßen Verfahrens eignet sich eine Vorrichtung zur Bewegungsschätzung in Videobilddaten, mit einem digitalen Bildspeicher, in welchem eine aktuelles und ein vorhergehendes Videobild speicherbar sind, mit Mitteln zur Bestimmung von Parametersätzen von zwei oder mehr Bewegungsmodellen und mit Mitteln zur Zuordnung der Bewegungsmodellen zu Bildobjekten, wobei die Vorrichtung Mittel zur Unterteilung des gesamten Bildbereiches in Blöcke, aus denen die Bildobjekte zusammengesetzt werden und zur Zuordnung der Blöcke zu den Bewegungsmodellen aufweist und wobei die Übereinstimmung der Bewegungsmodelle mit der Verschiebung innerhalb der jeweiligen Blöcke sowie die räumliche und zeitliche Konsistenz der Zuordnung bewertet werden. Derartige Vorrichtungen können vorteilhafterweise in Fernsehgeräte, Monitore, Video- oder DVD-Abspielgeräte usw. eingebaut werden. Der digitale Bildspeicher der erfindungsgemäßen Vorrichtung muß dabei nicht notwendigerweise eine ausreichende Kapazität haben, um das erste und das zweite Videobild gleichzeitig aufzunehmen. Die Speicherung der jeweiligen Bilder nacheinander reicht für das erfindungsgemäße Verfahren aus.

Vorrichtungen zur Darstellung von Videobildern, wie beispielsweise Fernsehgeräte, Monitore usw., mit einem digitalen Bildspeicher, in dem Videobilddaten speicherbar sind, mit elektronischen Mitteln zur Verarbeitung der in dem Bildspeicher gespeicherten Bilddaten und zur Darstellung von Videobildern auf einer Anzeigevorrichtung, wobei die Mittel zur Verarbeitung der Bilddaten Mittel zur Bestimmung von Parametersätzen von zwei oder mehr Bewegungsmodellen und Mittel zur Zuordnung der Bewegungsmodelle zu Bildobjekten aufweisen, können vorteilhafterweise von dem erfindungsgemäßen Verfahren profitieren, wenn die Mittel zur Verarbeitung der Bilddaten ferner Mittel zur Unterteilung des gesamten Bildbereiches in Blöcke, aus denen die Bildobjekte zusammengesetzt werden, und zur Zuordnung der Blöcke zu den Bewegungsmodellen aufweisen, wobei die Übereinstimmung der Bewegungsmodelle mit der Verschiebung innerhalb der jeweiligen Blöcke sowie die räumliche und zeitliche Konsistenz der Zuordnung bewertet werden. Herkömmliche digital arbeitende Fernsehgeräte und Monitore können auf einfache Weise nach dem erfindungsgemäßen Verfahren betrieben werden, wodurch die Qualität der Bilddarstellung verbessert wird. Anzeigevorrichtungen im obigen Sinne sind dabei zum Beispiel die bei Fernsehgeräten und Monitoren üblichen Kathodenstrahlröhren oder

Punktmatrixdisplays. Andere Anzeigevorrichtungen zur visuellen Ausgabe von digitalen Bilddaten sind genauso denkbar.

Ein entsprechendes Computerprogrammprodukt zur Bewegungsschätzung in Videobilddaten, welches als Eingabe ein aktuelles und ein vorhergehendes Videobild erhält und davon ausgehend zunächst Parametersätze von zwei oder mehr Bewegungsmodellen berechnet, dann den Bewegungsmodellen Bildobjekte zuordnet und schließlich Bewegungsdaten ausgibt, welche die Verschiebung der Bildobjekte vom vorhergehenden zum aktuellen Bild beschreiben, unterteilt den gesamten Bildbereich in Blöcke, aus denen die Bildobjekte zusammengesetzt werden, wobei es für die Zuordnung der Blöcke zu den Bewegungsmodellen die Übereinstimmung der Bewegungsmodelle mit der Verschiebung innerhalb der jeweiligen Blöcke sowie die räumliche und zeitlich Konsistenz der Zuordnung bewertet. Ein derartiges Computerprogrammprodukt kann beispielsweise eine Software umfassen, welche zur Programmierung von Videogeräten, Fernsehern usw. geeignet ist. Herkömmliche digital arbeitende Geräte können vorteilhafterweise mittels dieser Software erfindungsgemäß verbessert werden. Die Software kann dabei auf Datenträgern, wie Diskette oder CD-ROM vorliegen, oder kann über ein Datennetz (Internet) zum Herunterladen bereitgestellt werden.

Im folgenden werden, auch anhand der Figuren, Ausführungsbeispiele der Erfindung erläutert. Es zeigen:

Fig. 1 Blockdiagramm eines Bewegungsschätzungsverfahrens gemäß der Erfindung;

Fig. 2 Blockdiagramm einer erfindungsgemäßen Vorrichtung zur Darstellung von Videobildern.

Bei der Ermittlung der Parametersätze für die Bewegungsmodelle gemäß der Erfindung wird ein Auswahlkriterium auf ausgewählte Bildbereiche angewandt. Das Auswahlkriterium besteht dabei zum Beispiel in der Bewertung eines Anpassungsfehlers ε. Dieser wird als Summe von absoluten Differenzen von einzelnen bewegungskompensierten Pixelintensitäten zwischen aktuellem und vorhergehendem Videobild folgendermaßen berechnet:

$$\varepsilon(\bar{C}_o,n)=\sum_{\bar{x}\in I(n)} W_o(\bar{x})\cdot|F_s(\bar{x},n)-F_s(\bar{x}-\bar{C}_o(\bar{x},n),n-1)|$$

Es erfolgt eine Summierung über Bildkoordinaten $\bar{x} = \begin{pmatrix} x \\ y \end{pmatrix}$, die in einer Menge $I(n)$ von ausgewählten Bildbereichen enthalten sind. An diesen Bildkoordinaten werden die absoluten Differenzen zwischen den Pixelintensitäten im aktuellen bzw. vorhergehenden Videobild aufsummiert. $F_t(\bar{x},n)$ ist dabei die Pixelintensität an der Bildkoordinate $\bar{x}$ in einem Videobild mit reduziertem Bildraster. Es hat sich nämlich gezeigt, daß bei der Parameterbestimmung die Verwendung eines in der Auflösung reduzierten Bildes (sub-sampled image) völlig ausreichend ist. Dies führt vorteilhafterweise zu einer erheblichen Reduktion des Rechenaufwandes. Durch den fortlaufenden Index n wird die Nummer des Einzelbildes und damit der Zeitpunkt innerhalb der Videosequenz bezeichnet. $\bar{C}_o(\bar{x},n)$ bezeichnet für das Bildpaar n als aktuelles und $n-1$ als vorhergehendes Videobild den Verschiebungsvektor an der Bildkoordinate $\bar{x}$ gemäß dem Bewegungsmodell mit dem Index o. $W_o(\bar{x})$ stellt einen Wichtungsfaktor dar, mit dem berücksichtigt wird, welchem Bewegungsmodell o die Bildkoordinate $\bar{x}$ bei früheren Bilddaten der Videosequenz zugeordnet wurde. Damit läßt sich eine Verknüpfung zwischen Parameterbestimmung und Bildsegmentierung herstellen, was Vorteile im Hinblick auf die zeitliche Konsistenz der Bewegungsschätzung und die Leistungsfähigkeit des Verfahrens hat.

Mittels des folgenden Bewegungsmodells lassen sich ausgehend von vier Parametern Verschiebungsvektoren berechnen:

$$\bar{C}_o(\bar{x},n) = \begin{pmatrix} s_x(o,n) + x \cdot d_x(o,n) \\ s_y(o,n) + y \cdot d_y(o,n) \end{pmatrix}$$

Es handelt sich hierbei um ein einfaches lineares Modell erster Ordnung, mit dem Translationen und Skalierungen beschreibbar sind. Das Modell wird bestimmt durch den Parametersatz $\vec{P}_o(n) = \left(s_x(o,n), d_x(o,n), s_y(o,n), d_y(o,n)\right)^T$. Der Parametersatz wird so bestimmt, daß der obige Anpassungsfehler für das entsprechende Bewegungsmodell o einen minimalen Wert annimmt. Bei dem Bewegungsschätzungsverfahren gemäß der Erfindung kommen stets wenigstens zwei Bewegungsmodelle zum Einsatz, wovon eines immer den Nullvektor als Parametersatz hat, so daß durch dieses Bewegungsmodell mit dem Verschiebungsvektor $\bar{C}_0(\bar{x},n) = \bar{0}$ die stationären Bildbereiche beschrieben werden.

Der nächste Schritt bei der Bewegungsschätzung gemäß der Erfindung ist die Bildsegmentierung, also die Zuordnung von Bildbereichen zu den Bewegungsmodellen. Hierzu wird zunächst der gesamte Bildbereich in Blöcke unterteilt. In der Praxis haben sich

quadratische Blöcke aus 8 mal 8 Pixeln bewährt. Für alle Bildkoordinaten innerhalb des Blocks an der Stelle $\vec{X}$ gilt dann $\vec{x} \in B(\vec{X})$. Für jeden Block wird wiederum ein Anpassungsfehler an ein Bewegungsmodell o berechnet:

$$\varepsilon_o(\vec{X},n) = \sum_{\vec{x}\in B(\vec{X})} | F_s(\vec{x}+(1-\alpha)\vec{C}_o(\vec{x},n),n) - F_s(\vec{x}-\alpha\vec{C}_o(\vec{x},n),n-1) |$$

Der Zeitpunkt, zudem die Segmentierung gültig sein soll, wird durch α festgelegt. Im einfachsten Fall wird dem Block $\vec{X}$ dasjenige Bewegungsmodell o zugeordnet, für das $\varepsilon_o(\vec{X},n)$ minimal ist. Die Zuordnung wird dann in der Segmentierungsmaske $M(\vec{X},n)$ abgelegt.

Für die erfindungsgemäße Berücksichtigung der räumlichen und zeitlichen Konsistenz der Bildsegmentierung, also der Zuordnung von Bildbereichen zu Bewegungsmodellen, wird dem Anpassungfehler $\varepsilon_o(\vec{X},n)$ ein geeigneter Straf-Faktor $P(\vec{X},n)$ hinzugefügt:

$$\varepsilon_o(\vec{X},n) = \sum_{\vec{x}\in B(\vec{X})} | F_s(\vec{x}+(1-\alpha)\vec{C}_o(\vec{x},n),n) - F_s(\vec{x}-\alpha\vec{C}_o(\vec{x},n),n-1) | + P(\vec{X},n)$$

Der Straf-Faktor nimmt nach der folgenden Regel verschiedene Werte an:

$$P(\vec{X},n) = \begin{cases} P_s, \text{für } M(\vec{X}+\vec{\delta},n) = o \\ P_t, \text{für } M(\vec{X}-\vec{\delta},n-1) = o \\ P_u \quad \text{sonst} \end{cases} \quad \text{mit} \quad \vec{\delta} = \begin{pmatrix} i \\ j \end{pmatrix},\ i,j = 0,\pm 1$$

Durch relativ kleine Strafwerte P_s und P_t werden diejenigen Zuordnungen bevorzugt, bei denen zeitlich und räumlich benachbarte Blöcke dem gleichen Bewegungsmodell o zugeordnet sind, während alle übrigen Zuordnungen durch einen vergleichsweise großen Wert P_u benachteiligt werden.

Die Figur 1 zeigt schematisch den Ablauf eines Bewegungsschätzungsverfahrens gemäß der Erfindung. Ausgehend von einem vorhergehenden Videobild 6 und einem aktuellen Videobild 7 werden in einem ersten Schritt 9 für die Bestimmung der Parametersätze Bildbereiche ausgewählt. Für eine Mehrzahl von Bewegungsmodellen werden alle diese Bildbereiche mit Wichtungsfaktoren 10 versehen und dann in einem Verfahrensschritt 11, bei dem die Parameter der Bewegungsmodelle nach einem Auswahlkriterium ermittelt werden, weiterverarbeitet. Ausgehend von den vollständig

bestimmten Bewegungsmodellen wird dann in einem Schritt 12 der gesamte Bildbereich in Blöcke unterteilt und für jeden Block werden die den einzelnen Bewegungsmodellen entsprechenden Verschiebungsvektoren berechnet. Danach erfolgt in 13 die Segmentierung des Bildbereiches, wobei die Zuordnung der Blöcke zu den Bewegungsmodellen vorgenommen wird. In einer dabei entstehenden Segmentierungsmaske 14 werden die Zuordnungen, die beim nächsten Bildpaar in die Wichtung 10 Eingang finden, gespeichert. Auf die gespeicherte Segmentierungsmaske 14 wird dann während der Segmentierung zur Berücksichtigung der zeitlichen und räumlichen Konsistenz zugegriffen.

Die Figur 2 zeigt schematisch den Aufbau eines digital arbeitenden Anzeigegerätes, bei dem es sich z.B. um ein Fernsehgerät oder einen Videomonitor handeln kann. Der Vorrichtung wird ein Videosignal 20 zugeführt, das in einer digitalen Bildverarbeitungseinheit 21 gespeichert und aufbereitet wird. Hierzu weist die Bildverarbeitungseinheit einen Bildspeicher 22, einen Prozessor 23 und einen Programmspeicher 24 auf. Gegebenenfalls können diese Elemente auch, zumindest teilweise, in einem diskreten Bauteil vereinigt sein. Auf dem Prozessor 23 läuft ein in dem Programmspeicher 24 abgelegtes Programm ab, durch das die erfindungsgemäße Bildverarbeitung gesteuert wird. Eine Bilddarstellungseinheit 25 erhält von der Bildverarbeitungseinheit 21 aufbereitete Bilddaten 26 und erzeugt daraus ein Signal 27 zur Ansteuerung einer Kathodenstrahlröhre 28, über welche die Videobilder visuell ausgegeben werden.

PATENTANSPRÜCHE:

1. Verfahren zur Bewegungsschätzung in Videobilddaten, wobei zunächst ausgehend von einem ersten und einem zweiten Videobild (6,7) Parametersätze von zwei oder mehr Bewegungsmodellen ermittelt werden (11) und wobei dann den Bewegungsmodellen Bildobjekte zugeordnet werden (13),
dadurch gekennzeichnet, daß
die Bildobjekte aus einer Mehrzahl von Blöcken zusammengesetzt werden, in die der gesamte Bildbereich unterteilt wird (12), und daß für die Zuordnung der Bewegungsmodelle zu den Blöcken die Übereinstimmung der Bewegungsmodelle mit der Verschiebung innerhalb der jeweiligen Blöcke sowie die räumliche und/oder die zeitliche Konsistenz der Zuordnung bewertet werden.

2. Verfahren nach Anspruch 1, dadurch gekennzeichnet, daß für die Bewertung der räumlichen Konsistenz benachbarte Blöcke berücksichtigt werden.

3. Verfahren nach Anspruch 1, dadurch gekennzeichnet, daß für die Bewertung der zeitlichen Konsistenz frühere Videobilddaten in einer Sequenz von Videobildern berücksichtigt werden.

4. Verfahren nach Anspruch 1, dadurch gekennzeichnet, daß die Parametersätze der zwei oder mehr Bewegungsmodelle aus jeweils zwei bis vier Parametern bestehen und der Parametersatz eines zusätzlichen Bewegungsmodell aus sechs oder mehr Parametern besteht.

5. Verfahren nach Anspruch 4, dadurch gekennzeichnet, daß der Parametersatz des zusätzlichen Bewegungsmodells bei der Zuordnung der Bewegungsmodelle zu den Bildobjekten bestimmt wird.

6. Vorrichtung zur Bewegungsschätzung in Videobilddaten, mit einem digitalen Bildspeicher, in dem ein erstes und ein zweites Videobild speicherbar sind, mit Mitteln zur

Bestimmung von Parametersätzen von zwei oder mehr Bewegungsmodellen und mit Mitteln zur Zuordnung der Bewegungsmodelle zu Bildobjekten, gekennzeichnet durch Mittel zur Unterteilung des gesamten Bildbereiches in Blöcke, aus denen die Bildobjekte zusammengesetzt werden, und zur Zuordnung der Blöcke zu den Bewegungsmodellen, wobei die Übereinstimmung der Bewegungsmodelle mit der Verschiebung innerhalb der jeweiligen Blöcke sowie die räumliche und zeitliche Konsistenz der Zuordnung bewertet werden.

7. Vorrichtung zur Darstellung von Videobildern, insbesondere Fernsehgerät oder Monitor, mit einem digitalen Bildspeicher (22), in dem Videobilddaten speicherbar sind, mit elektronischen Mitteln (21,25) zur Verarbeitung der in dem Bildspeicher gespeicherten Bilddaten und zur Darstellung von Videobildern auf einer Anzeigevorrichtung (28), wobei die Mittel (21) zur Verarbeitung der Bilddaten Mittel zur Bestimmung von Parametersätzen von zwei oder mehr Bewegungsmodellen und Mittel zur Zuordnung der Bewegungsmodelle zu Bildobjekten aufweisen, dadurch gekennzeichnet, daß die Mittel (21) zur Verarbeitung der Bilddaten ferner Mittel zur Unterteilung des gesamten Bildbereiches in Blöcke, aus denen die Bildobjekte zusammengesetzt werden, und zur Zuordnung der Blöcke zu den Bewegungsmodellen aufweisen, wobei die Übereinstimmung der Bewegungsmodelle mit der Verschiebung innerhalb der jeweiligen Blöcke sowie die räumliche und zeitliche Konsistenz der Zuordnung bewertet werden.

8. Computerprogrammprodukt zur Bewegungsschätzung in Videobilddaten, welches als Eingabe ein Paar von Videobilddatensätzen erhält und davon ausgehend zunächst Parametersätze von zwei oder mehr Bewegungsmodellen berechnet, dann den Bewegungsmodellen Bildobjekte zuordnet und schließlich Bewegungsdaten ausgibt, welche die Verschiebung der Bildobjekte vom einen zum anderen Bild beschreiben, dadurch gekennzeichnet, daß es den gesamten Bildbereich in Blöcke unterteilt, aus denen die Bildobjekte zusammengesetzt werden, wobei es für die Zuordnung der Blöcke zu den Bewegungsmodellen die Übereinstimmung der Bewegungsmodelle mit der Verschiebung innerhalb der jeweiligen Blöcke sowie die räumliche und zeitliche Konsistenz der Zuordnung bewertet.

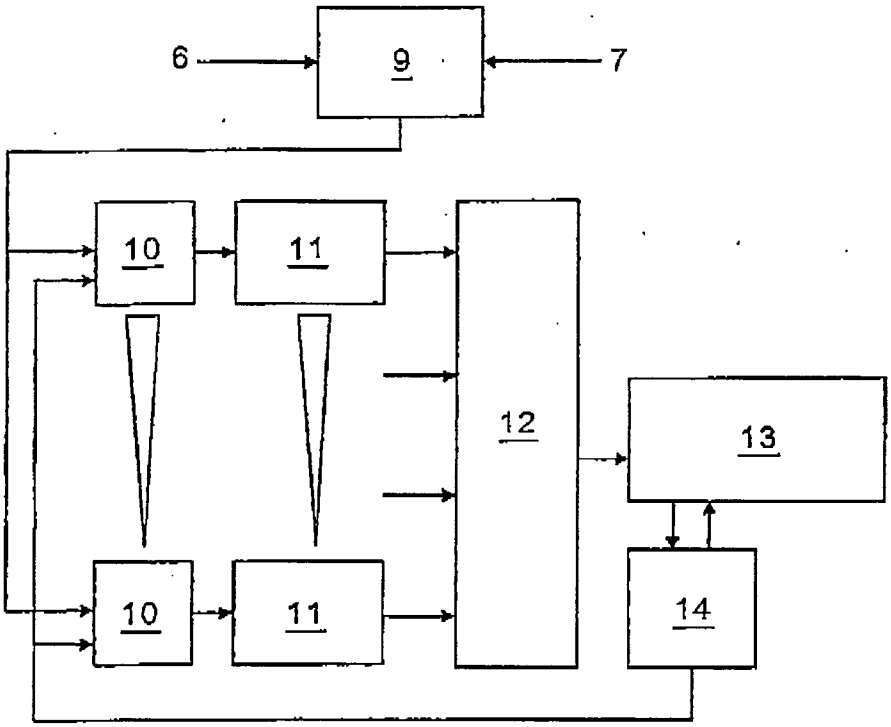


FIG. 1

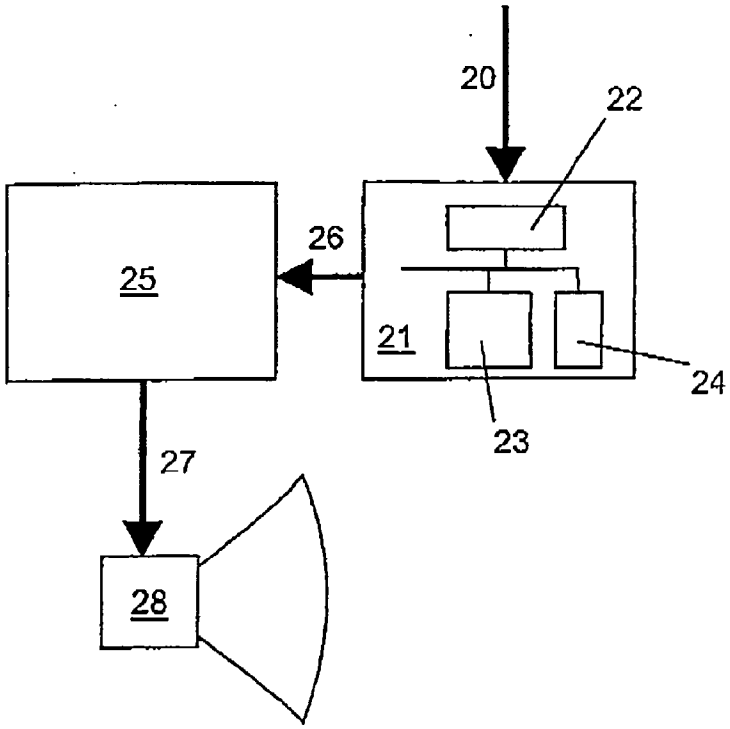


FIG. 2

INTERNATIONAL SEARCH REPORT

Inter nal Application No
PCT/EP 00/11653

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04N7/26

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H04N

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, INSPEC, WPI Data, PAJ

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	WO 99 16251 A (KONINKL PHILIPS ELECTRONICS NV ;PHILIPS SVENSKA AB (SE)) 1 April 1999 (1999-04-01) cited in the application claims 1-6	1-8
Y	BLACK M J ET AL: "THE ROBUST ESTIMATION OF MULTIPLE MOTIONS: PARAMETRIC AND PIECEWISE-SMOOTH FLOW FIELDS" COMPUTER VISION AND IMAGE UNDERSTANDING,US,ACADEMIC PRESS, vol. 63, no. 1, 1996, pages 75-104, XP000582617 ISSN: 1077-3142 page 76, right-hand column, paragraph 2 -page 77, left-hand column, paragraph 2.1 page 80, paragraph 2.2 -/--	1-8

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
9 March 2001	12/04/2001

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040. Tx. 31 651 epo nl, Fax (+31-70) 340-3016	Foglia, P

Form PCT/ISA/210 (second sheet) (July 1992)

PAGE 57/62 * RCVD AT 6/21/2006 11:39:02 AM [Eastern Daylight Time] * SVR:USPTO-EFXRF-5/8 * DNIS:2738300 * CSID:6508575487 * DURATION (mm-ss):18-08

INTERNATIONAL SEARCH REPORT

Intern. nal Application No
PCT/EP 00/11653

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	WEISS Y ET AL: "A UNIFIED MIXTURE FRAMEWORK FOR MOTION SEGMENTATION: INCORPORATING SPATIAL COHERENCE AND ESTIMATING THE NUMBER OF MODELS" PROCEEDINGS OF THE IEEE COMPUTER SOCIETY CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION,US,LOS ALAMITOS, IEEE COMP. SOC. PRESS, 18 June 1996 (1996-06-18), pages 321-326, XP000640250 ISBN: 0-8186-7258-7 page 322, paragraph 2.1 -page 323, paragraph 22	1-8
A	WANG J Y A ET AL: "SPATIO-TEMPORAL SEGMENTATION OF VIDEO DATA" PROCEEDINGS OF THE SPIE, 7 February 1994 (1994-02-07), XP000574646 page 121, paragraph 2.1 -page 122, paragraph 2.2	1-8
A	ELIAS ET AL.: "Image Sequence Segmentation using Mixture-Models with Uniqueness and Spatio-Temporal Consistency Constraints" IEE COLLOQUIUM ON MOTION ANALYSIS AND TRACKING, 10 May 1999 (1999-05-10), pages 6/1-6/6, XP002162507 paragraph '03.2!	1-8
A	US 5 999 651 A (CHANG YUH-LIN ET AL) 7 December 1999 (1999-12-07) column 9, line 39 -column 10, line 45; figures 1,3A,3B	1-8
A	STEINBACH ET AL.: "Using Multiple Global Motion Models for Improved Block-Based Video Coding" ICIP 99, vol. 2, 24 October 1999 (1999-10-24), pages 56-60, XP002162482 page 57, paragraph 2 -page 58, paragraph 4; figure 1	1-8
A	GB 2 317 525 A (NOKIA MOBILE PHONES LTD) 25 March 1998 (1998-03-25) the whole document	1-8

1

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

PAGE 58/62 * RCVD AT 6/21/2006 11:39:02 AM [Eastern Daylight Time] * SVR:USPTO-EFXRF-5/8 * DNIS:2738300 * CSID:6508575487 * DURATION (mm-ss):18-08

INTERNATIONAL SEARCH REPORT

information on patent family members

Inter nal Application No
PCT/EP 00/11653

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9916251 A	01-04-1999	EP 0940041 A	08-09-1999
		EP 0943209 A	22-09-1999
		WO 9916248 A	01-04-1999
US 5999651 A	07-12-1999	NONE	
GB 2317525 A	25-03-1998	AU 4133197 A	14-04-1998
		EP 0927494 A	07-07-1999
		WO 9812877 A	26-03-1998

Form PCT/ISA/210 (patent family annex) (July 1992)

INTERNATIONALER RECHERCHENBERICHT

Intern. nales Aktenzeichen
PCT/EP 00/11653

A. KLASSIFIZIERUNG DES ANMELDUNGSGEGENSTANDES
IPK 7 H04N7/26

Nach der internationalen Patentklassifikation (IPK) oder nach der nationalen Klassifikation und der IPK

B. RECHERCHIERTE GEBIETE

Recherchierter Mindestprüfstoff (Klassifikationssystem und Klassifikationssymbole)
IPK 7 H04N

Recherchierte aber nicht zum Mindestprüfstoff gehörende Veröffentlichungen, soweit diese unter die recherchierten Gebiete fallen

Während der internationalen Recherche konsultierte elektronische Datenbank (Name der Datenbank und evtl. verwendete Suchbegriffe)
EPO-Internal, INSPEC, WPI Data, PAJ

C. ALS WESENTLICH ANGESEHENE UNTERLAGEN

Kategorie*	Bezeichnung der Veröffentlichung, soweit erforderlich unter Angabe der in Betracht kommenden Teile	Betr. Anspruch Nr.
Y	WO 99 16251 A (KONINKL PHILIPS ELECTRONICS NV ;PHILIPS SVENSKA AB (SE)) 1. April 1999 (1999-04-01) in der Anmeldung erwähnt Ansprüche 1-6	1-8
Y	BLACK M J ET AL: "THE ROBUST ESTIMATION OF MULTIPLE MOTIONS: PARAMETRIC AND PIECEWISE-SMOOTH FLOW FIELDS" COMPUTER VISION AND IMAGE UNDERSTANDING,US,ACADEMIC PRESS, Bd. 63, Nr. 1, 1996, Seiten 75-104, XP000582617 ISSN: 1077-3142 Seite 76, rechte Spalte, Absatz 2 -Seite 77, linke Spalte, Absatz 2.1 Seite 80, Absatz 2.2	1-8

-/--

[X] Weitere Veröffentlichungen sind der Fortsetzung von Feld C zu entnehmen

[X] Siehe Anhang Patentfamilie

* Besondere Kategorien von angegebenen Veröffentlichungen :
"A" Veröffentlichung, die den allgemeinen Stand der Technik definiert, aber nicht als besonders bedeutsam anzusehen ist
"E" älteres Dokument, das jedoch erst am oder nach dem internationalen Anmeldedatum veröffentlicht worden ist
"L" Veröffentlichung, die geeignet ist, einen Prioritätsanspruch zweifelhaft erscheinen zu lassen, oder durch die das Veröffentlichungsdatum einer anderen im Recherchenbericht genannten Veröffentlichung belegt werden soll oder die aus einem anderen besonderen Grund angegeben ist (wie ausgeführt)
"O" Veröffentlichung, die sich auf eine mündliche Offenbarung, eine Benutzung, eine Ausstellung oder andere Maßnahmen bezieht
"P" Veröffentlichung, die vor dem internationalen Anmeldedatum, aber nach dem beanspruchten Prioritätsdatum veröffentlicht worden ist
"T" Spätere Veröffentlichung, die nach dem internationalen Anmeldedatum oder dem Prioritätsdatum veröffentlicht worden ist und mit der Anmeldung nicht kollidiert, sondern nur zum Verständnis des der Erfindung zugrundeliegenden Prinzips oder der ihr zugrundeliegenden Theorie angegeben ist
"X" Veröffentlichung von besonderer Bedeutung; die beanspruchte Erfindung kann allein aufgrund dieser Veröffentlichung nicht als neu oder auf erfinderischer Tätigkeit beruhend betrachtet werden
"Y" Veröffentlichung von besonderer Bedeutung; die beanspruchte Erfindung kann nicht als auf erfinderischer Tätigkeit beruhend betrachtet werden, wenn die Veröffentlichung mit einer oder mehreren anderen Veröffentlichungen dieser Kategorie in Verbindung gebracht wird und diese Verbindung für einen Fachmann naheliegend ist
"&" Veröffentlichung, die Mitglied derselben Patentfamilie ist

Datum des Abschlusses der internationalen Recherche	Absendedatum des internationalen Recherchenberichts
9. März 2001	12/04/2001

Name und Postanschrift der Internationalen Recherchenbehörde	Bevollmächtigter Bediensteter
Europäisches Patentamt, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl Fax (+31-70) 340-3016	Foglia, P

1

Formblatt PCT/ISA/210 (Blatt 2) (Juli 1992)

PAGE 60/62 * RCVD AT 6/21/2006 11:39:02 AM [Eastern Daylight Time] * SVR:USPTO-EFXRF-5/8 * DNIS:2738300 * CSID:6508575487 * DURATION (mm-ss):18-08

INTERNATIONALER RECHERCHENBERICHT

Internales Aktenzeichen
PCT/EP 00/11653

C.(Fortsetzung) ALS WESENTLICH ANGESEHENE UNTERLAGEN

Kategorie°	Bezeichnung der Veröffentlichung, soweit erforderlich unter Angabe der in Betracht kommenden Teile	Betr. Anspruch Nr.
A	WEISS Y ET AL: "A UNIFIED MIXTURE FRAMEWORK FOR MOTION SEGMENTATION: INCORPORATING SPATIAL COHERENCE AND ESTIMATING THE NUMBER OF MODELS" PROCEEDINGS OF THE IEEE COMPUTER SOCIETY CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION,US,LOS ALAMITOS, IEEE COMP. SOC. PRESS, 18. Juni 1996 (1996-06-18), Seiten 321-326, XP000640250 ISBN: 0-8186-7258-7 Seite 322, Absatz 2.1 -Seite 323, Absatz 22	1-8
A	WANG J Y A ET AL: "SPATIO-TEMPORAL SEGMENTATION OF VIDEO DATA" PROCEEDINGS OF THE SPIE, 7. Februar 1994 (1994-02-07), XP000574646 Seite 121, Absatz 2.1 -Seite 122, Absatz 2.2	1-8
A	ELIAS ET AL.: "Image Sequence Segmentation using Mixture-Models with Uniqueness and Spatio-Temporal Consistency Constraints" IEE COLLOQUIUM ON MOTION ANALYSIS AND TRACKING, 10. Mai 1999 (1999-05-10), Seiten 6/1-6/6, XP002162507 Absatz '03.2!	1-8
A	US 5 999 651 A (CHANG YUH-LIN ET AL) 7. Dezember 1999 (1999-12-07) Spalte 9, Zeile 39 -Spalte 10, Zeile 45; Abbildungen 1,3A,3B	1-8
A	STEINBACH ET AL.: "Using Multiple Global Motion Models for Improved Block-Based Video Coding" ICIP 99, Bd. 2, 24. Oktober 1999 (1999-10-24), Seiten 56-60, XP002162482 Seite 57, Absatz 2 -Seite 58, Absatz 4; Abbildung 1	1-8
A	GB 2 317 525 A (NOKIA MOBILE PHONES LTD) 25. März 1998 (1998-03-25) das ganze Dokument	1-8

1

Formblatt PCT/ISA/210 (Fortsetzung von Blatt 2) (Juli 1992)

PAGE 61/62 * RCVD AT 6/21/2006 11:39:02 AM [Eastern Daylight Time] * SVR:USPTO-EFXRF-5/8 * DNIS:2738300 * CSID:6508575487 * DURATION (mm-ss):18-08

INTERNATIONALER RECHERCHENBERICHT

Angaben zu Veröffentlichungen, die zur selben Patentfamilie gehören

Intern ...ales Aktenzeichen
PCT/EP 00/11653

Im Recherchenbericht angeführtes Patentdokument		Datum der Veröffentlichung	Mitglied(er) der Patentfamilie		Datum der Veröffentlichung
WO 9916251	A	01-04-1999	EP	0940041 A	08-09-1999
			EP	0943209 A	22-09-1999
			WO	9916248 A	01-04-1999
US 5999651	A	07-12-1999	KEINE		
GB 2317525	A	25-03-1998	AU	4133197 A	14-04-1998
			EP	0927494 A	07-07-1999
			WO	9812877 A	26-03-1998

Formblatt PCT/ISA/210 (Anhang Patentfamilie)(Juli 1992)

PAGE 62/62 * RCVD AT 6/21/2006 11:39:02 AM [Eastern Daylight Time] * SVR:USPTO-EFXRF-5/8 * DNIS:2738300 * CSID:6508575487 * DURATION (mm-ss):18-08

Electronic Acknowledgement Receipt

EFS ID:	5416814
Application Number:	10972588
International Application Number:	
Confirmation Number:	1079
Title of Invention:	SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR
First Named Inventor/Applicant Name:	Scott L. Michaelis
Customer Number:	22879
Filer:	Robert Wayne Nelson/Kalyn Black
Filer Authorized By:	Robert Wayne Nelson
Attorney Docket Number:	200404973-1
Receipt Date:	28-MAY-2009
Filing Date:	25-OCT-2004
Time Stamp:	19:27:01
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Transmittal Letter	E-IDS.pdf	535632 9874f45496f4b448c8a1c021430f814d55c3d76a	no	1

Warnings:	
Information:	

2	Information Disclosure Statement (IDS) Filed (SB/08)	sb0008a-1.pdf	287377 a14dc61b08744aeeb17a921c0a582acddbefde17	no	1
Warnings:					
Information:					
This is not an USPTO supplied IDS fillable form					
3	Foreign Reference	WO_01_45421.pdf	861429 d395ece498004ac3b16c91f72c98fed0b7a072fb	no	21
Warnings:					
Information:					
			Total Files Size (in bytes):	1684438	

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 200404973-1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): Scott L. Michaelis	Confirmation No.: 1079
Application No.: 10/972,588	Examiner: Le, Dieu Minh T.
Filing Date: 10/25/2004	Group Art Unit: 2114

Title: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

Mail Stop Amendment
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

INFORMATION DISCLOSURE STATEMENT

This Information Disclosure Statement is submitted:

☒ under 37 CFR 1.97 (b), or
(Within three months of filing national application; or date of entry of national application; or before mailing date of first action on the merits; whichever occurs last)

☐ under 37 CFR 1.97 (c), together with either a:
- ☐ Statement under 37 CFR 1.97 (e), or
- ☐ a $180 fee under 37 CFR 1.17 (p), or

(After the CFR 1.97 (b) time period, but before final action or notice of allowance, whichever occurs first)

☐ under 37 CFR 1.97 (d), together with a:
- ☐ Statement under 37 CFR 1.97 (e)(1) or (2), and
- ☐ a $180 fee set forth in 37 CFR 1.17 (p)

(Filed after final action, a notice of allowance, on or before payment of the issue fee)

Please charge to Deposit Account **08-2025** the sum of $ 0.00 . At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account **08-2025** pursuant to 37 CFR 1.25.

Applicant(s) submit herewith form PTO/SB/08B - Information Disclosure Statement together with any required copies of patents, publications or other information of which applicant(s) are aware, which applicant(s) believe(s) may be material to the examination of this application and for which there may be a duty to disclose in accordance with 37 CFR 1.56.

☐ A concise explanation of the relevance of foreign language patents, foreign language publications and other foreign language information listed on PTO/SB/08B, as presently understood by the individual(s) designated in 37 CFR 1.56 (c) most knowledgeable about the content is given on the attached sheet, or where a foreign language patent is cited in a search report or other action by a foreign patent office in a counterpart foreign application, an english language version of the search report or action which indicates the degree of relevance found by the foreign office is listed on form PTO/SB/08B and is enclosed.

It is requested that the information disclosed herein be made of record in this application.

Respectfully submitted,

Scott L. Michaelis

By: /Robert W. Nelson/

Robert W. Nelson, Esq.

Attorney/Agent for Applicant(s)

Reg No. : 37,898

Date : May 28, 2009

Telephone : (303) 298-9888

PTO/SB/08b (03-09)
Approved for use through 04/30/2009. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 1

Complete if Known	
Application Number	10/972,588
Filing Date	10/25/2004
First Named Inventor	Scott L. Michaelis
Art Unit	2114
Examiner Name	LE, DIEU MINH T.
Attorney Docket Number	200404973-1

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		Notice of Allowance dated 2-11-2009 for Serial Number 10/973,077 in the United States	
		Notice of Allowance dated 9-23-2008 for Serial Number 10/973,077 in the United States	
		Office Action dated 3-19-2008 for Serial Number 10/973,077 in the United States	
		Office Action dated 9-20-2007 for Serial Number 10/973,077 in the United States	

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

1 Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

Electronic Patent Application Fee Transmittal

Application Number:	10972588
Filing Date:	25-Oct-2004
Title of Invention:	SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR
First Named Inventor/Applicant Name:	Scott L. Michaelis
Filer:	Robert Wayne Nelson/Joy Reinhart
Attorney Docket Number:	200404973-1

Filed as Large Entity

Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
Request for continued examination	1801	1	810	810
		Total in USD ($)		**810**

Electronic Acknowledgement Receipt

EFS ID:	5321771
Application Number:	10972588
International Application Number:	
Confirmation Number:	1079
Title of Invention:	SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR
First Named Inventor/Applicant Name:	Scott L. Michaelis
Customer Number:	22879
Filer:	Robert Wayne Nelson/Joy Reinhart
Filer Authorized By:	Robert Wayne Nelson
Attorney Docket Number:	200404973-1
Receipt Date:	12-MAY-2009
Filing Date:	25-OCT-2004
Time Stamp:	17:06:16
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Request for Continued Examination (RCE)	RCE.pdf	94300 e53924bece3f095ee139333818aaf97227164530	no	2

Warnings:

This is not a USPTO-supplied RCE SB30 form.

Information:					
2	Information Disclosure Statement (IDS) Filed (SB/08)	sb0008b.pdf	221151 398004ad03d7716dfcfffbfdbb7bde34caa003d0	no	1
Warnings:					
Information:					
This is not an USPTO supplied IDS fillable form					
3	NPL Documents	10973077_NOA_2-11-2009.pdf	2036476 d3a41e61c354ba789db1130ab1058f425f7002f1	no	33
Warnings:					
Information:					
4	NPL Documents	10973077_NOA_9-23-2008.pdf	1017594 685f56bf32e0c8e880647a2164ca45a6a09328d0	no	20
Warnings:					
Information:					
5	NPL Documents	10973077_OA_3-19-2008.pdf	839166 aca4c08012b5a79121033157cf0188b46f4fa11e	no	25
Warnings:					
Information:					
6	NPL Documents	10973077_OA_9-20-2007.pdf	966565 0a6f57eea1d12336992f7dc78210495749ca49f7	no	30
Warnings:					
Information:					
7	Fee Worksheet (PTO-875)	fee-info.pdf	30791 630b352631f90570cbb275e09df6ee8187a295ec	no	2
Warnings:					
Information:					
			Total Files Size (in bytes):	5206043	

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 200404973-1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): Scott L. Michaelis	Confirmation No.: 1079
Application No.: 10/972,588	Examiner: Le, Dieu Minh T.
Filing Date: 10/25/2004	Group Art Unit: 2114

Title: SYSTEM AND METHOD FOR SWITCHING THE ROLE BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

Mail Stop RCE
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

REQUEST FOR CONTINUED EXAMINATION (RCE) 37 CFR 1.114

Subsection (b) of 35 U.S.C. 132, effective on May 29, 2000, provides for continued examination of a utiltity or plant application filed on or after June 8, 1995.
See The American Inventors Protection Act of 1999 (AIPA).

This is a Request for Continued Examination (RCE) under CFR 1.114 of the above-identified application.

NOTE: 37 CFR 1.114 is effective on May 20, 2000. If the above- application was filed prior to May 29, 2000, applicant may wish to consider filing a continued prosecution application (CPA) under CFR 1.53(d) (PTO/SB/29) instead of a RCE to be eligible for the patent term adjustment provisions of the AIPA. See Changes to Application Examination and Provisional Application Practice, Interim Rule, 65 Fed. Reg. 14865 (Mar. 20, 2000), 1233 off. Gaz. Pat. Office

Submission under 37 CFR 1.114

☐ Previously submitted

- ☐ Consider the amendment(s)/reply under 37 CFR 1.116 previously filed on _____.
  (Any unentered amendment(s) referred to above will be entered).
- ☐ Consider the arguments in the Appeal Brief or Reply Brief prevously filed on _____.
- ☐ Other _____

☒ Enclosed

- ☐ Amendment/Reply
- ☐ Affidavit(s)/Declaration(s)
- ☒ Information Disclosure Statement (IDS)
- ☐ Other _____

Miscellaneous

- ☐ Suspension of action is requested under 37 CFR 1.103(c) for a period of _____ months.
  The fee for this Suspension is (37 CFR 1.17(i)) $130.00
- ☐ Other _____

CONTINUED EXAMINATION TRANSMITTAL (RCE) (37 CFR 1.114) (continued)	PATENT APPLICATION ATTORNEY DOCKET NO. 200404973-1

☒ RCE filing fee $810.00

☐ A Petition for Extension of Time

☐ 1st Month $130 ☐ 2nd Month $490 ☐ 3rd Month $1110 ☐ 4th Month $1730

☒ Please charge to Deposit Account 08-2025 the sum of $ 810 . At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that my regulate fees.

Respectfully submitted,

Scott L. Michaelis

By: /Robert W. Nelson/

Robert W. Nelson, Esq.

Attorney/Agent for Applicant(s)

Reg No. : 37,898

Date : May 12, 2009

Telephone : (303) 298-9888

Document code: WFEE

United States Patent and Trademark Office
Sales Receipt for Accounting Date: 08/03/2009

ABOARDLE SALE #00000046 Mailroom Dt: 05/12/2009 082025 10972588
01 FC : 1801 810.00 DA

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

NOTICE OF ALLOWANCE AND FEE(S) DUE

22879 7590 02/12/2009

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

EXAMINER	
LE, DIEU MINH T	
ART UNIT	PAPER NUMBER
2114	

DATE MAILED: 02/12/2009

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/972,588	10/25/2004	Scott L. Michaelis	200404973-1	1079

TITLE OF INVENTION: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

APPLN. TYPE	SMALL ENTITY	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	$1510	$300	$0	$1810	05/12/2009

THE APPLICATION IDENTIFIED ABOVE HAS BEEN EXAMINED AND IS ALLOWED FOR ISSUANCE AS A PATENT. PROSECUTION ON THE MERITS IS CLOSED. THIS NOTICE OF ALLOWANCE IS NOT A GRANT OF PATENT RIGHTS. THIS APPLICATION IS SUBJECT TO WITHDRAWAL FROM ISSUE AT THE INITIATIVE OF THE OFFICE OR UPON PETITION BY THE APPLICANT. SEE 37 CFR 1.313 AND MPEP 1308.

THE ISSUE FEE AND PUBLICATION FEE (IF REQUIRED) MUST BE PAID WITHIN THREE MONTHS FROM THE MAILING DATE OF THIS NOTICE OR THIS APPLICATION SHALL BE REGARDED AS ABANDONED. THIS STATUTORY PERIOD CANNOT BE EXTENDED. SEE 35 U.S.C. 151. THE ISSUE FEE DUE INDICATED ABOVE DOES NOT REFLECT A CREDIT FOR ANY PREVIOUSLY PAID ISSUE FEE IN THIS APPLICATION. IF AN ISSUE FEE HAS PREVIOUSLY BEEN PAID IN THIS APPLICATION (AS SHOWN ABOVE), THE RETURN OF PART B OF THIS FORM WILL BE CONSIDERED A REQUEST TO REAPPLY THE PREVIOUSLY PAID ISSUE FEE TOWARD THE ISSUE FEE NOW DUE.

HOW TO REPLY TO THIS NOTICE:

I. Review the SMALL ENTITY status shown above.

If the SMALL ENTITY is shown as YES, verify your current SMALL ENTITY status:

A. If the status is the same, pay the TOTAL FEE(S) DUE shown above.

B. If the status above is to be removed, check box 5b on Part B - Fee(s) Transmittal and pay the PUBLICATION FEE (if required) and twice the amount of the ISSUE FEE shown above, or

If the SMALL ENTITY is shown as NO:

A. Pay TOTAL FEE(S) DUE shown above, or

B. If applicant claimed SMALL ENTITY status before, or is now claiming SMALL ENTITY status, check box 5a on Part B - Fee(s) Transmittal and pay the PUBLICATION FEE (if required) and 1/2 the ISSUE FEE shown above.

II. PART B - FEE(S) TRANSMITTAL, or its equivalent, must be completed and returned to the United States Patent and Trademark Office (USPTO) with your ISSUE FEE and PUBLICATION FEE (if required). If you are charging the fee(s) to your deposit account, section "4b" of Part B - Fee(s) Transmittal should be completed and an extra copy of the form should be submitted. If an equivalent of Part B is filed, a request to reapply a previously paid issue fee must be clearly made, and delays in processing may occur due to the difficulty in recognizing the paper as an equivalent of Part B.

III. All communications regarding this application must give the application number. Please direct all communications prior to issuance to Mail Stop ISSUE FEE unless advised to the contrary.

IMPORTANT REMINDER: Utility patents issuing on applications filed on or after Dec. 12, 1980 may require payment of maintenance fees. It is patentee's responsibility to ensure timely payment of maintenance fees when due.

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail **Mail Stop ISSUE FEE**
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (571)-273-2885

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

22879 7590 02/12/2009

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (571) 273-2885, on the date indicated below.

(Depositor's name)
(Signature)
(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/972,588	10/25/2004	Scott L. Michaelis	200404973-1	1079

TITLE OF INVENTION: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

APPLN. TYPE	SMALL ENTITY	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	$1510	$300	$0	$1810	05/12/2009

EXAMINER	ART UNIT	CLASS-SUBCLASS
LE, DIEU MINH T	2114	714-012000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).

❑ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.

❑ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. **Use of a Customer Number is required.**

2. For printing on the patent front page, list

(1) the names of up to 3 registered patent attorneys or agents OR, alternatively,

(2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.

1 _____

2 _____

3 _____

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE (B) RESIDENCE: (CITY and STATE OR COUNTRY)

Please check the appropriate assignee category or categories (will not be printed on the patent) : ❑ Individual ❑ Corporation or other private group entity ❑ Government

4a. The following fee(s) are submitted:

❑ Issue Fee

❑ Publication Fee (No small entity discount permitted)

❑ Advance Order - # of Copies _____

4b. Payment of Fee(s): **(Please first reapply any previously paid issue fee shown above)**

❑ A check is enclosed.

❑ Payment by credit card. Form PTO-2038 is attached.

❑ The Director is hereby authorized to charge the required fee(s), any deficiency, or credit any overpayment, to Deposit Account Number _____ (enclose an extra copy of this form).

5. **Change in Entity Status** (from status indicated above)

❑ a. Applicant claims SMALL ENTITY status. See 37 CFR 1.27.

❑ b. Applicant is no longer claiming SMALL ENTITY status. See 37 CFR 1.27(g)(2).

NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

Authorized Signature _____ Date _____

Typed or printed name _____ Registration No. _____

This collection of information is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, Virginia 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PTOL-85 (Rev. 08/07) Approved for use through 08/31/2010. OMB 0651-0033 U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/972,588	10/25/2004	Scott L. Michaelis	200404973-1	1079

22879 7590 02/12/2009

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

EXAMINER	
LE, DIEU MINH T	
ART UNIT	PAPER NUMBER
2114	

DATE MAILED: 02/12/2009

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)

(application filed on or after May 29, 2000)

The Patent Term Adjustment to date is 529 day(s). If the issue fee is paid on the date that is three months after the mailing date of this notice and the patent issues on the Tuesday before the date that is 28 weeks (six and a half months) after the mailing date of this notice, the Patent Term Adjustment will be 529 day(s).

If a Continued Prosecution Application (CPA) was filed in the above-identified application, the filing date that determines Patent Term Adjustment is the filing date of the most recent CPA.

Applicant will be able to obtain more detailed information by accessing the Patent Application Information Retrieval (PAIR) WEB site (http://pair.uspto.gov).

Any questions regarding the Patent Term Extension or Adjustment determination should be directed to the Office of Patent Legal Administration at (571)-272-7702. Questions relating to issue and publication fee payments should be directed to the Customer Service Center of the Office of Patent Publication at 1-(888)-786-0101 or (571)-272-4200.

Notice of Allowability	Application No.	Applicant(s)	
	10/972,588	MICHAELIS ET AL.	
	Examiner	Art Unit	
	Dieu-Minh Le	2114	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to *the communication filed on 12/09/2008*.

2. ☒ The allowed claim(s) is/are *1-12, and 16-30 [now as 1-27]*.

3. ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
   a) ☐ All b) ☐ Some* c) ☐ None of the:
      1. ☐ Certified copies of the priority documents have been received.
      2. ☐ Certified copies of the priority documents have been received in Application No. _____ .
      3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

   * Certified copies not received: _____.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application.
THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.

4. ☐ A SUBSTITUTE OATH OR DECLARATION must be submitted. Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL PATENT APPLICATION (PTO-152) which gives reason(s) why the oath or declaration is deficient.

5. ☐ CORRECTED DRAWINGS (as "replacement sheets") must be submitted.
   (a) ☐ including changes required by the Notice of Draftsperson's Patent Drawing Review (PTO-948) attached
      1) ☐ hereto or 2) ☐ to Paper No./Mail Date _____.
   (b) ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date _____.

   Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).

6. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1. ☐ Notice of References Cited (PTO-892)
2. ☐ Notice of Draftperson's Patent Drawing Review (PTO-948)
3. ☒ Information Disclosure Statements (PTO/SB/08), Paper No./Mail Date 12/09/2008
4. ☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
5. ☐ Notice of Informal Patent Application
6. ☐ Interview Summary (PTO-413), Paper No./Mail Date _____ .
7. ☐ Examiner's Amendment/Comment
8. ☐ Examiner's Statement of Reasons for Allowance
9. ☐ Other _____.

/Dieu-Minh Le/
Primary Examiner, Art Unit 2114

Issue Classification

Application/Control No.	Applicant(s)/Patent Under Reexamination
10972588	MICHAELIS ET AL.
Examiner	**Art Unit**
Dieu-Minh Le	2114

ORIGINAL		INTERNATIONAL CLASSIFICATION														
CLASS	SUBCLASS	CLAIMED					NON-CLAIMED									
714	12	G	0	6	F	11 / 00 (2006.01.01)										

CROSS REFERENCE(S)					
CLASS	SUBCLASS (ONE SUBCLASS PER BLOCK)				
714	11				

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original
1	1	14	17												
2	2	15	18												
3	3	16	19												
4	4	17	20												
5	5	18	21												
6	6	19	22												
7	7	20	23												
8	8	21	24												
9	9	22	25												
10	10	23	26												
11	11	24	27												
12	12	25	28												
	13	26	29												
	14	27	30												
	15														
13	16														

NONE		**Total Claims Allowed:**	
(Assistant Examiner)	(Date)	27	
/Dieu-Minh Le/ Primary Examiner.Art Unit 2114	2/1/2009	O.G. Print Claim(s)	O.G. Print Figure
(Primary Examiner)	(Date)	1	1

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	10972588	MICHAELIS ET AL.
	Examiner	Art Unit
	Dieu-Minh Le	2114

SEARCHED			
Class	Subclass	Date	Examiner
714	10, 11, 12, 13	1/31/09	dml
712	31	1/31/09	dml

SEARCH NOTES		
Search Notes	Date	Examiner
EAST search all files updated	1/31/09	dml
Inventor name search	1/31/09	dml
NPL search files [IEEE, Google Scholar]	2/1/09	dml

INTERFERENCE SEARCH			
Class	Subclass	Date	Examiner
714	10-13	1/31/09	dml
712	31	1/31/09	dml
	See Interference search data file		

	/Dieu-Minh Le/ Primary Examiner.Art Unit 2114

EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	34	("20020144177" | "20020152420" | "20030051190" | "20030070050" | "20030126498" | "20030135711" | "20030140281" | "20040006722" | "20040019771" | "20040078650" | "20040078651" | "20040153857" | "20050172164" | "20050240806" | "20050240811" | "20050240829" | "20060085677" | "20060248384" | "20060248684" | "4358823" | "5537655" | "5675794" | "5915082" | "5991900" | "6220581" | "6263452" | "6438687" | "6625749" | "6687851" | "6754787" | "6920581" | "7085959" | "7225355" | "7251746").PN.	US-PGPUB; USPAT; USOCR	OR	ON	2009/01/31 22:53
L2	2532	lockstep lock-step	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/01/31 23:15
L3	25197	(loss adj L2) LOL	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/01/31 23:15
L4	25667	multi-processor "multi processor" and L3	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/01/31 23:15

L5	1817	(boot$4 or initializ$4) adj processor	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/01/31 23:30
L6	302	spare adj processor	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/01/31 23:30
L7	20	L5 and L6	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/01/31 23:30
L8	18	4 and 7	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/01/31 23:31
L9	169476	without same ("shut$4 down" stop$4 idle interrupt$4) same (OS "operating system")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/01/31 23:32
L10	7	L9 and 8	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/01/31 23:32
L11	0	1 and 10	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/01/31 23:37
L12	30	(michaelis-scott$ or myer-sylvia$ or mchardy-william$).in.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/01/31 23:38
L13	836	(714/11).ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2009/01/31 23:38
L14	589	(714/10).ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2009/01/31 23:38

L15	396	(714/12).ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2009/01/31 23:38
L16	726	(714/13).ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2009/01/31 23:38
L17	395	(714/31).ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2009/01/31 23:38
L18	4	(L13 L14 L15 L16 L17) and L12	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/01/31 23:38
L19	2	(L13 L14 L15 L16 L17) and 10	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2009/01/31 23:38
L20	7	("714"/$).ccls. and 8 and 9	US-PGPUB; USPAT; USOCR	OR	ON	2009/01/31 23:43
L21	0	("711"/$).ccls. and 8 and 9	US-PGPUB; USPAT; USOCR	OR	ON	2009/01/31 23:44
L22	0	("709"/$).ccls. and 8 and 9	US-PGPUB; USPAT; USOCR	OR	ON	2009/01/31 23:44
S1	14	(("7085959") or ("20050172164") or ("5675794") or ("6438687") or ("6920581") or ("5915082") or ("6625749")).PN.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/26 17:54
S2	2	("7085959").PN.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/27 15:05
S3	730	(714/11).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/27 15:06
S4	337	(714/12).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 15:32

S5	606	(714/13).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/27 15:07
S6	36	("20020144177" | "20030070050" | "20040006722" | "20040019771" | "20040153857" | "5226152" | "5249188" | "5751932" | "5758058" | "5764660" | "6065135" | "6148348" | "6473869" | "6560682" | "6604177" | "6615366" | "6625749" | "6687851" | "6754787").PN.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/27 15:08
S7	44	("20020144177" | "20030070050" | "20040006722" | "20040019771" | "20040153857" | "6625749" | "6687851" | "6754787")".PN"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/27 15:09
S9	2222	lockstep lock-step	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 15:21
S11	25008	(loss adj S9) LOL	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 15:22
S12	51	(detect$3 or track$4) adj S11	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 15:23
S13	1547	(boot$4 or initializ$4) adj processor	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:41

S14	246	spare adj processor	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 15:27
S15	12	S12 and S13 and S14	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 15:30
S16	25	(michaelis-scott$ or myer-sylvia$ or mchardy-william$).in.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:14
S18	632	(714/10).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 15:33
S19	95	(712/31).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:08
S20	52536	(switch$4 chang$4 failover fail-over "fail over") same (spare slave redundan$4 backup back-up "back up" second$4) same processor	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:54
S21	81	S20 same S13	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:28
S22	168219	without same ("shut$4 down" stop$4 idle interrupt$4) same (OS "operating system")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:33

S23	6	S21 and S22	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:33
S24	20342	multi-processor "multi processor" and S11 and S13	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:45
S25	176	S24 and S20 and S22	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:46
S26	7	S25 and S14	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:53
S27	87	(recover$4 reestablish$3 restart$3) with S9	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:54
S28	31827	(pair spare slave redundan$4 backup back-up "back up" second$4) adj2 processor	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:55
S29	101834	firmware	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:56
S30	21	S27 and S28 and S29	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:57

S31	15153	trigger$3 same ("operating system" OS operating-system) same (idle stop$4 interrupt$3 halt$3)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 14:49
S32	207	(loss lose los$3 lost$3) same (lockstep lock-step "lock step")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 14:52
S33	86	(detect$3 track$3 monitor$3) same S32	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 14:56
S34	3	S33 same S31	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 14:56
S35	3	S32 same S31	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 15:02
S36	10	S32 and S31	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 15:02
S37	33339	(pair spare slave redundan$4 backup back-up "back up" second$4) adj2 processor	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 15:05
S38	2323	lockstep lock-step	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 15:06

S39	92	(recover$4 reestablish$3 restart$3) with S38	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 15:06
S40	114677	firmware	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 15:06
S41	11	S37 same S39 same S40	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 15:06
S42	764	(714/11).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 15:07
S43	352	(714/12).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 15:07
S44	659	(714/13).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 15:07
S45	658	(714/10).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 15:07
S46	96	(712/31).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 15:07

S47	2	(S42 S43 S44 S45 S46) and S31 and S33	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 15:07
S48	2	(S42 S43 S44 S45 S46) and S31 and S32 and S37 and S40	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/14 15:08
S49	804	(714/11).ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:19
S50	565	(714/10).ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:19
S51	371	(714/12).ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:19
S52	696	(714/13).ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:19
S53	381	(714/31).ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:19
S54	204	(loss lose los$3 lost$3) same (lockstep lock-step "lock step")	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:21
S55	1586	(boot$4 or initializ$4) adj processor	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:22
S56	52623	(switch$4 chang$4 failover fail-over "fail over") same (spare slave redundan$4 backup back-up "back up" second$4) same processor	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:22
S57	30192	(pair spare slave redundan$4 backup back-up "back up" second$4) adj2 processor	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:22
S58	82	(detect$3 track$3 monitor$3) same S54	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:23
S59	163830	without same ("shut$4 down" stop$4 idle interrupt$4) same (OS "operating system")	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:24
S60	32	S58 and S56 and S57	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:29
S61	7	S60 and S49	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:29

S62	3	S60 and S50 and S59	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:30
S63	7	S60 and (S51 S52 S53) and S59	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:31
S64	22	S60 and ("714"/$.ccls.) and S59	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:38
S65	12	(switch$4 chang$4) same role same S55	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:39
S66	7	S64 and S65	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:40

1/31/09 11:48:52 PM
C:\Documents and Settings\MLe\My Documents\EAST\Workspaces\10972588rce.wsp

IEEE Xplore
Home | Login | Logout | Access Information | Alerts | Purchase History | Cart | Sitemap | Help
Welcome United States Patent and Trademark Office
Celebrating 125 Years of Engineering the Future
IEEE
Search Session History
BROWSE SEARCH IEEE XPLORE GUIDE SUPPORT

Edit an existing query or compose a new query in the Search Query Display.

Select a search number (#) to:

- Add a query to the Search Query Display
- Combine search queries using AND, OR, or NOT
- Delete a search
- Run a search

Sun, 1 Feb 2009, 2:32:38 PM EST

Search Query Display

Run Search | Reset

Recent Search Queries		Results
#1	(((spare processor)<in>metadata) <and> ((system identifier) <in>metadata))<and> ((shutting down os)<in>metadata)	0
#2	(((switching changing)<in>metadata) <and> ((boot processor) <in>metadata))<and> ((spare processor)<in>metadata)	0
#3	(((detecting tracking monitoring)<in>metadata) <and> (((loss of lockstep) lol)<in>metadata))<and> ((boot processor) <in>metadata)	0
#4	((((loss of lockstep) lol)<in>metadata) <and> ((boot procesor) <in>metadata))<and> ((system identifier)<in>metadata)	0
#5	(((system firmware)<in>metadata) <and> ((lol)<in>metadata)) <and> ((error detection logic)<in>metadata)	0

Clear Search History

Help Contact Us Privacy & Security IEEE.org
Indexed by Inspec
© Copyright 2009 IEEE – All Rights Reserved

Web Results 1 - 10 of about 290 for **detect (loss of lockstep) (switch recover) (boot processor) (spare processor) (without) interrup OS**. (0.14 seconds)

1. (WO/1998/055922) LOOSELY-COUPLED, SYNCHRONIZED EXECUTION

The **processor** services the maskable **interrupt** some number of instructions .... TS already was set prior to the task **switch** into the **lockstep** control task. ...
www.wipo.int/pctdb/en/wo.jsp?wo=1998055922&IA=US1998011423&DISPLAY=DESC - 93k - Cached - Similar pages

2. EIDE flaw : Java Glossary

Because **OS**/2 has already signalled EOI (End Of **Interrupt**) to the PIC (Programmable .... Support line at 1-800-628-8686 forhelp with your **processor** issues. ...
mindprod.com/jg**loss**/eideflaw.html - 62k - Cached - Similar pages

3. [PDF] Manual Cover.fm

File Format: PDF/Adobe Acrobat - View as HTML
interface and does not use any legacy I/O or **interrupt** resources. ..... temperature sensor, and a fast acting TCC, the **processor**, **without** ...
www.powerbridge.de/download/manual/cpci/cpci-cpu/CPCI-6910_UM_0705.pdf - Similar pages

4. [PDF] Software Optimization Guide for the AMD Hammer **Processor**

File Format: PDF/Adobe Acrobat
AMD-V processors automatically deliver a pending virtual **interrupt** to the guest when ..... The **processor** continues to combine writes to this buffer **without** ...
www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/40546.pdf - Similar pages

5. Architecture and implementation for a high reliability long-term ...

Another important parameter, the **processor** throughput, was **... detection** of the CPU is to have two CPUs operate in **lock-step ...**
ieeexplore.ieee.org/iel2/1033/6983/00282118.pdf?arnumber=282118 - Similar pages
by D Gomez - 1992

6. [PDF] ATCA-4300/ATCA-4310 Compute Processing Module Reference

File Format: PDF/Adobe Acrobat
OS boot failure. Failure to **boot** from a PXE server or **detection** of a bad **boot .....** The **switch** from PIC mode to **Processor** System Bus **Interrupt ...**
www.radisys.com/files/support_downloads/02237-03_ATCA-43xx_CPM_Reference.pdf - Similar pages

7. [PDF] *EP000986784B1*

File Format: PDF/Adobe Acrobat - View as HTML
and directed to the **processor** through a specific **interrupt**. vector. **.....** the task **switch** into the **lockstep** control task. If it was, **...**
https://publications.european-patent-office.org/PublicationServer/documentpdf.jsp?iDocId=5130155&iebug=.pdf - Similar pages

8. [PDF] Fingerprinting: Hash-Based Error **Detection** in Microprocessors

File Format: PDF/Adobe Acrobat
In each run, the **processor boots** and starts the test program. **..... recovery** and **OS**-transparent redundant execution **without lockstep**. **...**
www.ece.cmu.edu/~jhoe/distribution/2008/jsmolens.pdf - Similar pages

9. [PDF] Intel 5000 Series Chipsets Server Board Family Datasheet

File Format: PDF/Adobe Acrobat
Setup Utility — Specific **Processor** Information Screen Display **.....** The system can **boot without** a keyboard or mouse attached. If present, the BIOS will **...**
www.cs.earlham.edu/~charliep/d38960004-s5000.pdf - Similar pages

10. [PS] Synthesis: An Ecient Implementation of Fundamental Operating ...

File Format: Adobe PostScript - View as HTML
symmetric multiprocessor, scheduling tasks on either **processor without .....** An

interrupt causing a context **switch** will execute the current program's ...
www.cs.columbia.edu/~library/TR-repository/reports/reports-1992/cucs-039-92.ps.gz
- Similar pages
by H Massalin - 1992 - Cited by 131 - Related articles

Gooooooooogle ▶
1 2 3 4 5 6 7 8 9 10 Next

detect (loss of lockstep) (switch recover) (boot processor) (spare processor) (wit Search

Search within results | Language Tools | Search Tips | Dissatisfied? Help us improve | Try Google Experimental

Google Home - Advertising Programs - Business Solutions - Privacy - About Google

PTO/SB/08a (09-08)
Approved for use through 10/31/2008. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 2

Complete if Known	
Application Number	~~10/972,588~~ 10/972,588
Filing Date	October 25, 2004
First Named Inventor	Scott L. Michaelis
Art Unit	2114
Examiner Name	Le, Dieu Minh T.
Attorney Docket Number	200404973-1

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number Number-Kind Code[2] *(if known)*	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
/M.L./		US- 2002/0152420	10-17-2002	Chaudhry et al.	
		US- 2003/0051190	03-13-2003	Marisetty et al.	
		US- 2003/0135711	07-17-2003	Shoemaker et al.	
		US- 2004/0078651	04-22-2004	Stafford et al.	
		US- 2005/0240811	10-27-2005	Stafford et al.	
		US- 2006/0248684	11-02-2006	Stafford	
		US- 2006/0085677	04-20-2006	Stafford et al.	
		US- 2006/0248384	11-9-2006	Vosbikian et al.	
		US- 4,358,823	11-9-1982	McDonald et al.	
		US- 5,537,655	07-16-1996	Truong	
		US- 5,675,794	10-07-1997	Meredith	
		US- 5,991,900	11-23-1999	Garnett	
		US- 6,220,581	04-24-2001	Mueller	
		US- 6.263,452	07-17-2001	Jewett et al.	
		US- 6,438,687	08-20-2002	Klein	
		US- 7,225,355	05-29-2007	Yamazaki et al.	
		US- 2003/0126498	07-03-2003	Bigbee et al.	
		US- 2004/0078650	04-22-2004	Stafford et al.	
		US- 2005/0240806	10-27-2005	Bruckert et al.	

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3]-Number[4]-Kind Code[5] *(if known)*	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages Or Relevant Figures Appear	T[6]

Examiner Signature	/Dieu Minh Le/	Date Considered	01/31/2009

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional). [2] See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3] Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4] For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5]Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [6]Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

PTO/SB/08a (11-08)
Approved for use through 12/31/2008. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 2 of 2

Complete if Known	
Application Number	10/972,588
Filing Date	October 25, 2004
First Named Inventor	Scott L. Michaelis
Art Unit	2114
Examiner Name	Le, Dieu Minh T.
Attorney Docket Number	200404973-1

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number Number-Kind Code[2] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
/M.L./		US- 2006/0248684	11-09-2006	Vosbikian et al.	
		US- 5,896,523	04-20-1999	Bissett et al.	
		US- 6,141,770	10-31-2000	Fuchs et al.	
		US- 6,675,324	01-06-2004	Marisetty et al.	
		US- 6,948,092	09-20-2005	Kondo et al.	
		US- 7,003,691	02-21-2006	Stafford et al.	
		US- 7,020,800	03-28-2006	Fu et al.	
		US- 7,134,047	11-07-2006	Quach	
		US- 7,155,721	12-26-2006	Stafford et al.	
		US- 7,237,144	06-26-2007	Stafford et al.	
		US- 7,191,292	03-13-2007	Chaudhry et al.	
		US- 2006/0136672	06-22-2006	Chaudhry et al.	
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3]-Number[4]-Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages Or Relevant Figures Appear	T[6]

Examiner Signature	/Dieu Minh Le/	Date Considered	01/31/2009

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional). [2] See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3] Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4] For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5] Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [6] Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	10972588	MICHAELIS ET AL.
	Examiner	Art Unit
	Dieu-Minh Le	2114

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	02/28/2008	09/11/2008	02/01/2009						
1	1	✓	=	=						
2	2	✓	=	=						
3	3	✓	=	=						
4	4	✓	=	=						
5	5	✓	=	=						
6	6	✓	=	=						
7	7	✓	=	=						
8	8	✓	=	=						
9	9	✓	=	=						
10	10	✓	=	=						
11	11	✓	=	=						
12	12	✓	=	=						
	13	✓	-	-						
	14	✓	-	-						
	15	✓	-	-						
13	16	✓	=	=						
14	17	✓	=	=						
15	18	✓	=	=						
16	19	✓	=	=						
17	20	✓	=	=						
18	21	✓	=	=						
19	22	✓	=	=						
20	23	✓	=	=						
21	24	=	=	=						
22	25	=	=	=						
23	26	=	=	=						
24	27	=	=	=						
25	28	=	=	=						
26	29	✓	=	=						
27	30	✓	=	=						

PTO/SB/08a (09-08)
Approved for use through 10/31/2008. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 2

Complete if Known	
Application Number	10/973,588
Filing Date	October 25, 2004
First Named Inventor	Scott L. Michaelis
Art Unit	2114
Examiner Name	Le, Dieu Minh T.
Attorney Docket Number	200404973-1

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number Number-Kind Code[2] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US- 2002/0152420	10-17-2002	Chaudhry et al.	
		US- 2003/0051190	03-13-2003	Marisetty et al.	
		US- 2003/0135711	07-17-2003	Shoemaker et al.	
		US- 2004/0078651	04-22-2004	Stafford et al.	
		US- 2005/0240811	10-27-2005	Stafford et al.	
		US- 2006/0248684	11-02-2006	Stafford	
		US- 2006/0085677	04-20-2006	Stafford et al.	
		US- 2006/0248384	11-9-2006	Vosbikian et al.	
		US- 4,358,823	11-9-1982	McDonald et al.	
		US- 5,537,655	07-16-1996	Truong	
		US- 5,675,794	10-07-1997	Meredith	
		US- 5,991,900	11-23-1999	Garnett	
		US- 6,220,581	04-24-2001	Mueller	
		US- 6,263,452	07-17-2001	Jewett et al.	
		US- 6,438,687	08-20-2002	Klein	
		US- 7,225,355	05-29-2007	Yamazaki et al.	
		US- 2003/0126498	07-03-2003	Bigbee et al.	
		US- 2004/0078650	04-22-2004	Stafford et al.	
		US- 2005/0240806	10-27-2005	Bruckert et al.	

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3]-Number[4]-Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages Or Relevant Figures Appear	T[6]

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional). [2] See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3] Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4] For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5] Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [6] Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

PTO/SB/08a (11-08)
Approved for use through 12/31/2008. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 2 of 2

Complete if Known	
Application Number	10/972,588
Filing Date	October 25, 2004
First Named Inventor	Scott L. Michaelis
Art Unit	2114
Examiner Name	Le, Dieu Minh T.
Attorney Docket Number	200404973-1

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number Number-Kind Code[2] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US- 2006/0248684	11-09-2006	Vosbikian et al.	
		US- 5,896,523	04-20-1999	Bissett et al.	
		US- 6,141,770	10-31-2000	Fuchs et al.	
		US- 6,675,324	01-06-2004	Marisetty et al.	
		US- 6,948,092	09-20-2005	Kondo et al.	
		US- 7,003,691	02-21-2006	Stafford et al.	
		US- 7,020,800	03-28-2006	Fu et al.	
		US- 7,134,047	11-07-2006	Quach	
		US- 7,155,721	12-26-2006	Stafford et al.	
		US- 7,237,144	06-26-2007	Stafford et al.	
		US- 7,191,292	03-13-2007	Chaudhry et al.	
		US- 2006/0136672	06-22-2006	Chaudhry et al.	
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3]-Number[4]-Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages Or Relevant Figures Appear	T[6]

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional). [2] See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3] Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4] For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5] Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [6] Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

Electronic Patent Application Fee Transmittal

Application Number:	10972588
Filing Date:	25-Oct-2004
Title of Invention:	SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR
First Named Inventor/Applicant Name:	Scott L. Michaelis
Filer:	Robert Wayne Nelson/Kalyn Black
Attorney Docket Number:	200404973-1

Filed as Large Entity

Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
Request for continued examination	1801	1	810	810
		Total in USD ($)		**810**

Electronic Acknowledgement Receipt

EFS ID:	4419258
Application Number:	10972588
International Application Number:	
Confirmation Number:	1079
Title of Invention:	SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR
First Named Inventor/Applicant Name:	Scott L. Michaelis
Customer Number:	22879
Filer:	Robert Wayne Nelson/Kalyn Black
Filer Authorized By:	Robert Wayne Nelson
Attorney Docket Number:	200404973-1
Receipt Date:	09-DEC-2008
Filing Date:	25-OCT-2004
Time Stamp:	11:59:52
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Miscellaneous Incoming Letter	E-RCE.pdf	573164 25301ca0575b4c0ed842f83c27b438ed7dcea041	no	2

Warnings:	
Information:	

2	Information Disclosure Statement (IDS) Filed (SB/08)	200404973-1_IDS.pdf	114650 ec317c6c67198344843aa986ae9dd828687a2cbd	no	2
Warnings:					
Information:					
This is not an USPTO supplied IDS fillable form					
3	Fee Worksheet (PTO-06)	fee-info.pdf	30795 b925711c38507d46e3baad462e42dc17bb2f4478	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	718609		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 200404973-1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): Scott L. Michaelis

Confirmation No.: 1079

Application No.: 10/972,588

Examiner: Le, Dieu Minh T.

Filing Date: October 25, 2004

Group Art Unit: 2114

Title: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

Mail Stop RCE
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

REQUEST FOR CONTINUED EXAMINATION (RCE) 37 CFR 1.114

Subsection (b) of 35 U.S.C. 132, effective on May 29, 2000, provides for continued examination of a utiltity or plant application filed on or after June 8, 1995.
See The American Inventors Protection Act of 1999 (AIPA).

This is a Request for Continued Examination (RCE) under CFR 1.114 of the above-identified application.

NOTE: 37 CFR 1.114 is effective on May 20, 2000. If the above- application was filed prior to May 29, 2000, applicant may wish to consider filing a continued prosecution application (CPA) under CFR 1.53(d) (PTO/SB/29) instead of a RCE to be eligible for the patent term adjustment provisions of the AIPA. See Changes to Application Examination and Provisional Application Practice, Interim Rule, 65 Fed. Reg. 14865 (Mar. 20, 2000), 1233 off. Gaz. Pat. Office

Submission under 37 CFR 1.114

☐ Previously submitted

☐ Consider the amendment(s)/reply under 37 CFR 1.116 previously filed on _____.
(Any unentered amendment(s) referred to above will be entered).

☐ Consider the arguments in the Appeal Brief or Reply Brief prevously filed on _____.

☐ Other _____

☒ Enclosed

☐ Amendment/Reply

☐ Affidavit(s)/Declaration(s)

☒ Information Disclosure Statement (IDS)

☐ Other _____

Miscellaneous

☐ Suspension of action is requested under 37 CFR 1.103(c) for a period of _____ months.
The fee for this Suspension is (37 CFR 1.17(i)) $130.00

☐ Other _____

CONTINUED EXAMINATION TRANSMITTAL (RCE) (37 CFR 1.114) (continued)	PATENT APPLICATION
	ATTORNEY DOCKET NO. 200404973-1

☒ RCE filing fee $810.00

☐ A Petition for Extension of Time

☐ 1st Month $130 ☐ 2nd Month $490 ☐ 3rd Month $1110 ☐ 4th Month $1730

☒ Please charge to Deposit Account 08-2025 the sum of $ 810 . At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that my regulate fees.

Respectfully submitted,

Scott L. Michaelis

By: /Robert W. Nelson/

Robert W. Nelson, Esq.

Attorney/Agent for Applicant(s)

Reg No. : 37,898

Date : December 9, 2008

Telephone : (303) 298-9888

Document code: WFEE

United States Patent and Trademark Office
Sales Receipt for Accounting Date: 08/03/2009

ABOARDLE SALE #00000047 Mailroom Dt: 12/09/2008 082025 10972588
01 FC : 1801 810.00 DA

FORM PTO-1449 LIST OF PATENTS AND PUBLICATIONS FOR APPLICANT'S INFORMATION DISCLOSURE STATEMENT (Use several sheets if necessary)	ATTY. DOCKET NO. 200404973-1	APPLICATION NO. 10/972,588	CONFIRMATION NO.
	APPLICANT Scott L. Michaelis et al.		
	FILING DATE	GROUP 2114	

REFERENCE DESIGNATION **U.S. PATENT DOCUMENTS**

EXAMINER INITIAL		DOCUMENT NUMBER	PUBLICATION DATE	NAME	Pages, Columns, Lines Where Relevant Passages or Figures Appear
AML	1A	5,226,152	07/06/1993	Klug et al.	
	1B	5,249,188	09/28/1993	McDonald	
	1C	5,751,932	05/12/1998	Horst et al.	
	1D	5,758,058	05/26/1998	Milburn	
	1E	5,764,660	06/09/1998	Mohat	
	1F	6,065,135	05/16/2000	Marshall et al.	
	1G	6,148,348	11/14/2000	Garnett et al.	
	1H	6,473,869	10/29/2002	Bissett et al.	
	1I	6,560,682	05/06/2003	Miller et al.	
✓	1J	6,604,177	08/05/2003	Kondo et al.	
AML	1K	6,615,366	09/02/2003	Grochowski et al.	

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	PUBLICATION DATE	NAME OF PATENTEE OR APPLICANT	Pages/Columns/Lines Where Relevant Passages/Figures Appear	Check if Translation attached
	1L					
	1M					
	1N					
	1O					
	1P					

OTHER REFERENCES (Including Author, Title, Date, Pertinent Pages, etc.)

AML	1Q	10/972,796 ~~Attorney Docket No. 200404974-1~~, filed concurrently herewith, entitled "SYSTEM AND METHOD FOR MAINTAINING IN A MULTI-PROCESSOR SYSTEM A SPARE PROCESSOR THAT IS IN LOCKSTEP FOR USE IN RECOVERING FROM LOSS OF LOCKSTEP FOR ANOTHER PROCESSOR"
AML	1R	10/973,927 ~~Attorney Docket No. 200404944-1~~, filed concurrently herewith, entitled "SYSTEM AND METHOD FOR ESTABLISHING A SPARE PROCESSOR FOR RECOVERING FROM LOSS OF LOCKSTEP IN A BOOT PROCESSOR"
AML	1S	10/972,835 ~~Attorney Docket No. 200404946-1~~, filed concurrently herewith, entitled "SYSTEM AND METHOD FOR USING INFORMATION RELATING TO A DETECTED LOSS OF LOCKSTEP FOR DETERMINING A RESPONSIVE ACTION"

PAP 11/12/07

EXAMINER	DATE CONSIDERED
/David Lade/	8/29/07

Rev 05/04 (PTO1449)

FORM PTO-1449 LIST OF PATENTS AND PUBLICATIONS FOR APPLICANT'S INFORMATION DISCLOSURE STATEMENT (Use several sheets if necessary)	ATTY. DOCKET NO. 200404973-1	APPLICATION NO. 10/972,588	CONFIRMATION NO.
	APPLICANT Scott L. Michaelis et al.		
	FILING DATE	GROUP 2114	

REFERENCE DESIGNATION **U.S. PATENT DOCUMENTS**

EXAMINER INITIAL		DOCUMENT NUMBER	PUBLICATION DATE	NAME	Pages, Columns, Lines Where Relevant Passages or Figures Appear
DM	2A	6,625,749	09/23/2003	Quach	
	2B	6,687,851	02/03/2004	Somers et al.	
	2C	6,754,787	06/22/2004	Miller et al.	
	2D	2002/0144177	10/03/2002	Kondo et al.	
	2E	2003/0070050	04/10/2003	Miller et al.	
	2F	2004/0006722	01/08/2004	Safford	
✓	2G	2004/0019771	01/29/2004	Quach	
DM	2H	2004/0153857	08/05/2004	Yamazaki et al.	
	2I				
	2J				
	2K				

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	PUBLICATION DATE	NAME OF PATENTEE OR APPLICANT	Pages/Columns/Lines Where Relevant Passages/Figures Appear	Check if Translation attached
	2L					
	2M					
	2N					
	2O					
	2P					

OTHER REFERENCES (including Author, Title, Date, Pertinent Pages, etc.)

PAP 11/12/08

DM	2Q	~~Attorney Docket No. 200404941-1~~ 10/973,076, filed concurrently herewith, entitled "SYSTEM AND METHOD FOR PROVIDING FIRMWARE RECOVERABLE LOCKSTEP PROTECTION"
DM	2R	~~Attorney Docket No. 200404943-1~~ 10/973,004, filed concurrently herewith, entitled "SYSTEM AND METHOD FOR CONFIGURING LOCKSTEP MODE OF A PRCESSOR MODULE"
DM	2S	~~Attorney Docket No. 200404971-1~~ 10/973,003, filed concurrently herewith, entitled "SYSTEM AND METHOD FOR REESTABLISHING LOCKSTEP FOR A PROCESSOR MODULE FOR WHICH LOSS OF LOCKSTEP IS DETECTED"

EXAMINER	DATE CONSIDERED
David Ton	8/29/07

Rev 05/04 (PTO1449)

FORM PTO-1449 LIST OF PATENTS AND PUBLICATIONS FOR APPLICANT'S INFORMATION DISCLOSURE STATEMENT (Use several sheets if necessary)	ATTY. DOCKET NO. 200404973-1	APPLICATION NO. 10/972,588	CONFIRMATION NO.
	APPLICANT Scott L. Michaelis et al.		
	FILING DATE	GROUP 2114	

REFERENCE DESIGNATION **U.S. PATENT DOCUMENTS**

EXAMINER INITIAL		DOCUMENT NUMBER	PUBLICATION DATE	NAME	Pages, Columns, Lines Where Relevant Passages or Figures Appear
	3A				
	3B				
	3C				
	3D				
	3E				
	3F				
	3G				
	3H				
	3I				
	3J				
	3K				

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	PUBLICATION DATE	NAME OF PATENTEE OR APPLICANT	Pages/Columns/Lines Where Relevant Passages/Figures Appear	Check if Translation attached
	3L					
	3M					
	3N					
	3O					
	3P					

OTHER REFERENCES (Including Author, Title, Date, Pertinent Pages, etc.)

AM	3Q	Attorney Docket No. 200404970-1, filed concurrently herewith, entitled "SYSTEM AND METHOD FOR SYSTEM FIRMWARE CAUSING AN OPERATING SYSTEM TO IDLE A PROCESSOR" 10/972,888
AM	3R	Attorney Docket No. 200404972-1, filed concurrently herewith, entitled "SYSTEM AND METHOD FOR REINTRODUCING A PROCESSOR MODULE TO AN OPERATING SYSTEM AFTER LOCKSTEP RECOVERY" 10/973,075
	3S	

PAP 11/12/08

EXAMINER	DATE CONSIDERED
[signature]	8/29/07

Rev 05/04 (PTO1449)

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

NOTICE OF ALLOWANCE AND FEE(S) DUE

22879 7590 09/23/2008

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

EXAMINER	
LE, DIEU MINH T	
ART UNIT	PAPER NUMBER
2114	

DATE MAILED: 09/23/2008

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/972,588	10/25/2004	Scott L. Michaelis	200404973-1	1079

TITLE OF INVENTION: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

APPLN. TYPE	SMALL ENTITY	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	$1440	$300	$0	$1740	12/23/2008

THE APPLICATION IDENTIFIED ABOVE HAS BEEN EXAMINED AND IS ALLOWED FOR ISSUANCE AS A PATENT. PROSECUTION ON THE MERITS IS CLOSED. THIS NOTICE OF ALLOWANCE IS NOT A GRANT OF PATENT RIGHTS. THIS APPLICATION IS SUBJECT TO WITHDRAWAL FROM ISSUE AT THE INITIATIVE OF THE OFFICE OR UPON PETITION BY THE APPLICANT. SEE 37 CFR 1.313 AND MPEP 1308.

THE ISSUE FEE AND PUBLICATION FEE (IF REQUIRED) MUST BE PAID WITHIN THREE MONTHS FROM THE MAILING DATE OF THIS NOTICE OR THIS APPLICATION SHALL BE REGARDED AS ABANDONED. THIS STATUTORY PERIOD CANNOT BE EXTENDED. SEE 35 U.S.C. 151. THE ISSUE FEE DUE INDICATED ABOVE DOES NOT REFLECT A CREDIT FOR ANY PREVIOUSLY PAID ISSUE FEE IN THIS APPLICATION. IF AN ISSUE FEE HAS PREVIOUSLY BEEN PAID IN THIS APPLICATION (AS SHOWN ABOVE), THE RETURN OF PART B OF THIS FORM WILL BE CONSIDERED A REQUEST TO REAPPLY THE PREVIOUSLY PAID ISSUE FEE TOWARD THE ISSUE FEE NOW DUE.

HOW TO REPLY TO THIS NOTICE:

I. Review the SMALL ENTITY status shown above.

If the SMALL ENTITY is shown as YES, verify your current SMALL ENTITY status:

A. If the status is the same, pay the TOTAL FEE(S) DUE shown above.

B. If the status above is to be removed, check box 5b on Part B - Fee(s) Transmittal and pay the PUBLICATION FEE (if required) and twice the amount of the ISSUE FEE shown above, or

If the SMALL ENTITY is shown as NO:

A. Pay TOTAL FEE(S) DUE shown above, or

B. If applicant claimed SMALL ENTITY status before, or is now claiming SMALL ENTITY status, check box 5a on Part B - Fee(s) Transmittal and pay the PUBLICATION FEE (if required) and 1/2 the ISSUE FEE shown above.

II. PART B - FEE(S) TRANSMITTAL, or its equivalent, must be completed and returned to the United States Patent and Trademark Office (USPTO) with your ISSUE FEE and PUBLICATION FEE (if required). If you are charging the fee(s) to your deposit account, section "4b" of Part B - Fee(s) Transmittal should be completed and an extra copy of the form should be submitted. If an equivalent of Part B is filed, a request to reapply a previously paid issue fee must be clearly made, and delays in processing may occur due to the difficulty in recognizing the paper as an equivalent of Part B.

III. All communications regarding this application must give the application number. Please direct all communications prior to issuance to Mail Stop ISSUE FEE unless advised to the contrary.

IMPORTANT REMINDER: Utility patents issuing on applications filed on or after Dec. 12, 1980 may require payment of maintenance fees. It is patentee's responsibility to ensure timely payment of maintenance fees when due.

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail **Mail Stop ISSUE FEE**
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (571)-273-2885

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

22879 7590 09/23/2008

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (571) 273-2885, on the date indicated below.

(Depositor's name)
(Signature)
(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/972,588	10/25/2004	Scott L. Michaelis	200404973-1	1079

TITLE OF INVENTION: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

APPLN. TYPE	SMALL ENTITY	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	$1440	$300	$0	$1740	12/23/2008

EXAMINER	ART UNIT	CLASS-SUBCLASS
LE, DIEU MINH T	2114	714-012000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).

❑ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.

❑ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. **Use of a Customer Number is required.**

2. For printing on the patent front page, list

(1) the names of up to 3 registered patent attorneys or agents OR, alternatively,

(2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.

1 _____

2 _____

3 _____

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE (B) RESIDENCE: (CITY and STATE OR COUNTRY)

Please check the appropriate assignee category or categories (will not be printed on the patent) : ❑ Individual ❑ Corporation or other private group entity ❑ Government

4a. The following fee(s) are submitted:

❑ Issue Fee

❑ Publication Fee (No small entity discount permitted)

❑ Advance Order - # of Copies _____

4b. Payment of Fee(s): **(Please first reapply any previously paid issue fee shown above)**

❑ A check is enclosed.

❑ Payment by credit card. Form PTO-2038 is attached.

❑ The Director is hereby authorized to charge the required fee(s), any deficiency, or credit any overpayment, to Deposit Account Number _____ (enclose an extra copy of this form).

5. **Change in Entity Status** (from status indicated above)

❑ a. Applicant claims SMALL ENTITY status. See 37 CFR 1.27.

❑ b. Applicant is no longer claiming SMALL ENTITY status. See 37 CFR 1.27(g)(2).

NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

Authorized Signature _____ Date _____

Typed or printed name _____ Registration No. _____

This collection of information is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, Virginia 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PTOL-85 (Rev. 08/07) Approved for use through 08/31/2010. OMB 0651-0033 U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/972,588	10/25/2004	Scott L. Michaelis	200404973-1	1079

22879 7590 09/23/2008

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

EXAMINER	
LE, DIEU MINH T	
ART UNIT	PAPER NUMBER
2114	

DATE MAILED: 09/23/2008

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)

(application filed on or after May 29, 2000)

The Patent Term Adjustment to date is 529 day(s). If the issue fee is paid on the date that is three months after the mailing date of this notice and the patent issues on the Tuesday before the date that is 28 weeks (six and a half months) after the mailing date of this notice, the Patent Term Adjustment will be 529 day(s).

If a Continued Prosecution Application (CPA) was filed in the above-identified application, the filing date that determines Patent Term Adjustment is the filing date of the most recent CPA.

Applicant will be able to obtain more detailed information by accessing the Patent Application Information Retrieval (PAIR) WEB site (http://pair.uspto.gov).

Any questions regarding the Patent Term Extension or Adjustment determination should be directed to the Office of Patent Legal Administration at (571)-272-7702. Questions relating to issue and publication fee payments should be directed to the Customer Service Center of the Office of Patent Publication at 1-(888)-786-0101 or (571)-272-4200.

Notice of Allowability	Application No.	Applicant(s)	
	10/972,588	MICHAELIS ET AL.	
	Examiner	Art Unit	
	Dieu-Minh Le	2114	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to *the interview on 9/5/08 and communications filed 6/5/08 and 9/10/08*.

2. ☒ The allowed claim(s) is/are *1-12 and 16-30 now as [1-27]*.

3. ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
   a) ☐ All b) ☐ Some* c) ☐ None of the:
      1. ☐ Certified copies of the priority documents have been received.
      2. ☐ Certified copies of the priority documents have been received in Application No. _____ .
      3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

   * Certified copies not received: _____.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application.
THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.

4. ☐ A SUBSTITUTE OATH OR DECLARATION must be submitted. Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL PATENT APPLICATION (PTO-152) which gives reason(s) why the oath or declaration is deficient.

5. ☐ CORRECTED DRAWINGS (as "replacement sheets") must be submitted.
   (a) ☐ including changes required by the Notice of Draftsperson's Patent Drawing Review (PTO-948) attached
      1) ☐ hereto or 2) ☐ to Paper No./Mail Date _____.
   (b) ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date _____.

   Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).

6. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1. ☐ Notice of References Cited (PTO-892)
2. ☐ Notice of Draftperson's Patent Drawing Review (PTO-948)
3. ☐ Information Disclosure Statements (PTO/SB/08), Paper No./Mail Date _____
4. ☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
5. ☐ Notice of Informal Patent Application
6. ☒ Interview Summary (PTO-413), Paper No./Mail Date _____ .
7. ☐ Examiner's Amendment/Comment
8. ☐ Examiner's Statement of Reasons for Allowance
9. ☐ Other _____.

/Dieu-Minh Le/
Primary Examiner, Art Unit 2114

Interview Summary	Application No.	Applicant(s)	
	10/972,588	MICHAELIS ET AL.	
	Examiner	Art Unit	
	Dieu-Minh Le	2114	

All participants (applicant, applicant's representative, PTO personnel):

(1) *Dieu-Minh Le*. (3)_____.

(2) *Augustus Winfield [Reg. No. 34,046]*. (4)_____.

Date of Interview: *05 September 2008*.

Type: a)☒ Telephonic b)☐ Video Conference
c)☐ Personal [copy given to: 1)☐ applicant 2)☐ applicant's representative]

Exhibit shown or demonstration conducted: d)☐ Yes e)☒ No.
If Yes, brief description: _____.

Claim(s) discussed: *1-30*.

Identification of prior art discussed: *Safford, Fox and Quach*.

Agreement with respect to the claims f)☒ was reached. g)☐ was not reached. h)☐ N/A.

Substance of Interview including description of the general nature of what was agreed to if an agreement was reached, or any other comments: *Main issues discussed were the claimed invention over the above prior arts, double patenting rejection, 35 USC 101 for claims 29-30 and the proposed amended claims 1-30. Applicant's representative agreed to file TD and supplemental amendment according to communications filed 9/10/2008*.

(A fuller description, if necessary, and a copy of the amendments which the examiner agreed would render the claims allowable, if available, must be attached. Also, where no copy of the amendments that would render the claims allowable is available, a summary thereof must be attached.)

THE FORMAL WRITTEN REPLY TO THE LAST OFFICE ACTION MUST INCLUDE THE SUBSTANCE OF THE INTERVIEW. (See MPEP Section 713.04). If a reply to the last Office action has already been filed, APPLICANT IS GIVEN A NON-EXTENDABLE PERIOD OF THE LONGER OF ONE MONTH OR THIRTY DAYS FROM THIS INTERVIEW DATE, OR THE MAILING DATE OF THIS INTERVIEW SUMMARY FORM, WHICHEVER IS LATER, TO FILE A STATEMENT OF THE SUBSTANCE OF THE INTERVIEW. See Summary of Record of Interview requirements on reverse side or on attached sheet.

/Dieu-Minh Le/ Primary Examiner, Art Unit 2114	

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	10972588	MICHAELIS ET AL.
	Examiner	Art Unit
	Dieu-Minh Le	2114

SEARCHED			
Class	Subclass	Date	Examiner
714	10, 11, 12, 13	9/4/08	dml
712	31	9/4/08	dml

SEARCH NOTES		
Search Notes	Date	Examiner
EAST search all files updated	9/4/08	dml
Inventor name search	9/4/08	dml
NPL search files [IEEE, Google Scholar]	9/11/08	dml

INTERFERENCE SEARCH			
Class	Subclass	Date	Examiner
714	10-13	9/4/08	dml
712	31	9/4/08	dml
	See Interference search data file		

	/Dieu-Minh Le/ Primary Examiner.Art Unit 2114

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

BIB DATA SHEET

CONFIRMATION NO. 1079

SERIAL NUMBER	FILING or 371(c) DATE RULE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.
10/972,588	10/25/2004	713	2114	200404973-1

APPLICANTS

Scott L. Michaelis, Plano, TX;
Anurupa Rajkumari, Round Rock, TX;
William B. McHardy, Fort Collins, CO;

** CONTINUING DATA ********** na dml

** FOREIGN APPLICATIONS **********

** IF REQUIRED, FOREIGN FILING LICENSE GRANTED ** na dml
01/14/2005

Foreign Priority claimed ☐ Yes ☑ No 35 USC 119(a-d) conditions met ☐ Yes ☑ No Verified and Acknowledged /DIEU-MINH T LE/ Examiner's Signature	☐ Met after Allowance dml Initials	STATE OR COUNTRY	SHEETS DRAWINGS	TOTAL CLAIMS	INDEPENDENT CLAIMS
		TX	4	30 27	4

ADDRESS

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400
UNITED STATES

TITLE

System and method for switching the role of boot processor to a spare processor responsive to detection of loss of lockstep in a boot processor

FILING FEE RECEIVED		
1058	FEES: Authority has been given in Paper No._____ to charge/credit DEPOSIT ACCOUNT No._____ for following:	☐ All Fees ☐ 1.16 Fees (Filing) ☐ 1.17 Fees (Processing Ext. of time) ☐ 1.18 Fees (Issue) ☐ Other _____ ☐ Credit

EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	804	(714/11).ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:19
L2	565	(714/10).ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:19
L3	371	(714/12).ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:19
L4	696	(714/13).ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:19
L5	381	(714/31).ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:19
L6	204	(loss lose los $3 lost$3) same (lockstep lock-step "lock step")	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:21
L7	1586	(boot$4 or initializ$4) adj processor	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:22
L8	52623	(switch$4 chang$4 failover fail-over "fail over") same (spare slave redundan$4 backup back-up "back up" second$4) same processor	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:22

L9	30192	(pair spare slave redundan $4 backup back-up "back up" second$4) adj2 processor	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:22
L10	82	(detect$3 track $3 monitor$3) same 6	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:23
L11	163830	without same ("shut$4 down" stop$4 idle interrupt $4) same (OS "operating system")	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:24
L12	32	10 and 8 and 9	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:29
L13	7	12 and 1	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:29
L14	3	12 and 2 and 11	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:30
L15	7	12 and (3 4 5) and 11	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:31
L16	22	12 and ("714"/ $.ccls.) and 11	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:38
L17	12	(switch$4 chang$4) same role same 7	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:39
L18	7	16 and 17	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/11 12:40

9/11/08 12:54:58 PM
C:\Documents and Settings\MLe\My Documents\EAST\Workspaces\10972588i.wsp

EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	40917	multi-processor "multi processor" (plural$3 adj processor)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 12:32
L2	1742	(boot$4 or initializ$4) adj processor	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 12:32
L3	94	L2 same 1	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 12:32
L4	16	(assign$4 giv$3 order$3 decid$3 mak$3) same role same 2	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 12:35
L5	2459	lockstep lock-step	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:00
L6	25133	(loss adj L5) LOL	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:00
L7	56	(detect$3 or track$4) adj L6	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:00

L8	13	3 same L7	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:00
L9	0	8 not 4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:01
L10	803	(714/11).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:01
L11	370	(714/12).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:01
L12	693	(714/13).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:01
L13	682	(714/10).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:01
L14	104	(712/31).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:01
L15	2237	L10 L11 L12 L13 L14	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:01

L16	803	(714/11).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:02
L17	289	spare adj processor	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:02
L18	15	L7 and L2 and L17	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:02
L19	221	(loss lose los$3 lost$3) same (lockstep lock-step "lock step")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:02
L20	92	(detect$3 track$3 monitor$3) same L19	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:02
L21	3	L20 and L16 and L18	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 13:02
L22	803	(714/11).ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/04 13:46
L23	370	(714/12).ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/04 13:46
L24	693	(714/13).ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/04 13:46
L25	563	(714/10).ccls.	US-PGPUB; USPAT; USOCR	OR	ON	2008/09/04 13:46
L26	15199	without same (shutt$3 turn-off power-off) same ((operating adj system) OS)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 14:09

L27	9	4 and 26	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 14:10
L28	0	27 not 8	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/09/04 14:12

9/4/08 2:14:58 PM
C:\Documents and Settings\MLe\My Documents\EAST\Workspaces\10972588b.wsp

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	10972588	MICHAELIS ET AL.
	Examiner	Art Unit
	Dieu-Minh Le	2114

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	02/28/2008	09/11/2008							
1	1	✓	=							
2	2	✓	=							
3	3	✓	=							
4	4	✓	=							
5	5	✓	=							
6	6	✓	=							
7	7	✓	=							
8	8	✓	=							
9	9	✓	=							
10	10	✓	=							
11	11	✓	=							
12	12	✓	=							
	13	✓	-							
	14	✓	-							
	15	✓	-							
13	16	✓	=							
14	17	✓	=							
15	18	✓	=							
16	19	✓	=							
17	20	✓	=							
18	21	✓	=							
19	22	✓	=							
20	23	✓	=							
21	24	=	=							
22	25	=	=							
23	26	=	=							
24	27	=	=							
25	28	=	=							
26	29	✓	=							
27	30	✓	=							

Issue Classification

Application/Control No.	Applicant(s)/Patent Under Reexamination
10972588	MICHAELIS ET AL.
Examiner	**Art Unit**
Dieu-Minh Le	2114

ORIGINAL		INTERNATIONAL CLASSIFICATION														
CLASS	SUBCLASS	CLAIMED					NON-CLAIMED									
714	12	G	0	6	F	11 / 00 (2006.01.01)										

CROSS REFERENCE(S)					
CLASS	SUBCLASS (ONE SUBCLASS PER BLOCK)				
714	11				

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original
1	1	14	17												
2	2	15	18												
3	3	16	19												
4	4	17	20												
5	5	18	21												
6	6	19	22												
7	7	20	23												
8	8	21	24												
9	9	22	25												
10	10	23	26												
11	11	24	27												
12	12	25	28												
	13	26	29												
	14	27	30												
	15														
13	16														

NONE		**Total Claims Allowed:**	
(Assistant Examiner)	(Date)	27	
/Dieu-Minh Le/ Primary Examiner.Art Unit 2114	09/12/2008	O.G. Print Claim(s)	O.G. Print Figure
(Primary Examiner)	(Date)	1	1

Application Number	Application/Control No.	Applicant(s)/Patent under Reexamination	
	10/972,588	MICHAELIS ET AL.	
		Le	2114

Document Code - DISQ	Internal Document – DO NOT MAIL

TERMINAL DISCLAIMER	☒ APPROVED 2 TD's	☐ DISAPPROVED
Date Filed : 9.10.08	This patent is subject to a Terminal Disclaimer	

Approved/Disapproved by: Argie

U.S. Patent and Trademark Office

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION NO.: 10/972,588 EXAMINER: Le, Dieu Minh T

FILED: 10/25/2004 ART UNIT: 2114

INVENTOR: Scott L. Michaelis *et al.* CONFIRMATION NO.: 1079

ATTORNEY DOCKET NUMBER 200404973-1

AMENDMENT IN RESPONSE TO PHONE INTERVIEW ON 09/05/2008

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Amendments to the claims are reflected in the listing of claims beginning on page 2 of this paper.

Remarks begin on page 7 of this paper.

1. (currently amended) A method comprising:

detecting loss of lockstep (LOL) for a processor in a multi-processor system;

determining that the processor for which said LOL is detected is assigned a role of boot processor; ~~and~~

changing a system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected, to a different value, which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor; and

changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to a spare processor without shutting down the system's operating system.

2. (original) The method of claim 1 wherein the spare processor is a hot spare that is held in idle until switched the role of boot processor.

3. (original) The method of claim 1 wherein the spare processor is an active spare.

4. (original) The method of claim 3 wherein said switching further comprises:

causing the system's operating system to idle the active spare.

5. (original) The method of claim 1 wherein the spare processor is a processor module that includes a master and slave processors that operate in lockstep.

6. (original) The method of claim 1 wherein said determining that the processor for which said LOL is detected is assigned the role of boot processor comprises:

accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor.

7. (original) The method of claim 1 further comprising:

determining the processor that is assigned the role of spare processor.

8. (original) The method of claim 7 wherein said determining the processor that is assigned the role of spare processor comprises:

accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of spare processor.

9. (original) The method of claim 1 wherein said determining step is performed by system firmware.

10. (original) The method of claim 1 wherein said switching step is performed by system firmware.

11. (original) The method of claim 1 wherein the processor for which said LOL is detected includes a lockstep pair of processors, and wherein said switching comprises:

determining one of the lockstep pair of processors that is not the cause of said LOL; and

copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to said spare processor.

12. (original) The method of claim 11 wherein said spare processor includes a lockstep pair of processors, and wherein said copying comprises:

copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to each of said lockstep pair of processors of said spare processor.

13. (canceled)

14. (canceled)

15. (canceled)

16. (currently amended) A method comprising:

system firmware determining that loss of lockstep (LOL) is detected for a

lockstep pair of processors that are assigned the role of boot processor in a system;

determining one of said lockstep pair of processors that is not the cause of said LOL;

copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to a spare processor; ~~and~~

changing a system identifier, used by the system's operating system to recognize the boot processor, of said lockstep pair of processors that are assigned the role of boot processor, to a different value, which causes the system's operating system to not recognize the lockstep pair of processors for which said LOL is detected as the boot processor; and

changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to said spare processor.

17. (original) The method of claim 16 wherein the spare processor is a hot spare that is held in idle until switched the role of boot processor.

18. (original) The method of claim 16 wherein the spare processor is an active spare.

19. (original) The method of claim 18 further comprising:

causing the system's operating system to idle the active spare.

20. (original) The method of claim 16 wherein said switching causes the system's operating system to recognize said spare processor as said boot processor without shutting down said operating system.

21. (original) The method of claim 16 wherein said determining step is performed by system firmware.

22. (original) The method of claim 16 wherein said copying step is performed by system firmware.

23. (original) The method of claim 16 wherein said switching step is performed by system firmware.

24. (original) A system comprising:

a plurality of processor modules that each include a master processor and a slave processor that operate in lockstep;

an operating system;

error detection logic operable to detect loss of lockstep (LOL) for at least one of said processor modules; and

system firmware operable, responsive to the detection of LOL for a first of said processor modules, to determine whether said first processor module is assigned a role of system boot processor;

wherein if determined that said first processor module is assigned the role of system boot processor, said system firmware is further operable to cause said operating system to recognize another processor module as system boot processor without shutting down said operating system.

25. (original) The system of claim 24 wherein the another processor module is a processor module assigned a role of spare processor.

26. (original) The system of claim 24 wherein the another processor module is held in idle until said system firmware causes said operating system to recognize said another processor module as said system boot processor.

27. (original) The system of claim 24 wherein the system firmware causes the operating system to idle the another processor module before causing said operating system to recognize said another processor module as said system boot processor.

28. (original) The system of claim 24 further comprising:

non-volatile data storage storing information that assigns the role of system boot processor to one of said plurality of processor modules.

29. (currently amended) A system comprising:

system firmware means for detecting loss of lockstep (LOL) for a processor in a multi-processor system;

system firmware means for determining whether the processor for which said

LOL is detected is assigned the role of boot processor; and

system firmware means for switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which said LOL is detected is assigned the role of boot processor.

30. (original) The system of claim 29 wherein said processor in said multi-processor system is an Itanium Processor Family (IPF) processor.

REMARKS

Claims 1, 16, and 29 have been amended. Claims 13-15 have been canceled. Claims 1-12 and 16-30 remain in the application. Reexamination and reconsideration is respectfully requested.

Double Patenting Rejections

Claims 1-30 have been rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1-32 of commonly owned U.S. application 10/973,077 and claims 1-26 of commonly owned U.S. application 10/972,796. Note that U.S. application 10/972,796 has issued as U.S. Patent Number 7,366,948. Terminal Disclaimers have been submitted separately.

Rejections Under 35 U.S.C. § 102

Claims 1-4, 7, 9-23, and 29 have been rejected under 35 U.S.C. § 102(e) as allegedly being anticipated by U.S. Patent 7,085,959 (Safford). In a phone interview on September 5, 2008, the examiner and the below signed attorney discussed amendments that might make the claims allowable. In response to that interview, claim 1 has been amended to incorporate the limitations of claims 13-15 and claims 13-15 have been canceled. Claim 16 has been amended to incorporate similar limitations as claim 1. In addition, claim 29 has been amended to add system firmware as additional structure for the means plus function claims.

In view of the above, it is believed all of the pending claims are now in condition for allowance.

Respectfully submitted,
KLAAS, LAW, O'MEARA & MALKIN, P.C.

By: /Augustus W. Winfield/

Augustus W. Winfield, Reg. No. 34,046
1999 Broadway, Suite 2225
Denver, Colorado 80202
Telephone: (303) 298-9888

Date: September 10, 2008

PTO/SB/25 (08-08)
Approved for use through 09/30/2008. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TERMINAL DISCLAIMER TO OBVIATE A PROVISIONAL DOUBLE PATENTING REJECTION OVER A PENDING "REFERENCE" APPLICATION

Docket Number (Optional): 200404973-1

In re Application of: Scott L. Michaelis et al.

Application No.: 10/972,588

Filed: October 25, 2004

For: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

The owner*, Hewlett-Packard Development Company, L.P., of 100 percent interest in the instant application hereby disclaims, except as provided below, the terminal part of the statutory term of any patent granted on the instant application which would extend beyond the expiration date of the full statutory term of any patent granted on pending **reference** Application Number 10/973,077, filed on October 25, 2004, as such term is defined in 35 U.S.C. 154 and 173, and as the term of any patent granted on said **reference** application may be shortened by any terminal disclaimer filed prior to the grant of any patent on the pending **reference** application. The owner hereby agrees that any patent so granted on the instant application shall be enforceable only for and during such period that it and any patent granted on the **reference** application are commonly owned. This agreement runs with any patent granted on the instant application and is binding upon the grantee, its successors or assigns.

In making the above disclaimer, the owner does not disclaim the terminal part of any patent granted on the instant application that would extend to the expiration date of the full statutory term as defined in 35 U.S.C. 154 and 173 of any patent granted on said **reference** application, "as the term of any patent granted on said **reference** application may be shortened by any terminal disclaimer filed prior to the grant of any patent on the pending **reference** application," in the event that: any such patent granted on the pending **reference** application: expires for failure to pay a maintenance fee, is held unenforceable, is found invalid by a court of competent jurisdiction, is statutorily disclaimed in whole or terminally disclaimed under 37 CFR 1.321, has all claims canceled by a reexamination certificate, is reissued, or is in any manner terminated prior to the expiration of its full statutory term as shortened by any terminal disclaimer filed prior to its grant.

Check either box 1 or 2 below, if appropriate.

1. ☐ For submissions on behalf of a business/organization (e.g., corporation, partnership, university, government agency, etc.), the undersigned is empowered to act on behalf of the business/organization.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

2. ☒ The undersigned is an attorney or agent of record. Reg. No. 44,395

/Steven L. Webb/ — Signature — 9/10/08 — Date

Steven L. Webb — Typed or printed name

(970) 898-3451 — Telephone Number

☐ Terminal disclaimer fee under 37 CFR 1.20(d) is included.

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

*Statement under 37 CFR 3.73(b) is required if terminal disclaimer is signed by the assignee (owner). Form PTO/SB/96 may be used for making this statement. See MPEP § 324.

This collection of information is required by 37 CFR 1.321. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

PTO/SB/26 (08-08)
Approved for use through 09/30/2008. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TERMINAL DISCLAIMER TO OBVIATE A DOUBLE PATENTING REJECTION OVER A "PRIOR" PATENT	Docket Number (Optional) 200404973-1

In re Application of: Scott L. Michaelis et al.

Application No.: 10/973,077

Filed: October 25, 2004

For: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT

HEWLETT-PACKARD DEVELOPMENT

The owner*, COMPANY, L.P., of 100 percent interest in the instant application hereby disclaims, except as provided below, the terminal part of the statutory term of any patent granted on the instant application which would extend beyond the expiration date of the full statutory term **prior patent** No. 7,366,948 as the term of said prior patent is defined in 35 U.S.C. 154 and 173, and as the term of said **prior patent** is presently shortened by any terminal disclaimer. The owner hereby agrees that any patent so granted on the instant application shall be enforceable only for and during such period that it and the **prior patent** are commonly owned. This agreement runs with any patent granted on the instant application and is binding upon the grantee, its successors or assigns.

In making the above disclaimer, the owner does not disclaim the terminal part of the term of any patent granted on the instant application that would extend to the expiration date of the full statutory term as defined in 35 U.S.C. 154 and 173 of the **prior patent**, "as the term of said **prior patent** is presently shortened by any terminal disclaimer," in the event that said **prior patent** later:

expires for failure to pay a maintenance fee;
is held unenforceable;
is found invalid by a court of competent jurisdiction;
is statutorily disclaimed in whole or terminally disclaimed under 37 CFR 1.321;
has all claims canceled by a reexamination certificate;
is reissued; or
is in any manner terminated prior to the expiration of its full statutory term as presently shortened by any terminal disclaimer.

Check either box 1 or 2 below, if appropriate.

1. ☐ For submissions on behalf of a business/organization (e.g., corporation, partnership, university, government agency, etc.), the undersigned is empowered to act on behalf of the business/organization.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

2. ☒ The undersigned is an attorney or agent of record. Reg. No. 44,395

/Steven L. Webb/ — Signature

9/10/08 — Date

Steven L. Webb — Typed or printed name

(970) 898-3451 — Telephone Number

☐ Terminal disclaimer fee under 37 CFR 1.20(d) included.

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

*Statement under 37 CFR 3.73(b) is required if terminal disclaimer is signed by the assignee (owner).
Form PTO/SB/96 may be used for making this certification. See MPEP § 324.

This collection of information is required by 37 CFR 1.321. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Electronic Patent Application Fee Transmittal

Application Number:	10972588
Filing Date:	25-Oct-2004
Title of Invention:	System and method for switching the role of boot processor to a spare processor responsive to detection of loss of lockstep in a boot processor
First Named Inventor/Applicant Name:	Scott L. Michaelis
Filer:	Augustus W. Winfield/Joy Reinhart
Attorney Docket Number:	200404973-1

Filed as Large Entity

Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
Statutory disclaimer	1814	2	130	260
		Total in USD ($)		**260**

Electronic Acknowledgement Receipt

EFS ID:	3917555
Application Number:	10972588
International Application Number:	
Confirmation Number:	1079
Title of Invention:	System and method for switching the role of boot processor to a spare processor responsive to detection of loss of lockstep in a boot processor
First Named Inventor/Applicant Name:	Scott L. Michaelis
Customer Number:	22879
Filer:	Augustus W. Winfield/Joy Reinhart
Filer Authorized By:	Augustus W. Winfield
Attorney Docket Number:	200404973-1
Receipt Date:	10-SEP-2008
Filing Date:	25-OCT-2004
Time Stamp:	13:13:46
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Miscellaneous Incoming Letter	Transmittal.pdf	128086 2ae2308aebb35b2de7eb2aceae54924efe1de809	no	1

Warnings:	
Information:	

2	Amendment/Req. Reconsideration-After Non-Final Reject	Amendment.pdf	51413 05a4b86b73959ea4983babbfeb96a0570c6603d2	no	1
Warnings:					
Information:					
3	Claims	Claims.pdf	554753 2608309fb2d753f27a64682bc4dffc02dd8d185d	no	5
Warnings:					
Information:					
4	Applicant Arguments/Remarks Made in an Amendment	Remarks.pdf	148620 e1a0f736879bf84da736d63b377792ac08d932a5	no	2
Warnings:					
Information:					
5	Terminal Disclaimer Filed	TerminalDisclaimer1.pdf	219620 fa252426323b946951b442a8e35a78732987d122	no	1
Warnings:					
Information:					
6	Terminal Disclaimer Filed	TerminalDisclaimer2.pdf	230698 52cd9abb2b0fb12e83a4ca43441922826 5f1a5e8	no	1
Warnings:					
Information:					
7	Fee Worksheet (PTO-06)	fee-info.pdf	30278 d883bca5415ffd0ab07488971b98f3b4f1b6ab05	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	1363468		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 200404973-1

Inventor(s): Scott L. Michaelis et al.

Confirmation No.: 1079

Application No.: 10/972,588

Examiner: Le, Dieu Minh T.

Filing Date: October 25, 2004

Group Art Unit: 2114

Title: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

Mail Stop Amendment
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL LETTER FOR RESPONSE/AMENDMENT

Transmitted herewith is/are the following in the above-identified application:

- [X] Response/Amendment
- [ ] New fee as calculated below
- [X] No additional fee
- [X] Other Terminal Disclaimer (2)
- [ ] Petition to extend time to respond
- [ ] Supplemental Declaration

Fee$ 260

CLAIMS AS AMENDED BY OTHER THAN A SMALL ENTITY						
(1) FOR	(2) CLAIMS REMAINING AFTER AMENDMENT	(3) NUMBER EXTRA	(4) HIGHEST NUMBER PREVIOUSLY PAID FOR	(5) PRESENT EXTRA	(6) RATE	(7) ADDITIONAL FEES
TOTAL CLAIMS	27	MINUS	30	= 0	X $50	$ 0
INDEP. CLAIMS	4	MINUS	4	= 0	X $210	$ 0
☐ FIRST PRESENTATION OF A MULTIPLE DEPENDENT CLAIM					+ $370	$ 0
EXTENSION FEE	☐ 1st Month $120	☐ 2nd Month $460	☐ 3rd Month $1050	☐ 4th Month $1640		$ 0
					OTHER FEES	$ 260
					TOTAL ADDITIONAL FEE FOR THIS AMENDMENT	$ 260

Charge $ 260 to Deposit Account 08-2025. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees. A duplicate copy of this sheet is enclosed.

Respectfully submitted,

Scott L. Michaelis et al.

By: /Augustus W. Winfield/

Augustus W. Winfield

Attorney/Agent for Applicant(s)

Reg No. : 34,046

Date : September 10, 2008

Telephone : (303) 298-9888

Rev 10/07 (E-TransAmd)

PTO/SB/06 (07-06)
Approved for use through 1/31/2007. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD

Substitute for Form PTO-875

Application or Docket Number	Filing Date	☐ To be Mailed
10/972,588	10/25/2004	

APPLICATION AS FILED – PART I

FOR	(Column 1) NUMBER FILED	(Column 2) NUMBER EXTRA	SMALL ENTITY ☐ RATE ($)	SMALL ENTITY FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY FEE ($)
☐ BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A	N/A			N/A	
☐ SEARCH FEE (37 CFR 1.16(k), (i), or (m))	N/A	N/A	N/A			N/A	
☐ EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A	N/A			N/A	
TOTAL CLAIMS (37 CFR 1.16(i))	minus 20 =	*	X $ =		OR	X $ =	
INDEPENDENT CLAIMS (37 CFR 1.16(h))	minus 3 =	*	X $ =			X $ =	
☐ APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).						
☐ MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))							
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL			TOTAL	

APPLICATION AS AMENDED – PART II

AMENDMENT	09/10/2008	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY ADDITIONAL FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY ADDITIONAL FEE ($)
	Total (37 CFR 1.16(i))	* 27	Minus	** 30	= 0	X $ =		OR	X $50=	0
	Independent (37 CFR 1.16(h))	* 4	Minus	***4	= 0	X $ =		OR	X $210=	0
	☐ Application Size Fee (37 CFR 1.16(s))									
	☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
						TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	0

AMENDMENT		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)		RATE ($)	ADDITIONAL FEE ($)
	Total (37 CFR 1.16(i))	*	Minus	**	=	X $ =		OR	X $ =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	X $ =		OR	X $ =	
	☐ Application Size Fee (37 CFR 1.16(s))									
	☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
						TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.

** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".

*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".

The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

Legal Instrument Examiner:
/DALE A. HALL/

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Electronic Acknowledgement Receipt

EFS ID:	3408883
Application Number:	10972588
International Application Number:	
Confirmation Number:	1079
Title of Invention:	System and method for switching the role of boot processor to a spare processor responsive to detection of loss of lockstep in a boot processor
First Named Inventor/Applicant Name:	Scott L. Michaelis
Customer Number:	22879
Filer:	Jay Kevin Malkin/Joy Reinhart
Filer Authorized By:	Jay Kevin Malkin
Attorney Docket Number:	200404973-1
Receipt Date:	05-JUN-2008
Filing Date:	25-OCT-2004
Time Stamp:	14:18:51
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes) /Message Digest	Multi Part /.zip	Pages (if appl.)
1	Miscellaneous Incoming Letter	Transmittal.pdf	76781 2dfde6840358fa209001c6d53fa6e3912943448d	no	1

Warnings:	
Information:	

2	Applicant Arguments/Remarks Made in an Amendment	Remarks.pdf	188315 284f8d4deadb7a2996c027513b04190ad650f37b	no	5

Warnings:	
Information:	
Total Files Size (in bytes):	265096

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION NO.: 10/972,588 EXAMINER: Le, Dieu Minh T

FILED: 10/25/2004 ART UNIT: 2114

INVENTOR: Scott L. Michaelis *et al.* CONFIRMATION NO.: 1079

ATTORNEY DOCKET NUMBER 200404973-1

RESPONSE TO NON-FINAL OFFICE ACTION MAILED 03/10/2008

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

This is in response to the Office action mailed March 10, 2008, in the above-captioned application.

REMARKS

No amendments have been made. Claims 1-30 remain in the application. Reexamination and reconsideration is respectfully requested.

Double Patenting Rejections

Claims 1-30 have been rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1-32 of commonly owned U.S. application 10/973,077 and claims 1-26 of commonly owned U.S. application 10/972,796. If the present application is allowed, then the applicant will take appropriate action, including a terminal disclaimer if appropriate.

Rejections Under 35 U.S.C. § 102

Claims 1-4, 7, 9-23, and 29 have been rejected under 35 U.S.C. § 102(e) as allegedly being anticipated by U.S. Patent 7,085,959 (Safford). Applicant respectfully traverses and requests reconsideration of the Examiner's rejection.

As discussed in the response to the previous office action, independent claims 1, 16, 24, and 29 specify determining that the processor for which LOL is detected is assigned a role of boot processor; and switching the role of boot processor to a spare processor without shutting down the system's operating system.

Regarding claim 1, in applicant's response to the previous office action, the applicant discussed the differences between switching processors in Safford, and switching the role of which processor is the boot processor. In the office action dated 03/10/2008, page 4, paragraph A., the examiner states that the features being discussed are not recited in the claims. Boot processors, and switching the role of boot processors are claimed, and the point of the applicant's discussion was that switching processors in

Safford is not the same as switching the role of which processor is the boot processor. Regarding recovery from lock step, the applicant has not argued that Safford does not recover from a loss of lock step, rather the applicant has argued that recovery from lock step does not equate to switching the role of boot processor. To reiterate, the following is a summary of the points being made by the applicant:

1. Independent claims 1, 16, 24, and 29 specify a boot processor. Safford does not discuss a boot processor.

2. Safford does not teach or suggest switching the role of boot processor.

3. Re-booting a single processor is not the same as switching the role of boot processor.

In the office action dated 03/10/2008, pages 5 and 6, paragraph B., the examiner cites Safford column 4, lines 13-15 and column 5, claims 3 and 7 for support for Safford switching a boot processor. As applicant discussed in the response to the previous office action, re-booting a single processor is not the same as switching the role of which processor is the boot processor.

In the office action dated 03/10/2008, pages 6 and 7, paragraph C., the examiner discusses recovery from loss of lock step, making a hot spare, etc. , none of which addresses the claim limitation of switching the role of which processor is the boot processor.

Rejections Under 35 U.S.C. § 103

NOTE: In applicant's reply to the previous office action, applicant asserted that Safford was not proper prior art under 35 U.S.C. § 103. However, that assertion was not correct, and is hereby retracted, with apologies for the mistake. Safford was published on January 8, 2004, before the filing of the present application on October 25, 2004.

Claims 1-23 and 29-30 have been rejected under 35 U.S.C. § 103(a) as allegedly being unpatentable over U.S. Patent number 6,625,749 (Quach) in light of U.S. Patent number 7,251,749 (Fox). Applicant respectfully traverses and requests reconsideration of the Examiner's rejection.

Regarding claims 1, 29 and 30, the examiner cites Quach, figure 1, and column 4, lines 38-41 for determining that a processor for which LOL is detected is assigned the role of boot processor. Figure 1 and the cited text do not illustrate or discuss anything about boot processors. The examiner stipulates that Quach does not explicitly address switching the role of boot processor, and states that Fox teaches a system response to processor failure and dynamically activates a spare processor. Nothing in that discussion provides support for a combination of Quach and Fox teaching or suggesting boot processors or switching the role of boot processors. The examiner then argues that it would be obvious to modify a combination of Quach and Fox, but the argument for modification does not address switching the role of boot processors. No *prima facie* case for obviousness has been presented.

Regarding claims 13-15, the examiner cites Quach, column 13, lines 24-26, for teaching changing a system identifier of a spare processor to that of the boot processor. The renaming being discussed has to do with registers, not processors, and in particular does not teach or suggest anything about a boot processor or switching the role of boot processor. The examiner then further cites Quach, column 13, lines 16-26, and column 14, lines 50-57. Again the discussion in column 13 is about registers. In column 14, claim 6 discusses initializing individual processors, which has nothing to do with a boot processor or switching the role of boot processor.

In view of the above, it is believed all of the pending claims are now in condition for allowance.

Respectfully submitted,
KLAAS, LAW, O'MEARA & MALKIN, P.C.

By: /Augustus W. Winfield/

Augustus W. Winfield, Reg. No. 34,046
1999 Broadway, Suite 2225
Denver, Colorado 80202
Telephone: (303) 298-9888

Date: June 5, 2008

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 200404973-1

Inventor(s): Scott L. Michaelis et al.

Application No.: 10/972,588

Filing Date: October 25, 2004

Confirmation No.: 1079

Examiner: Le, Dieu Minh T.

Group Art Unit: 2114

Title: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

Mail Stop
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL LETTER FOR RESPONSE/AMENDMENT

Transmitted herewith is/are the following in the above-identified application:

- [X] Response/Amendment
- [ ] New fee as calculated below
- [X] No additional fee
- [ ] Other _____ Fee$ _____
- [ ] Petition to extend time to respond
- [ ] Supplemental Declaration

CLAIMS AS AMENDED BY OTHER THAN A SMALL ENTITY						
(1) FOR	(2) CLAIMS REMAINING AFTER AMENDMENT	(3) NUMBER EXTRA	(4) HIGHEST NUMBER PREVIOUSLY PAID FOR	(5) PRESENT EXTRA	(6) RATE	(7) ADDITIONAL FEES
TOTAL CLAIMS	30	MINUS	30	= 0	X $50	$ 0
INDEP. CLAIMS	4	MINUS	4	= 0	X $210	$ 0
☐ FIRST PRESENTATION OF A MULTIPLE DEPENDENT CLAIM					+ $370	$ 0
EXTENSION FEE	☐ 1st Month $120	☐ 2nd Month $460	☐ 3rd Month $1050	☐ 4th Month $1640		$ 0
OTHER FEES						$
TOTAL ADDITIONAL FEE FOR THIS AMENDMENT						$ 0

Charge $ 0 to Deposit Account 08-2025. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees. A duplicate copy of this sheet is enclosed.

Respectfully submitted,

Scott L. Michaelis et al.

By: /Augustus W. Winfield/

Augustus W. Winfield

Attorney/Agent for Applicant(s)

Reg No. : 34,046

Date : June 5, 2008

Telephone : (303) 298-9888

Rev 10/07 (E-TransAmd)

PTO/SB/06 (07-06)
Approved for use through 1/31/2007. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD

Substitute for Form PTO-875

Application or Docket Number	Filing Date	☐ To be Mailed
10/972,588	10/25/2004	

APPLICATION AS FILED – PART I

FOR	(Column 1) NUMBER FILED	(Column 2) NUMBER EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY FEE ($)
☐ BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A	N/A			N/A	
☐ SEARCH FEE (37 CFR 1.16(k), (i), or (m))	N/A	N/A	N/A			N/A	
☐ EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A	N/A			N/A	
TOTAL CLAIMS (37 CFR 1.16(i))	minus 20 =	*	X $ =		OR	X $ =	
INDEPENDENT CLAIMS (37 CFR 1.16(h))	minus 3 =	*	X $ =			X $ =	
☐ APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).						
☐ MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))							
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL			TOTAL	

APPLICATION AS AMENDED – PART II

AMENDMENT 06/05/2008	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY ADDITIONAL FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY ADDITIONAL FEE ($)
Total (37 CFR 1.16(i))	* 30	Minus	** 30	= 0	X $ =		OR	X $50=	0
Independent (37 CFR 1.16(h))	* 4	Minus	***4	= 0	X $ =		OR	X $210=	0
☐ Application Size Fee (37 CFR 1.16(s))									
☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
					TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	0

AMENDMENT	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)		RATE ($)	ADDITIONAL FEE ($)
Total (37 CFR 1.16(i))	*	Minus	**	=	X $ =		OR	X $ =	
Independent (37 CFR 1.16(h))	*	Minus	***	=	X $ =		OR	X $ =	
☐ Application Size Fee (37 CFR 1.16(s))									
☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
					TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.

** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".

*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".

The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

Legal Instrument Examiner:
/BONNIE PHOENIX/

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/972,588	10/25/2004	Scott L. Michaelis	200404973-1	1079

22879 7590 03/10/2008

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

EXAMINER	
LE, DIEU MINH T	
ART UNIT	**PAPER NUMBER**
2114	

NOTIFICATION DATE	DELIVERY MODE
03/10/2008	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

JERRY.SHORMA@HP.COM
mkraft@hp.com
ipa.mail@hp.com

Office Action Summary	Application No.	Applicant(s)	
	10/972,588	MICHAELIS ET AL.	
	Examiner	Art Unit	
	Dieu-Minh Le	2114	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *12 December 2007*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-30* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☒ Claim(s) *24-28* is/are allowed.

6)☒ Claim(s) *1-23,29 and 30* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *25 October 2004* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This Office Action is in response to the communications filed 12/12/07 in application 10/972,588.

2. Claims 1-30 are again presented for examination.

3. The text of those sections of Title 35, U.S. Code not included in this action can be found in a prior Office action.

4. Claims 1-30 are still rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1-32 of U.S. application 10/973,077; and claims 1-26 of U.S. application 10/972,796. Although the conflicting claims are not identical, they are not patentably distinct from each other because the claimed subject matter contains obvious modifications to previous claims 1-32 of U.S. application 10/973,077; and claims 1-26 of U.S. application 10/972,796.

5. Claims 1-4, 7, 9-23, and 29 are again rejected under 35 U.S.C. 102(e) as being anticipated over Safford et al. (U.S. patent 7,085,959 hereafter referred to as Safford).

This rejection is being applied for the same reasons set forth in the previous Office Action mailed 09/12/2007.

As per claims 1-4, 7, 9-23, and 29 see the previous office action for the detailed teaching of Safford.

Applicant asserts that Safford does not disclose or suggest following limitations:

A. Safford does not discuss transfer of system information (which processor is the boot processor) that remains permanent and survives a system reset. Safford discusses rebooting individual processors while a system is running (for example, column 4, lines 57-60, and claim 3), but is not the same as rebooting the entire system, which is initiated by the one designated boot processor after a system reset or power shutdown.

B. Safford does not discuss boot processors, the system determining whether a processor is the boot processor, or switching the role of boot processor from one processor to another.

C. Safford does not discuss switching a hot spare to the role of boot processor and operating system to idle a spare processor .

Examiner respectfully transverses Applicant's argument as follows:

A. First, in response to Applicant's argument that the references fail to show certain features of Applicant's invention, it is noted that the feature upon which Applicant relies (i.e., ***Safford does not discuss transfer of system information (which processer is the boot processor) that remains permanent and survives a system reset. Safford discusses rebooting individual processors while a system is running (for example, column 4, lines 57-60, and claim 3), but is not the same as rebooting the entire system, which is initiated by the one designated boot processor after a system reset or power shutdown***) is not recited in the rejected claim. Although the claim is interpreted in light of the specification, limitations from the specification is not read into the claims. *In re Van Guens, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993).*

Second, Examiner would like to bring Applicant attention to Safford's method and apparatus for recovery from loss of lock step [abstract, col. 2, lines 13-32] comprising ***information stored in the memory*** [col. 1, lines 60 through col. 2, lines 10] and ***master/checker pair used in supporting the recovery from loss of lockstep in a multi-processors system*** [abstract, col. 2, lines 55-57]. In addition, Safford explicitly demonstrated a method and apparatus for recovery from loss of lock step [abstract, col. 2, lines 13-32] comprising ***capability to signal the computer system (i.e., operating system) to idle the lockstep of processors [col. 4, lines 30-63].*** Stafford clearly illustrated a computer system having a means for detecting and recovering of a loss of lock step including idling the lockstep of processor.

B. First, it is NOT true that Safford does not discuss boot processors, assigning the role of boot processor, data storage for assigning the role of boot processor, or transferring the role of boot processor. Safford explicitly disclosed a method and apparatus for recovery from loss of lock step [abstract, col. 2, lines 13-32] comprising ***information stored in the memory*** [col. 1, lines 60 through col. 2, lines 10] and ***master/checker pair used in supporting the recovery from loss of lockstep in a***

multi-processors system [abstract, col. 2, lines 55-57]. Safford demonstrated the re-boot processor and its role of re-boot processor to spare processor [col. 4, lines 13-35 and col. 5, claims 3 and 7] in supporting the computer data processing system for recovering from loss of lockstep in a boot processor (i.e., re-boot processor).

Second, Safford further illustrated capability of moving an architecture state of the loss of lock step processor to a spare of processor, rebooting processor unit becomes a new spare processor unit, and detecting/correcting the loss of lock step in processor [col. 5, claims 1-2, 7, and 14]. Therefore, it is clearly to an ordinary skill in the art that Stafford does teach applicant's limitation.

C. It is NOT true that Safford does not *switching hot spare to the role of boot processor and operating system to idle a spare processor*. Safford explicitly disclosed a method for recovery from loss of lock step in a computer system, comprising: receiving a loss of lock step signal from a processor unit; receiving a notice from the loss of lock step processor unit to take the loss of lock step processor unit off line; and moving an architected state of the loss of lock step processor unit to

a spare processor unit, wherein the spare processor unit becomes an active (i.e., ***hot spare***) processor unit in the computer system. Safford further demonstrated a capability of moving the architected state comprises: copying the architected state from the loss of lock step processor unit to a node controller; and copying the architected state from the node controller to the idle processor unit [col. 5, claims 1 and 5-6]. Therefore, it is clearly to an ordinary skill in the art that Stafford does teach applicant's limitation.

6. Claims 1-23, and 29-30 are rejected under 35 U.S.C. 103(a) as being unpatentable over Quach (U.S. patent 6,625,749) over Fox et al. (U.S. patent 7,251,749 hereafter referred to as Fox).

As per claims 1-2:

Quach explicitly teaches the invention. Quach teaches:

- detecting loss of lockstep (LOL) for a processor in a multi-processor system [abstract, col. 3, lines 35-36 and col. 5, lines 20-28].
- determining that the processor for which said LOL is detected is assigned a role of boot processor [fig. 1, col. 4, lines 38-48].

Quach does not explicitly address:

- switching the role of boot processor to a spare processor without shutting down the system's operating system.

However, Quach does disclose capability of:

- the processors execute in a redundant mode [abstract, figure 1];
- the error recovery routine is invoked when the processor detects a error while operating in the redundant execution mode [abstract, col. 4, lines 30-38].

In addition, Fox explicitly teaches:

- system response to processor failure in a multi-processor data processing system [abstract];
- dynamically activate spare processor when processor failure is detected in a multi-processor data processing system [col.1, lines 8-10].

Therefore, it would have been obvious to a person having ordinary skill in the art at the time of Applicant’s invention to realizing Fox’s dynamically activate spare processor when processor failure is detected in a multi-processor data processing system as being the use of switching the role of boot processor to a spare processor without shutting down the

system's operating system as claimed by Applicant. This is because Quach's mutli-data processors /fault tolerant system explicitly performed data failure detection and recovery (i.e., *loss of lockstep* and/or *fail-over*) within plurality of processors via *state information, parity information*, and *boot information* processes.

This modification would have been obvious because a person having ordinary skill in the art would have been motivated to do so to improve the multi-data processors, failover, and/or fault-tolerant system operation availability and network/system performance therein with a mechanism to enhance the data connectivity, data debugging, data reliability, and data throughput which eventually will increase its performance, such as data throughput between internal and external devices.

As per claims 3-4:

In addition, Fox teaches:

- the spare processor is active spare [col. 3, line 30-31].
- Causing the system's operating system to idle the active spare [col. 3 , lines 31-36].

As per claim 5:

Quach further teaches:

- the spare processor is a processor module that includes a master and slave processor that operate in lockstep [col. 3, lines 36-37].

As per claims 6 and 8:

Quach further teaches:

- accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor [col. 3, lines 35-65].

In addition, Fox teaches:

- At least one processor of the multiprocessor system [MP] is initially provided as a reserve [hot-spare] processor [abstract].

As per claim 7:

Fox further teaches:

- determining the processor that is assigned the role of spare processor [col. 4, lines 46-48].

As per claims 9-10:

Quach further teaches:

- determining step is performed by system firmware [col. 5, lines 17-18].

As per claims 11-12:

Quach further teaches:

- copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to said spare processor [col. 4, lines 12-24].
- copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to each of said lockstep pair of processors of said spare processor [fig. 5, col. 11, lines 1-18].

As per claims 13-15:

Quach further teaches:

- changing a system identifier of the spare processor to that of the boot processor [col. 13, lines 24-26];
changing a system identifier of said boot processor for which said LOL is detected to a different value, which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor [col. 13, lines 16-26; col. 14, lines 50-57].

As per claims 16-23:

Claims 16-23 are similar to claims 1-9 except using system firmware for determining LOL [Quach, col.5, lines 17-18] and copying the state of the determined one of said lockstep pair of processors that is not the cause of LOL to a spare processor [Quach, col.4, lines 12-28, fig.5, col. 11, lines 1-18]. Therefore, these claims are also rejected under the same rationale applied against claims 1-9. In addition, all of the limitations have been noted in the rejection as per claims 1-9.

As per claim 29:

Due to the similarity of claim 29 to claim 1 except for ***a system*** comprising MEANS for detecting loss of lockstep, MEANS for the determining whether the processor for which LOL is detected, MEANS for switching the role of boot processor, etc… instead of ***a method*** comprising detecting loss of lockstep, determining whether the processor for which LOL is detected, switching the role of boot processor, etc…; therefore, this claim is also rejected under the same rationale applied against claim 1. In addition, all of the limitations have been noted in the rejection as per claim 1.

As per claim 30:

Quach explicitly teaches the invention. Quach teaches:

- detecting loss of lockstep (LOL) for a processor in a multi-processor system [abstract, col. 3, lines 35-36 and col. 5, lines 20-28].

- determining that the processor for which said LOL is detected is assigned a role of boot processor [fig. 1, col. 4, lines 38-48].

Safford does not explicitly address:

- information stored to non-volatile data storage and an Itanium Processor Family (IPF) processor.

However, Quach does disclose capability of:

- a computer system includes processor having dual execution cores [abstract] comprising:

- *information stored in the non-volatile memory* [col. 3, lines 52-54] and redundant mode and split mode *in supporting the recovery from loss of lockstep in a multi-processors system*[col. 3, lines 35-37].

In addition, Fox explicitly teaches:

- system response to processor failure in a multi-processor data processing system [abstract];

- information stored in the memory and active/spare pair used in supporting the recovery from loss of lockstep in a multi-processors system[col.8, lines 60-68].

Therefore, it would have been obvious to a person having ordinary skill in the art at the time of Applicant's invention to realizing Fox's information stored in the memory and active/spare pair used in supporting the recovery from loss of lockstep in a multi-processors system as being information stored to non-volatile data storage and an Itanium Processor Family (IPF) processor as claimed by Applicant. This is because Quach's mutli-data processors /fault tolerant system explicitly performed data failure detection and recovery (i.e., *loss of lockstep* and/or *fail-over*) within plurality of processors via *state information*, *parity information*, and *boot information* processes.

This modification would have been obvious because a person having ordinary skill in the art would have been motivated to do so to improve the multi-data processors, failover, and/or fault-tolerant system operation availability and network/system performance.

Allowable Subject Matter

Claims 24-28 are allowed over the prior art of record.

However, These claims are still objected under the double patenting rejection stated in the paragraph above.

7. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

8. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Dieu-Minh Le whose telephone number is (571) 272-3660. The examiner can normally be reached on Monday - Thursday from 8:30 AM to 6:00 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Scott Baderman can be reached on (571)272-3644.
The Tech Center 2100 phone number is (571) 272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/DIEU-MINH T. LE/
DIEU-MINH THAI LE
PRIMARY EXAMINER
ART UNIT 2114

DML
02/28/08

Notice of References Cited	Application/Control No. 10/972,588	Applicant(s)/Patent Under Reexamination MICHAELIS ET AL.	
	Examiner Dieu-Minh Le	Art Unit 2114	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A	US-6,625,749	09-2003	Quach, Nhon	714/10
*	B	US-7,251,746	07-2007	Fox et al.	714/13
	C	US-			
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	10972588	MICHAELIS ET AL.
	Examiner	Art Unit
	Dieu-Minh Le	2114

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	02/28/2008								
	1	✓								
	2	✓								
	3	✓								
	4	✓								
	5	✓								
	6	✓								
	7	✓								
	8	✓								
	9	✓								
	10	✓								
	11	✓								
	12	✓								
	13	✓								
	14	✓								
	15	✓								
	16	✓								
	17	✓								
	18	✓								
	19	✓								
	20	✓								
	21	✓								
	22	✓								
	23	✓								
	24	=								
	25	=								
	26	=								
	27	=								
	28	=								
	29	✓								
	30	✓								

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	10972588	MICHAELIS ET AL.
	Examiner	Art Unit
	Dieu-Minh Le	2114

SEARCHED			
Class	Subclass	Date	Examiner
714	10-13; 712/31	2/21/08	dml

SEARCH NOTES		
Search Notes	Date	Examiner
EAST search all files updated	2/21/08	dml

INTERFERENCE SEARCH			
Class	Subclass	Date	Examiner

EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	25072	(loss adj (lockstep lock-step)) or LOL	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/21 17:49
L2	1637	(boot$3 initializ $3) adj processor	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/21 17:49
L3	59297	(plural$3 multipl $3 multi-) adj (processor CPU)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/21 17:49
L4	29535	(master slave primary second $3 pair spare redundan$2 back-up "back up") adj (processor CPU)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/21 17:49
L5	11	L1 same L2 same L3 same L4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/21 17:49
L6	96	(712/31).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/21 17:49
L7	2051	(714/10-13). ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/21 17:49
L8	2	L5 and (L6 L7)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/21 17:49

L9	31	1 and 3 and 4 and (nonvolatile non-volatile RAM NVRAM)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/21 17:52
L10	14	9 and firmware	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/21 17:52
L11	26	(michaelis-scott$ myer-sylvia$ mchardy-william $).in.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/21 17:53
L12	26	L11	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/21 17:53
L13	8	("714"/$).ccls. and 1 and 2 and 4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/21 17:55
L14	0	("712"/$).ccls. and 1 and 2 and 4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/21 17:55
L15	37	1 and 4 and (nonvolatile non-volatile RAM NVRAM flash remapp$3 map $3)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/21 18:00
L16	10	15 and 2	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2008/02/21 18:00

2/21/08 6:02:35 PM
C:\Documents and Settings\MLe\My Documents\EAST\Workspaces\10972588a.wsp

RECEIVED
CENTRAL FAX CENTER
DEC 12 2007

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION NO.: 10/972,588 EXAMINER: Le, Dieu Minh T

FILED: 10/25/2004 ART UNIT: 2114

INVENTOR: Scott L. Michaelis *et al.* CONFIRMATION NO.: 1079

ATTORNEY DOCKET NUMBER 200404973-1

RESPONSE TO NON-FINAL OFFICE ACTION MAILED 09/12/2007

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

This is in response to the Office action mailed September 12, 2007, in the above-captioned application.

Amendments to the specification are reflected in substitute paragraphs beginning on page 2 of this paper.

Remarks and arguments begin on page 8 of this paper.

RECEIVED
CENTRAL FAX CENTER
DEC 1 2 2007

AMENDMENTS TO THE SPECIFICATION

[0001] The present application is related to the following concurrently filed and commonly assigned U.S. Patent Applications: ~~Serial No. _____ [Attorney Docket No. 200404944-1]~~ U.S. Patent Application Serial No. 10/973,077 (U. S. Published Patent Application 20060107112) titled "SYSTEM AND METHOD FOR ESTABLISHING A SPARE PROCESSOR FOR RECOVERING FROM LOSS OF LOCKSTEP IN A BOOT PROCESSOR"; ~~Serial No. _____ [Attorney Docket No. 200404943-1]~~ U.S. Patent Application Serial No. 10/973,004 (U. S. Published Patent Application 20060107117, now U.S. Patent Number 7,308,566) titled "SYSTEM AND METHOD FOR CONFIGURING LOCKSTEP MODE OF A PROCESSOR MODULE"; ~~Serial No. _____ [Attorney Docket No. 200404946-1]~~ U.S. Patent Application Serial No. 10/972,835 (U. S. Published Patent Application 20060107114) titled "SYSTEM AND METHOD FOR USING INFORMATION RELATING TO A DETECTED LOSS OF LOCKSTEP FOR DETERMINING A RESPONSIVE ACTION"; ~~Serial No. _____ [Attorney Docket No. 200404973-1~~ U.S. Patent Application Serial No. 10/972,588 (U. S. Published Patent Application 20060090064) titled "SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR"; ~~Serial No. _____ [Attorney Docket No. 200404971-1]~~ U.S. Patent Application Serial No. 10/973,003 (U. S. Published Patent Application 20060107116) titled "SYSTEM AND METHOD FOR REESTABLISHING LOCKSTEP FOR A PROCESSOR MODULE FOR WHICH LOSS OF LOCKSTEP IS DETECTED"; ~~Serial No. _____ [Attorney Docket No. 200404970-1]~~ U.S. Patent Application Serial No. 10/972,888 (U. S. Published Patent Application 20060107115) titled "SYSTEM AND METHOD FOR SYSTEM FIRMWARE CAUSING AN OPERATING SYSTEM TO IDLE A PROCESSOR"; ~~Serial No. _____ [Attorney Docket No. 200404972-1]~~ U.S. Patent Application Serial No. 10/973,075 (U. S. Published Patent Application 20060107111) titled "SYSTEM AND METHOD FOR REINTRODUCING A PROCESSOR MODULE TO AN OPERATING SYSTEM AFTER LOCKSTEP RECOVERY"; and ~~Serial No. _____ [Attorney Docket No.~~

PAGE 4/14 * RCVD AT 12/12/2007 5:22:43 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/14 * DNIS:2738300 * CSID:303 297 2266 * DURATION (mm-ss):04-18

~~200404974-1~~] U.S. Patent Application Serial No. 10/972,796 (U. S. Published Patent Application 20060107106) titled "SYSTEM AND METHOD FOR MAINTAINING IN A MULTI-PROCESSOR SYSTEM A SPARE PROCESSOR THAT IS IN LOCKSTEP FOR USE IN RECOVERING FROM LOSS OF LOCKSTEP FOR ANOTHER PROCESSOR", the disclosures of which are hereby incorporated herein by reference.

[0019] As described further herein and in concurrently filed and commonly assigned U.S. Patent Application Serial No. _____ [~~Attorney Docket No. 200404941-1~~] 10/973,076 (U. S. Published Patent Application 20060107107) titled "SYSTEM AND METHOD FOR PROVIDING FIRMWARE RECOVERABLE LOCKSTEP PROTECTION," the disclosure of which is incorporated herein by reference, certain techniques are provided for recovering from LOL detected for a boot processor in a multi-processor system, in which a spare processor is utilized for such recovery. For instance, upon LOL being detected for the system's boot processor, the spare processor assumes the role of boot processor, and then recovery of lockstep for the original boot processor is attempted. Accordingly, for such lockstep recovery technique, a spare processor is used for attempting to recover from LOL detected for the boot processor. Exemplary techniques are described in concurrently filed and commonly assigned U.S. Patent Application Serial No. _____ [~~Attorney Docket No. 200404944-1~~] U.S. Patent Application Serial No. 10/973,077 (U. S. Published Patent Application 20060107112) titled "SYSTEM AND METHOD FOR ESTABLISHING A SPARE PROCESSOR FOR RECOVERING FROM LOSS OF LOCKSTEP IN A BOOT PROCESSOR," the disclosure of which is incorporated herein by reference, for establishing a spare processor for the boot processor. Such exemplary techniques include techniques for establishing a hot spare processor that is held in reserve until such time as it is used to assume the role of boot processor responsive to a detected LOL in the boot processor. The exemplary techniques further include techniques for establishing an active spare processor that is not held in reserve, but is instead made available to the OS for executing instructions during normal system operation and is dynamically transformed into a hot spare (i.e., it is idled) such that it can assume the role of the boot processor responsive to detection of LOL in the boot processor.

PAGE 5/14 * RCVD AT 12/12/2007 5:22:43 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/14 * DNIS:2738300 * CSID:303 297 2266 * DURATION (mm-ss):04-18

[0021] As described further in concurrently filed and commonly assigned U.S. Patent Application Serial No. ~~_____ [Attorney Docket No. 200404944-1]~~ U.S. Patent Application Serial No. 10/973,077 (U. S. Published Patent Application 20060107112) titled "SYSTEM AND METHOD FOR ESTABLISHING A SPARE PROCESSOR FOR RECOVERING FROM LOSS OF LOCKSTEP IN A BOOT PROCESSOR," in certain embodiments, information (referred to herein as a "device tree") is stored to a multi-processor system, wherein such information identifies the processor in the system that is the boot processor and identifies the processor that is the spare for the boot processor. The information is stored such that it is accessible by the system's firmware, and the firmware, upon detecting LOL for the boot processor, determines the spare processor from the device tree and assigns the spare processor the role of boot processor. The firmware then attempts to recover lockstep for the processor for which LOL was detected. If the recovery of lockstep is successful, the processor for which lockstep was recovered (i.e., the former boot processor) may be assigned the role of spare for the current boot processor. Thus, the multi-processor system may recover from LOL detected for the boot processor without requiring a crash of the system (at least for certain types of "recoverable" LOLs).

[0023] Concurrently filed and commonly assigned U.S. Patent Application Serial No. ~~_____ [Attorney Docket No. 200404941-1]~~ 10/973,076 (U. S. Published Patent Application 20060107107) titled "SYSTEM AND METHOD FOR PROVIDING FIRMWARE RECOVERABLE LOCKSTEP PROTECTION," discloses a technique in which lockstep can be recovered for processors (particularly, non-boot processors) without requiring use of a spare processor. Rather, upon LOL being detected for a lockstep pair of processors, the firmware instructs the OS to idle (or "eject") the lockstep pair of processors for which LOL was detected and then the firmware attempts to reestablish lockstep (e.g., by resetting the processors). If successful in reestablishing lockstep, the firmware triggers the OS to recognize the lockstep pair of processors as again being available for servicing instructions.

PAGE 6/14 * RCVD AT 12/12/2007 5:22:43 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/14 * DNIS:2738300 * CSID:303 297 2266 * DURATION (mm-ss):04-18

[0024] However, for various reasons, in certain system architectures problems arise in attempting to idle (or eject) the boot processor from the system. Thus, in certain embodiments, a spare processor is used for recovering from LOL for the system's boot processor. Again, embodiments are provided herein for switching the role of boot processor to an established spare processor responsive to detection of LOL for the boot processor. In certain embodiments, the spare processor is used for recovering from LOL only in the case in which LOL is detected for the boot processor. For instances in which LOL is detected for non-boot processors, the lockstep may be recovered without the use of a spare by using the recovery technique described in concurrently filed and commonly assigned U.S. Patent Application Serial No. ~~[Attorney Docket No. 200404941-1]~~ 10/973,076 (U. S. Published Patent Application 20060107107) titled "SYSTEM AND METHOD FOR PROVIDING FIRMWARE RECOVERABLE LOCKSTEP PROTECTION," for example.

[0041] Thus, operational block 109 essentially makes the spare processor pair 15 the system's boot processor, and then in certain embodiments firmware 18 resets the lockstep processor pair 12 and reestablishes its lockstep in block 110. Once lockstep is reestablished for lockstep processor pair 12, that pair is held in operational block 111 as a hot spare for the new boot processor pair 15. That is, firmware 18 updates the device tree 203 of FIGURE 2 to reflect that lockstep processor pair 12 is a hot spare for the boot processor pair 15. Thus, should a recoverable LOL be detected for the new boot processor 15, the above process may be used by firmware 18 to make the spare lockstep processor pair 12 the boot processor and then recover lockstep for pair 15. An example technique for resetting the lockstep processor pair 12 and reestablishing its lockstep in which the lockstep processor pair 12 is then held as a hot spare is described further in concurrently filed and commonly assigned U.S. Patent Application Serial No. ~~[Attorney Docket No. 200404974-1]~~ U.S. Patent Application Serial No. 10/972,796 (U. S. Published Patent Application 20060107106) titled "SYSTEM AND METHOD FOR MAINTAINING IN A MULTI-PROCESSOR SYSTEM A SPARE PROCESSOR THAT IS IN LOCKSTEP FOR USE IN RECOVERING FROM LOSS OF LOCKSTEP

PAGE 7/14 * RCVD AT 12/12/2007 5:22:43 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/14 * DNIS:2738300 * CSID:303 297 2266 * DURATION (mm-ss):04-18

FOR ANOTHER PROCESSOR", the disclosure of which is hereby incorporated herein by reference.

[0059] Turning to FIGURE 4, an exemplary operational flow diagram for switching the role of boot processor to a hot spare processor according to one embodiment is shown. In this case, an LOL has been detected for the boot processor. In operational block 401, the system firmware allocates space to save the status of the good processor of the lockstep pair of processors for which LOL was detected. Exemplary techniques for determining which of the lockstep pair of processors had the error are disclosed in concurrently filed and commonly assigned U.S. Patent Application Serial No. ~~[Attorney Docket No. 200404946-1]~~ U.S. Patent Application Serial No. 10/972,835 (U. S. Published Patent Application 20060107114) titled "SYSTEM AND METHOD FOR USING INFORMATION RELATING TO A DETECTED LOSS OF LOCKSTEP FOR DETERMINING A RESPONSIVE ACTION," the disclosure of which is hereby incorporated herein by reference. Accordingly, an appropriate amount of data storage space for storing the cache of the "good" processor in the lockstep pair of processors is allocated in block 401. For instance, an amount of SRAM may be allocated for storing the registers from the good processor.

[0066] In operational block 504, the system firmware allocates space to save the status of the good processor of the lockstep pair of processors for which LOL was detected (i.e., the lockstep pair of processors of the CBP). Exemplary techniques for determining which of the lockstep pair of processors had the error are disclosed in concurrently filed and commonly assigned U.S. Patent Application Serial No. ~~[Attorney Docket No. 200404946-1]~~ U.S. Patent Application Serial No. 10/972,835 (U. S. Published Patent Application 20060107114) titled "SYSTEM AND METHOD FOR USING INFORMATION RELATING TO A DETECTED LOSS OF LOCKSTEP FOR DETERMINING A RESPONSIVE ACTION," the disclosure of which is hereby incorporated herein by reference. Accordingly, an appropriate amount of data storage space for storing the cache of the "good" processor in the lockstep pair of processors is

PAGE 8/14 * RCVD AT 12/12/2007 5:22:43 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/14 * DNIS:2738300 * CSID:303 297 2266 * DURATION (mm-ss):04-18

allocated in block 504. For instance, an amount of SRAM may be allocated for storing the registers from the good processor.

PAGE 9/14 * RCVD AT 12/12/2007 5:22:43 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/14 * DNIS:2738300 * CSID:303 297 2266 * DURATION (mm-ss):04-18

REMARKS

The specification has been amended. No claims have been amended. Claims 1-30 remain in the application. Reexamination and reconsideration is respectfully requested.

Amendments to the Specification

In the office action mailed 09/12/2007, the examiner required amendments to the specification to provide application numbers corresponding to attorney docket numbers for co-pending applications. Paragraphs [0001], [0019], [0021], [0023], [0024], [0041], [0059] and [0056] in the specification have been amended to substitute patent application serial numbers, published application numbers, and issued patent numbers, for the originally submitted attorney docket numbers referencing co-pending applications. No new matter has been presented. The amendments are submitted to satisfy a formality, and they do not have any effect on the scope of the claims.

Double Patenting Rejections

Claims 1-30 have been rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1-32 of commonly owned U.S. application 10/973,077 and claims 1-26 of commonly owned U.S. application 10/972,796. Since none of the referenced applications have issued, the applicant will treat this as a provisional double patenting rejection, and no action will be taken at this time. If allowable subject matter is found in at least one of the referenced applications, the applicant will take appropriate action, including a terminal disclaimer if appropriate.

PAGE 10/14 * RCVD AT 12/12/2007 5:22:43 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/14 * DNIS:2738300 * CSID:303 297 2266 * DURATION (mm-ss):04-18

Rejections Under 35 U.S.C. § 102

Claims 1-4, 7, 9-23, and 29 have been rejected under 35 U.S.C. § 102(e) as allegedly being anticipated by U.S. Patent 7,085,959 (Safford). Applicant respectfully traverses and requests reconsideration of the Examiner's rejection.

From MPEP 2131, a claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference. Applicant respectfully submits that Safford does not describe the boot processor elements of the present claims, as discussed below.

Claims 1, 16, 24, and 29

Independent claim 1 specifies determining that the processor for which said LOL is detected is assigned a role of boot processor; and switching the role of boot processor to a spare processor without shutting down the system's operating system. Independent claim 16 specifies system firmware determining that loss of lockstep (LOL) is detected for a lockstep pair of processors that are assigned the role of boot processor in a system; and switching the role of boot processor to said spare processor. Independent claim 24 specifies system firmware operable, responsive to the detection of LOL for a first of said processor modules, to determine whether said first processor module is assigned a role of system boot processor; wherein if determined that said first processor module is assigned the role of system boot processor, said system firmware is further operable to cause said operating system to recognize another processor module as system boot processor without shutting down said operating system. Independent claim 29 specifies means for determining whether the processor for which said LOL is detected is assigned the role of boot processor; and means for switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which said LOL is detected is assigned the role of boot processor. Safford does not discuss boot processors, the system determining whether a processor is the boot processor, or switching the role of boot processor from one processor to another.

PAGE 11/14 * RCVD AT 12/12/2007 5:22:43 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/14 * DNIS:2738300 * CSID:303 297 2266 * DURATION (mm-ss):04-18

In the present application and in Safford, one processor may be halted, and sufficient run-time state information (called architected state in Safford) is transferred from the first processor to a second processor, so that the second processor may resume computation from the point where the first processor is halted. However, changing the designation of which processor is the boot processor requires steps that are different than transferring run-time state, and may be independent from transferring run-time state. That is, changing the designation of which processor is the boot processor may occur without transferring the run-time state. For example, in paragraph [0022], a spare processor that is designated the new boot processor may be idled, so that the process of transferring run-time state as discussed in Safford does not occur at all.

The boot processor is determined by the system, externally from the processors. For example, from paragraphs [0043] through [0045], as the system is booted, all processors are idling (executing do-nothing loops), and the system causes one processor identified by the system (the boot processor) to branch to the entry point of an operating system loader program. In contrast, in Safford, transfer of the architected state of a processor is a transfer of information from one processor to another, as required for one processor to start processing instructions at the point that another processor was halted, while the system is running. Safford does not discuss transfer of system information (which processer is the boot processor) that remains permanent and survives a system reset. Safford discusses rebooting individual processors while a system is running (for example, column 4, lines 57-60, and claim3), but that is not the same as rebooting the entire system, which is initiated by the one designated boot processor after a system reset or power shutdown.

Claims 2-4, 7, 9-15, and 17-23

Claims 2-4, 7, and 9-15 are allowable at least as ultimately depending from allowable base claim 1. Claims 17-23 are allowable at least as ultimately depending from allowable claim 16. Claims 2 and 13-15, 17, and 19-20 are allowable on further independent grounds as discussed below.

PAGE 12/14 * RCVD AT 12/12/2007 5:22:43 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/14 * DNIS:2738300 * CSID:303 297 2266 * DURATION (mm-ss):04-18

Claim 2, dependent on claim 1, further specifies the spare processor is a hot spare that is held in idle until switched the role of boot processor. Safford does discuss boot processors or switching to the role of boot processor.

Claim 13, dependent on claim 1, further specifies switching the role of boot processor to said spare processor comprises changing a system identifier of the spare processor to that of the boot processor. Safford does not discuss system identifiers of processors or changing a system identifier to that of the boot processor.

Claim 14, dependent on claim13, further specifies the system identifier is used by the system's operating system to recognize the boot processor. Safford does not discuss an operating system using a system identifier to recognize the boot processor.

Claim 15, dependent on claim 3, further specifies changing a system identifier of said boot processor for which said LOL is detected to a different value, which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor. Safford does not discuss changing a system identifier of the boot processor or an operating system not recognizing a processor as the boot processor.

Claim 17, dependent on claim 16, further specifies the spare processor is a hot spare that is held in idle until switched the role of boot processor. Safford does not discuss switching a hot spare to the role of boot processor, or switching a hot spare that is idle to the role of boot processor.

Claim 19, dependent on claim 18, further specifies causing the system's operating system to idle the active spare. Safford does not discuss operating systems or operating systems idling a spare processor.

Claim 20, dependent on claim 16, further specifies said switching causes the system's operating system to recognize said spare processor as said boot processor without shutting down said operating system. Safford does not discuss operating systems or operating systems recognizing a spare processor as the boot processor.

PAGE 13/14 * RCVD AT 12/12/2007 5:22:43 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/14 * DNIS:2738300 * CSID:303 297 2266 * DURATION (mm-ss):04-18

Rejections Under 35 U.S.C. § 103

Claims 5-6, 8, 24-28, and 30 have been rejected under 35 U.S.C. § 103(a) as allegedly being unpatentable over Safford in view of U.S. Patent Number 5,915,082 (Marshall). Applicant respectfully traverses and requests reconsideration of the Examiner's rejection.

Applicant respectfully asserts that Safford is not a valid prior art reference that may be used in a rejection under 35 U.S.C. § 103(a). 35 U.S.C. § 103(c) states:

> (c) Subject matter developed by another person, which qualifies as prior art only under one or more of sub-sections (e), (f), and (g) of section 102 of this title, shall not preclude patentability under this section where the subject matter and the claimed invention were, at the time the invention was made, owned by the same person or subject to an obligation of assignment to the same person.

Please note that Safford and this application are assigned to the same entity, Hewlett-Packard Development Company, L.P. Safford was filed before, but did not issue until after the filing date of the current application. Therefore, the disclosures of Safford are available only as 35 U.S.C. § 102(e)-type prior art. 35 U.S.C. § 103(c) specifies that Safford "shall not preclude patentability" of the claimed invention. Accordingly, the rejection of claims 5-6, 8, 24-28, and 30 should therefore be withdrawn.

In view of the above, it is believed all of the pending claims are now in condition for allowance.

Respectfully submitted,
KLAAS, LAW, O'MEARA & MALKIN, P.C.

By: /Augustus W. Winfield/

Augustus W. Winfield, Reg. No. 34,046
1999 Broadway, Suite 2225
Denver, Colorado 80202
Telephone: (303) 298-9888

Date: December 11, 2007

PAGE 14/14 * RCVD AT 12/12/2007 5:22:43 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/14 * DNIS:2738300 * CSID:303 297 2266 * DURATION (mm-ss):04-18

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 200404973-1

Inventor(s): Scott L. Michaelis et al.

Application No.: 10/972,588

Filing Date: October 25, 2004

Confirmation No.: 1079

Examiner: Le, Dieu Minh T.

Group Art Unit: 2114

RECEIVED
CENTRAL FAX CENTER
DEC 12 2007

Title: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

Mail Stop Amendment
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL LETTER FOR RESPONSE/AMENDMENT

Transmitted herewith is/are the following in the above-identified application:

- [X] Response/Amendment
- [ ] New fee as calculated below
- [X] No additional fee
- [ ] Other
- [ ] Petition to extend time to respond
- [ ] Supplemental Declaration

Fee$

CLAIMS AS AMENDED BY OTHER THAN A SMALL ENTITY						
(1) FOR	(2) CLAIMS REMAINING AFTER AMENDMENT	(3) NUMBER EXTRA	(4) HIGHEST NUMBER PREVIOUSLY PAID FOR	(5) PRESENT EXTRA	(6) RATE	(7) ADDITIONAL FEES
TOTAL CLAIMS	30	MINUS	30	= 0	X $50	$ 0
INDEP. CLAIMS	4	MINUS	4	= 0	X $200	$ 0
[ ] FIRST PRESENTATION OF A MULTIPLE DEPENDENT CLAIM					+ $360	$ 0
EXTENSION FEE	[ ] 1st Month $120	[ ] 2nd Month $450	[ ] 3rd Month $1020	[ ] 4th Month $1590		$ 0
					OTHER FEES	$
					TOTAL ADDITIONAL FEE FOR THIS AMENDMENT	$ 0

Charge $ 0 to Deposit Account 08-2025. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees.

[X] A duplicate copy of this transmittal letter is enclosed.

I hereby certify that this paper is being transmitted to the Patent and Trademark Office facsimile number (571) 273-8300.
Date of facsimile: December 12, 2007

Typed Name: Angela Trousset

Signature: Angela Trousset

Respectfully submitted,

Scott L. Michaelis et al.

By

Jay K. Mellon, Esq.
Attorney/Agent for Applicant(s)
Reg No. : 31,393
Date : December 12, 2007
Telephone : (303) 298-9888

Rev 10/06a(TransAmdFax)

PAGE 1/14 * RCVD AT 12/12/2007 5:22:43 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/14 * DNIS:2738300 * CSID:303 297 2266 * DURATION (mm-ss):04-18

RECEIVED
CENTRAL FAX CENTER
DEC 1 2 2007

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 200404973-1

Inventor(s): Scott L. Michaelis et al.

Confirmation No.: 1079

Application No.: 10/972,588

Examiner: Le, Dieu Minh T.

Filing Date: October 25, 2004

Group Art Unit: 2114

Title: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

Mail Stop Amendment
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL LETTER FOR RESPONSE/AMENDMENT

Transmitted herewith is/are the following in the above-identified application:

- [X] Response/Amendment
- [ ] New fee as calculated below
- [X] No additional fee
- [ ] Other
- [ ] Petition to extend time to respond
- [ ] Supplemental Declaration

Fee$

CLAIMS AS AMENDED BY OTHER THAN A SMALL ENTITY						
(1) FOR	(2) CLAIMS REMAINING AFTER AMENDMENT	(3) NUMBER EXTRA	(4) HIGHEST NUMBER PREVIOUSLY PAID FOR	(5) PRESENT EXTRA	(6) RATE	(7) ADDITIONAL FEES
TOTAL CLAIMS	30	MINUS	30	= 0	X $50	$ 0
INDEP. CLAIMS	4	MINUS	4	= 0	X $200	$ 0
[ ] FIRST PRESENTATION OF A MULTIPLE DEPENDENT CLAIM					+ $360	$ 0
EXTENSION FEE	[ ] 1st Month $120	[ ] 2nd Month $450	[ ] 3rd Month $1020	[ ] 4th Month $1590		$ 0
					OTHER FEES	$
					TOTAL ADDITIONAL FEE FOR THIS AMENDMENT	$ 0

Charge $ 0 to Deposit Account 08-2025. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees.

[X] A duplicate copy of this transmittal letter is enclosed.

I hereby certify that this paper is being transmitted to the Patent and Trademark Office facsimile number (571) 273-8300.
Date of facsimile: December 12, 2007

Typed Name: Angela Trousset
Signature: Angela Trousset

Respectfully submitted,
Scott L. Michaelis et al.

By _____
Jay K. Malkin, Esq.
Attorney/Agent for Applicant(s)
Reg No.: 31,393
Date: December 12, 2007
Telephone: (303) 298-9888

Rev 10/06a(TransAmdFax)

PAGE 2/14 * RCVD AT 12/12/2007 5:22:43 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/14 * DNIS:2738300 * CSID:303 297 2266 * DURATION (mm-ss):04-18

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/972,588	10/25/2004	Scott L. Michaelis	200404973-1	1079

22879 7590 09/12/2007

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

EXAMINER	
LE, DIEU MINH T	

ART UNIT	PAPER NUMBER
2114	

MAIL DATE	DELIVERY MODE
09/12/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/972,588	MICHAELIS ET AL.	
	Examiner	Art Unit	
	Dieu-Minh Le	2114	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *25 October 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-30* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-30* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *10/25/07* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *10/25/07*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

Part III DETAILED ACTION

Specification

1. This Office Action is in response to the application 10/972,588 filed on 10/25/04.

2. Claims 1-30 are presented for examination.

3. The disclosure is objected to because of the following informalities: Applicant is required to update all the related applications with the Serial No. or U.S. Patents No. on pages 1, 6-8, 13, 15, 22, and 24.

Appropriate correction is required.

Double Patenting Rejections

4. Claims 1-30 are rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1-32 of U.S. application 10/973,077; and claims 1-26 of U.S. application 10/972,796. Although the conflicting claims are not identical, they are not patentably distinct from each other because the claimed subject matter contains obvious modifications to previous claims 1-32 of U.S. application 10/973,077; and claims 1-26 of U.S. application 10/972,796.

As to claims 1, 16, 24 and 29, these claims include limitations of: detecting loss of lockstep, detecting a role of boot processor, which already included in claims 1-32 of U.S. application 10/973,077; and claims 1-26 of U.S. application 10/972,796. It is well settled that the omission of an element and its function [i.e.,system firmware] is an obvious expedient if the remaining elements perform the same function as before. In re Karlson, 136, USPQ 184 (CCPA 1963). Also note Ex parte Rainu, 168 USPQ 375 (Bd. App. 1969). Therefore, omitting various elements from the previous claimed subject matter would have been obvious to one of ordinary skill in the art in this case since the remaining elements do in fact perform the same

functions as before. Elimination/Changing of an element or its function will not serve as a basis for patentability.

4. The obviousness-type double patenting rejection is a judicially established doctrine based upon public policy and is primarily intended to prevent prolongation of the patent term by prohibiting claims in a second patent not patentably distinct from claims in a first patent. *In re Vogel*, 164 USPQ 619 (CCPA 1970). A timely filed terminal disclaimer in compliance with 37 C.F.R. § 1.321(b) would overcome an actual or provisional rejection on this ground provided the conflicting application or patent is shown to be commonly owned with this application. See 37 C.F.R. § 1.78(d).

Claim Rejections - 35 USC § 102

5. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless -
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section

351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

6. Claims 1-4, 7, 9-23, and 29 are rejected under 35 U.S.C. 102(e) as being anticipated over Safford et al. (U.S. patent 7,085,959 hereafter referred to as Safford).

As per claim 1:

Safford explicitly teaches the invention. Safford teaches:

- A method comprising:

- detecting loss of lockstep (LOL) for a processor in a multi-processor system [col. 4, lines 13-35 and col. 5, claim 7].

- determining that the processor for which said LOL is detected is assigned a role of boot processor [col. 5, claim 3].

- switching the role of boot processor to a spare processor without shutting down the system's operating system [col. 5, claim 1].

As per claim 2:

Safford explicitly teaches the invention. Safford further teaches:

- wherein the spare processor is a hot spare that is held in idle until switched the role of boot processor [abstract, col. 4, lines 19-29 and col. 5, claim 6].

As per claim 3:

Safford explicitly teaches the invention. Safford further teaches:

- the spare processor is an active spare [col. 5, claim 1].

As per claim 4:

Safford explicitly teaches the invention. Safford further teaches:

- wherein said switching further comprises: causing the system's operating system to idle the active spare [col. 5, claims 1 and 6].

This is clearly shown that Safford's method and apparatus for recovery from loss of lock step [abstract, col. 2, lines 13-32] teaching capabilities are corresponding to Applicant's invention.

As per claim 7:

Safford explicitly teaches the invention. Safford further teaches:

- determining the processor that is assigned the role of spare processor [col. 5, claim 1].

As per claim 9:

Safford explicitly teaches the invention. Safford further teaches:

- determining step is performed by system firmware (i.e., hardware/software) [col. 1, lines 10-31; col. 2, lines 55 through col. 3, lines 51].

As per claim 10:

Safford explicitly teaches the invention. Safford further teaches:

- switching step is performed by system firmware (i.e., hardware/software) [col. 1, lines 10-31; col. 2, lines 55 through col. 3, lines 51].

As per claim 11:

Safford explicitly teaches the invention. Safford further teaches:

- determining one of the lockstep pair of processors that is not the cause of said LOL [col. 4, lines 30-63 and col. 5, claim 1].

- copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to said spare processor [col. 5, claims 1, 6, and 13].

As per claim 12:

Safford explicitly teaches the invention. Safford further teaches:

- copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to each of said lockstep pair of processors of said spare processor [col. 5, claims 1, 6, and 13].

As per claims 13-15:

Safford explicitly teaches the invention. Safford further teaches:

- changing a system identifier of the spare processor to that of the boot processor [col. 4, lines 30-63 and col. 5, claim 3];

changing a system identifier of said boot processor for which said LOL is detected to a different value, which

causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor [col. 2, lines 55-67; col. 4, lines 54-63; col. 5, claim 3].

This is clearly shown that Safford's method and apparatus for recovery from loss of lock step [abstract, col. 2, lines 13-32] teaching capabilities are corresponding to Applicant's invention.

As per claim 16:

Safford explicitly teaches the invention. Safford further teaches:

- system firmware determining that loss of lockstep (LOL) is detected for a lockstep pair of processors that are assigned the role of boot processor in a system (i.e., hardware/software) [col. 1, lines 10-31; col. 2, lines 55 through col. 3, lines 51].
- determining one of said lockstep pair of processors that is not the cause of said LOL [col. 4, lines 30-63 and col. 5, claim 1].

- copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to a spare processor [col. 5, claims 1, 6, and 13].

- switching the role of boot processor to said spare processor [col. 1, lines 10-31; col. 2, lines 55 through col. 3, lines 51].

As per claim 17:

Safford explicitly teaches the invention. Safford further teaches:

- wherein the spare processor is a hot spare that is held in idle until switched the role of boot processor [abstract, col. 4, lines 19-29 and col. 5, claim 6].

As per claim 18:

Safford explicitly teaches the invention. Safford further teaches:

- the spare processor is an active spare [col. 5, claim 1].

As per claim 19:

Safford explicitly teaches the invention. Safford further teaches:

- causing the system's operating system to idle the active spare[col. 5, claims 1 and 6].

As per claim 20:

Safford explicitly teaches the invention. Safford further teaches:

- wherein said switching causes the system's operating system to recognize said spare processor as said boot processor without shutting down said operating system [col. 4, lines 30-63 and col. 5, claim 1].

As per claims 21-23:

Safford explicitly teaches the invention. Safford further teaches:

- determining step is performed by system firmware (i.e., hardware/software) [col. 1, lines 10-31; col. 2, lines 55 through col. 3, lines 51];
- copy step is performed by system firmware (i.e., hardware/software) [col. 1, lines 10-31; col. 2, lines 55 through col. 3, lines 51];
- switching step is performed by system firmware (i.e., hardware/software) [col. 1, lines 10-31; col. 2, lines 55 through col. 3, lines 51].

As per claim 29:

Due to the similarity of claim 29 to claim 1 except for ***a system*** comprising MEANS for detecting loss of lockstep, MEANS for the determining whether the processor for which LOL is detected, MEANS for switching the role of boot processor, etc… instead of ***a method*** comprising detecting loss of lockstep, determining whether the processor for which LOL is detected, switching the role of boot processor, etc…; therefore, these claims are also rejected under the same rationale applied against claim 1. **In addition, all of the limitations have been noted in the rejection as per claim 1. Such as,** Safford explicilty illustrates ***switching or changing spare processor to an active processor via an architecture state*** (i.e., ***system information state***) [col. 5, claim 1].

This is clearly shown that Safford's method and apparatus for recovery from loss of lock step [abstract, col. 2, lines 13-32] teaching capabilities are corresponding to Applicant's invention.

Claim Rejections - 35 USC § 103

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8. Claims 5-6, 8, 24-28, and 30 are rejected under 35 U.S.C. 103(a) as being unpatentable over Safford et al. (U.S. patent 7,085,959 hereafter referred to as Safford) in view of Marshall et al. (U.S. patent 5,915,082 hereafter referred to as Marshall).

As per claim 5:

Safford substantially teaches the invention. Safford teaches:

- A method comprising:

- detecting loss of lockstep (LOL) for a processor in a multi-processor system [col. 4, lines 13-35 and col. 5, claim 7];

- spare processor [col. 5, claim 1].

Safford does not explicitly address:

- master and slave processors.

However, Safford does disclose capability of:

- A method and apparatus for recovery from loss of lock step [abstract, col. 2, lines 13-32] comprising:

- ***master/checker pair used in supporting the recovery from loss of lockstep in a multi-processors system*** [abstract, col. 2, lines 55-57].

In addition, Marshall explicitly teaches:

- A method comprising:

- An error detection and fault isolation for lockstep processor systems [abstract, col. 1, lines 10-20] comprising:

- a data connectivity among multi-processors, firmware, OS, error detection and recovery, user interface, etc…. [fig. 2, col. 4, lines 13-35];

- ***configuring and error signaling used to enhance recovery from loss of lockstep in a multi-processors system*** [abstract, col. 2, lines 11-24 and col. 12, claim 1] via ***master/slave processors*** [abstract, col. 3, lines 1-15].

Therefore, it would have been obvious to a person having ordinary skill in the art at the time of Applicant's invention to first realizing Safford's ***master/checker pair used in supporting the recovery from loss of lockstep in a multi-processors system*** as being the master and slave processors as claimed by Applicant. This is because Safford's mutli-data processors /fault tolerant system explicitly performed data failure detection and recovery (i.e., ***loss of lockstep*** and/or ***fail-over***) within plurality of processors via ***state information, parity information***, and ***boot information*** processes. By utilizing these capabilities, the multi-data processing system can be directed or redirected promptly and functioned properly during failover recovery process in supporting the system operation; second, by applying the ***configuring and error signaling used to enhance recovery from loss of lockstep in a multi-processors***

system via ***master/slave processors*** as taught by Marshall in conjunction with the method and apparatus for recovery from loss of lock step as taught by Safford, the multi-processors within fault tolerant/failure recovery networking system including master/slave capability (i.e., failover) can enhance its operation performance, more specifically to ensuring the error detected, corrected, and replaced (i.e., backup) in proper and efficient manner via its state information functionality.

This modification would have been obvious because a person having ordinary skill in the art would have been motivated to do so to improve the multi-data processors, failover, and/or fault-tolerant system operation availability and network/system performance therein with a mechanism to enhance the data connectivity, data debugging, data reliability, and data throughput which eventually will increase its performance, such as data throughput between internal and external devices.

As per claims 6, 8, 28, 30:

Safford substantially teaches the invention. Safford teaches:

- wherein the spare processor is a hot spare that is held in idle until switched the role of boot processor [abstract, col. 4, lines 19-29 and col. 5, claim 6];

- the spare processor is an active spare [col. 5, claim 1].

- wherein said switching further comprises: causing the system's operating system to idle the active spare [col. 5, claims 1 and 6].

Safford does not explicitly address:

- information stored to non-volatile data storage and an Itanium Processor Family (IPF) processor.

However, Safford does disclose capability of:

- A method and apparatus for recovery from loss of lock step [abstract, col. 2, lines 13-32] comprising:

- ***information stored in the memory*** [col. 1, lines 60 through col. 2, lines 10] and ***master/checker pair used in supporting the recovery from loss of lockstep in a multi-processors system*** [abstract, col. 2, lines 55-57].

In addition, Marshall explicitly teaches:

- A method comprising:

- An error detection and fault isolation for lockstep processor systems [abstract, col. 1, lines 10-20] comprising:

- a data connectivity among multi-processors, firmware, OS, error detection and recovery, user interface, etc…. [fig. 2, col. 4, lines 13-35];

- ***memory used to store data*** [col. 3, lines 35-50] ***in supporting the recovery from loss of lockstep in a multi-processors system*** [abstract, col. 2, lines 11-24 and col. 12, claim 1] via ***master/slave processors*** [abstract, col. 3, lines 1-15].

Therefore, it would have been obvious to a person having ordinary skill in the art at the time of Applicant's invention to first realizing Safford's ***information stored in the memory*** and ***master/checker pair used in supporting the recovery from loss of lockstep in a multi-processors system*** as being information stored to non-volatile data storage and an Itanium Processor Family (IPF) processor as claimed by Applicant. This is because Safford's mutli-data processors /fault tolerant system explicitly performed data failure detection and recovery (i.e., ***loss of lockstep*** and/or ***fail-over***) within plurality of processors via ***state information, parity information***, and ***boot information*** processes; second, by applying the ***memory used to store data in supporting the recovery from loss of lockstep in a multi-processors system*** via ***master/slave processors*** as taught

by Marshall in conjunction with the method and apparatus for recovery from loss of lock step as taught by Safford, the multi-processors within fault tolerant/failure recovery networking system including master/slave capability (i.e., failover) can enhance its operation performance, more specifically to ensuring the error detected, corrected, and replaced (i.e., backup) in proper and efficient manner via its state information functionality.

This modification would have been obvious because a person having ordinary skill in the art would have been motivated to do so to improve the multi-data processors, failover, and/or fault-tolerant system operation availability and network/system performance therein.

As per claim 24:

Safford substantially teaches the invention. Safford teaches:

- A method comprising:
- detecting loss of lockstep (LOL) for a processor in a multi-processor system [col. 4, lines 13-35 and col. 5, claim 7];
- an operating system [col. 5, claim 1].

- error detection logic operable to detect loss of lockstep (LOL) for at least one of said processor modules [col. 4, lines 13-35 and col. 5, claim 7].

- system firmware operable, responsive to the detection of LOL for a first of said processor modules, to determine whether said first processor module is assigned a role of system boot processor [col. 5, claim 3].

- wherein if determined that said first processor module is assigned the role of system boot processor, said system firmware is further operable to cause said operating system to recognize another processor module as system boot processor without shutting down said operating system [col. 5, claim 1].

Safford does not explicitly address:

- master and slave processors.

However, Safford does disclose capability of:

- A method and apparatus for recovery from loss of lock step [abstract, col. 2, lines 13-32] comprising:

- ***master/checker pair used in supporting the recovery from loss of lockstep in a multi-processors system*** [abstract, col. 2, lines 55-57].

In addition, Marshall explicitly teaches:

- A method comprising:

- An error detection and fault isolation for lockstep processor systems [abstract, col. 1, lines 10-20] comprising:

- a data connectivity among multi-processors, firmware, OS, error detection and recovery, user interface, etc…. [fig. 2, col. 4, lines 13-35];

- ***configuring and error signaling used to enhance recovery from loss of lockstep in a multi-processors system*** [abstract, col. 2, lines 11-24 and col. 12, claim 1] via ***master/slave processors*** [abstract, col. 3, lines 1-15].

Therefore, it would have been obvious to a person having ordinary skill in the art at the time of Applicant's invention to first realizing Safford's ***master/checker pair used in supporting the recovery from loss of lockstep in a multi-processors system*** as being the master and slave processors as claimed by Applicant. This is because Safford's mutli-data processors /fault tolerant system explicitly performed data failure detection and recovery (i.e., ***loss of lockstep*** and/or ***fail-over***) within plurality of processors via ***state information, parity information***, and ***boot information*** processes. By utilizing

these capabilities, the multi-data processing system can be directed or redirected promptly and functioned properly during failover recovery process in supporting the system operation; second, by applying the ***configuring and error signaling used to enhance recovery from loss of lockstep in a multi-processors system*** via ***master/slave processors*** as taught by Marshall in conjunction with the method and apparatus for recovery from loss of lock step as taught by Safford, the multi-processors within fault tolerant/failure recovery networking system including master/slave capability (i.e., failover) can enhance its operation performance, more specifically to ensuring the error detected, corrected, and replaced (i.e., backup) in proper and efficient manner via its state information functionality.

This modification would have been obvious because a person having ordinary skill in the art would have been motivated to do so to improve the multi-data processors, failover, and/or fault-tolerant system operation availability and network/system performance therein with a mechanism to enhance the data connectivity, data debugging, data reliability, and data throughput which eventually will increase its performance, such as data throughput between internal and external devices.

As per claim 25:

Safford substantially teaches the invention. Safford teaches:

- wherein the another processor module is a processor module assigned a role of spare processor [col. 5, claim 1].

As per claim 26:

Safford substantially teaches the invention. Safford teaches:

- wherein the another processor module is held in idle until said system firmware causes said operating system to recognize said another processor module as said system boot processor [abstract, col. 4, lines 19-29 and col. 5, claim 6].

As per claim 27:

Safford substantially teaches the invention. Safford teaches:

- wherein the system firmware causes the operating system to idle the another processor module before causing said operating system to recognize said another processor module as said system boot processor [col. 4, lines 30-63 and col. 5, claim 1].

onclusion

9. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

10. A shortened statutory period for response to this action is set to expired THREE (3) months, ZERO days from the date of this letter. Failure to respond within the period for response will cause the application to be abandoned. 35 U.S.C. 133.

11. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Dieu-Minh Le whose telephone number is (703)305-9408. The examiner can normally be reached on Monday - Thursday from 8:30 AM to 6:30 PM.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Dieu-Minh Le whose telephone number is (571) 272-3660. The examiner can normally be reached on Monday - Thursday from 8:30 AM to 5:00 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Scott Baderman can be reached on (571)272-3644. The Tech Center 2100 phone number is (571) 272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

DIEU-MINH THAI LE
PRIMARY EXAMINER
ART UNIT 2114

DML
09/05/07

FORM PTO-1449 LIST OF PATENTS AND PUBLICATIONS FOR APPLICANT'S INFORMATION DISCLOSURE STATEMENT (Use several sheets if necessary)	ATTY. DOCKET NO. 200404973-1	APPLICATION NO. 10/977,588	CONFIRMATION NO.
	APPLICANT Scott L. Michaelis et al.		
	FILING DATE	GROUP 2114	

REFERENCE DESIGNATION **U.S. PATENT DOCUMENTS**

EXAMINER INITIAL		DOCUMENT NUMBER	PUBLICATION DATE	NAME	Pages, Columns, Lines Where Relevant Passages or Figures Appear
AM	1A	5,226,152	07/06/1993	Klug et al.	
	1B	5,249,188	09/28/1993	McDonald	
	1C	5,751,932	05/12/1998	Horst et al.	
	1D	5,758,058	05/26/1998	Milburn	
	1E	5,764,660	06/09/1998	Mohat	
	1F	6,065,135	05/16/2000	Marshall et al.	
	1G	6,148,348	11/14/2000	Garnett et al.	
	1H	6,473,869	10/29/2002	Bissett et al.	
	1I	6,560,682	05/06/2003	Miller et al.	
✓	1J	6,604,177	08/05/2003	Kondo et al.	
AM	1K	6,615,366	09/02/2003	Grochowski et al.	

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	PUBLICATION DATE	NAME OF PATENTEE OR APPLICANT	Pages/Columns/Lines Where Relevant Passages/Figures Appear	Check if Translation attached
	1L					
	1M					
	1N					
	1O					
	1P					

OTHER REFERENCES (including Author, Title, Date, Pertinent Pages, etc.)

AM	1Q	Attorney Docket No. 200404974-1, filed concurrently herewith, entitled "SYSTEM AND METHOD FOR MAINTAINING IN A MULTI-PROCESSOR SYSTEM A SPARE PROCESSOR THAT IS IN LOCKSTEP FOR USE IN RECOVERING FROM LOSS OF LOCKSTEP FOR ANOTHER PROCESSOR"
AM	1R	Attorney Docket No. 200404944-1, filed concurrently herewith, entitled "SYSTEM AND METHOD FOR ESTABLISHING A SPARE PROCESSOR FOR RECOVERING FROM LOSS OF LOCKSTEP IN A BOOT PROCESSOR"
AM	1S	Attorney Docket No. 200404946-1, filed concurrently herewith, entitled "SYSTEM AND METHOD FOR USING INFORMATION RELATING TO A DETECTED LOSS OF LOCKSTEP FOR DETERMINING A RESPONSIVE ACTION"

EXAMINER	DATE CONSIDERED
David Lade	8/29/07

Rev 05/04 (PTO1449)

FORM PTO-1449 LIST OF PATENTS AND PUBLICATIONS FOR APPLICANT'S INFORMATION DISCLOSURE STATEMENT (Use several sheets if necessary)	ATTY. DOCKET NO. 200404973-1	APPLICATION NO. 10/972,588	CONFIRMATION NO.
	APPLICANT Scott L. Michaelis et al.		
	FILING DATE	GROUP 2114	

REFERENCE DESIGNATION **U.S. PATENT DOCUMENTS**

EXAMINER INITIAL		DOCUMENT NUMBER	PUBLICATION DATE	NAME	Pages, Columns, Lines Where Relevant Passages or Figures Appear
BM	2A	6,625,749	09/23/2003	Quach	
	2B	6,687,851	02/03/2004	Somers et al.	
	2C	6,754,787	06/22/2004	Miller et al.	
	2D	2002/0144177	10/03/2002	Kondo et al.	
	2E	2003/0070050	04/10/2003	Miller et al.	
	2F	2004/0006722	01/08/2004	Safford	
✓	2G	2004/0019771	01/29/2004	Quach	
BM	2H	2004/0153857	08/05/2004	Yamazaki et al.	
	2I				
	2J				
	2K				

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	PUBLICATION DATE	NAME OF PATENTEE OR APPLICANT	Pages/Columns/Lines Where Relevant Passages/Figures Appear	Check if Translation attached
	2L					
	2M					
	2N					
	2O					
	2P					

OTHER REFERENCES (including Author, Title, Date, Pertinent Pages, etc.)

BM	2Q	Attorney Docket No. 200404941-1, filed concurrently herewith, entitled "SYSTEM AND METHOD FOR PROVIDING FIRMWARE RECOVERABLE LOCKSTEP PROTECTION"
BM	2R	Attorney Docket No. 200404943-1, filed concurrently herewith, entitled "SYSTEM AND METHOD FOR CONFIGURING LOCKSTEP MODE OF A PRCESSOR MODULE"
BM	2S	Attorney Docket No. 200404971-1, filed concurrently herewith, entitled "SYSTEM AND METHOD FOR REESTABLISHING LOCKSTEP FOR A PROCESSOR MODULE FOR WHICH LOSS OF LOCKSTEP IS DETECTED"

EXAMINER	DATE CONSIDERED
David Ton	8/29/07

FORM PTO-1449 LIST OF PATENTS AND PUBLICATIONS FOR APPLICANT'S INFORMATION DISCLOSURE STATEMENT (Use several sheets if necessary)	ATTY. DOCKET NO. 200404973-1	APPLICATION NO. 10/972,588	CONFIRMATION NO.
	APPLICANT Scott L. Michaelis et al.		
	FILING DATE	GROUP 2114	

REFERENCE DESIGNATION **U.S. PATENT DOCUMENTS**

EXAMINER INITIAL		DOCUMENT NUMBER	PUBLICATION DATE	NAME	Pages, Columns, Lines Where Relevant Passages or Figures Appear
	3A				
	3B				
	3C				
	3D				
	3E				
	3F				
	3G				
	3H				
	3I				
	3J				
	3K				

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	PUBLICATION DATE	NAME OF PATENTEE OR APPLICANT	Pages/Columns/Lines Where Relevant Passages/Figures Appear	Check if Translation attached
	3L					
	3M					
	3N					
	3O					
	3P					

OTHER REFERENCES (including Author, Title, Date, Pertinent Pages, etc.)

RDM	3Q	Attorney Docket No. 200404970-1, filed concurrently herewith, entitled "SYSTEM AND METHOD FOR SYSTEM FIRMWARE CAUSING AN OPERATING SYSTEM TO IDLE A PROCESSOR"
RDM	3R	Attorney Docket No. 200404972-1, filed concurrently herewith, entitled "SYSTEM AND METHOD FOR REINTRODUCING A PROCESSOR MODULE TO AN OPERATING SYSTEM AFTER LOCKSTEP RECOVERY"
	3S	

EXAMINER	DATE CONSIDERED
/David Y. Eng/	8/29/07

Notice of References Cited	Application/Control No. 10/972,588	Applicant(s)/Patent Under Reexamination MICHAELIS ET AL.	
	Examiner Dieu-Minh Le	Art Unit 2114	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A	US-7,085,959	08-2006	Safford, Kevin David	714/11
*	B	US-5,915,082	06-1999	Marshall et al.	714/11
*	C	US-7,251,746	07-2007	Fox et al.	714/13
*	D	US-6,625,749	09-2003	Quach, Nhon	714/10
*	E	US-6,920,581	07-2005	Bigbee et al.	714/10
*	F	US-2005/0172164	08-2005	Fox et al.	714/013
*	G	US-2003/0140281	07-2003	Fu et al.	714/38
*	H	US-2005/0240829	10-2005	Safford et al.	714/043
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

Bib Data Sheet

CONFIRMATION NO. 1079

SERIAL NUMBER	FILING OR 371(c) DATE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.
10/972,588	10/25/2004 RULE	714	2114	200404973-1

APPLICANTS

Scott L. Michaelis, Plano, TX;
Anurupa Rajkumari, Round Rock, TX;
William B. McHardy, Fort Collins, CO;

**** CONTINUING DATA ************ N/A ***** None**

**** FOREIGN APPLICATIONS ************ N/A ***** None**

IF REQUIRED, FOREIGN FILING LICENSE GRANTED **
01/14/2005

Foreign Priority claimed	STATE OR COUNTRY	SHEETS DRAWING	TOTAL CLAIMS	INDEPENDENT CLAIMS
☐ yes ☑ no 35 USC 119 (a-d) conditions met ☐ yes ☑ no ☐ Met after Allowance Verified and Acknowledged _____ Examiner's Signature _____ Initials	TX	4	30	4

ADDRESS
22879

TITLE
System and method for switching the role of boot processor to a spare processor responsive to detection of loss of lockstep in a boot processor

FILING FEE RECEIVED	FEES: Authority has been given in Paper	
1058	No. _____ to charge/credit DEPOSIT ACCOUNT No. _____ for following:	☐ All Fees ☐ 1.16 Fees (Filing) ☐ 1.17 Fees (Processing Ext. of time) ☐ 1.18 Fees (Issue) ☐ Other _____ ☐ Credit

Index of Claims

Application/Control No.	Applicant(s)/Patent under Reexamination
10/972,588	MICHAELIS ET AL.
Examiner	**Art Unit**
Dieu-Minh Le	2114

√	Rejected	–	(Through numeral) Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

Claim		Date
Final	Original	9/6/07
	1	v
	2	v
	3	v
	4	v
	5	v
	6	v
	7	v
	8	v
	9	v
	10	v
	11	v
	12	v
	13	v
	14	v
	15	v
	16	v
	17	v
	18	v
	19	v
	20	v
	21	v
	22	v
	23	v
	24	v
	25	v
	26	v
	27	v
	28	v
	29	v
	30	v
	31	
	32	
	33	
	34	
	35	
	36	
	37	
	38	
	39	
	40	
	41	
	42	
	43	
	44	
	45	
	46	
	47	
	48	
	49	
	50	

Claim		Date
Final	Original	
	51	
	52	
	53	
	54	
	55	
	56	
	57	
	58	
	59	
	60	
	61	
	62	
	63	
	64	
	65	
	66	
	67	
	68	
	69	
	70	
	71	
	72	
	73	
	74	
	75	
	76	
	77	
	78	
	79	
	80	
	81	
	82	
	83	
	84	
	85	
	86	
	87	
	88	
	89	
	90	
	91	
	92	
	93	
	94	
	95	
	96	
	97	
	98	
	99	
	100	

Claim		Date
Final	Original	
	101	
	102	
	103	
	104	
	105	
	106	
	107	
	108	
	109	
	110	
	111	
	112	
	113	
	114	
	115	
	116	
	117	
	118	
	119	
	120	
	121	
	122	
	123	
	124	
	125	
	126	
	127	
	128	
	129	
	130	
	131	
	132	
	133	
	134	
	135	
	136	
	137	
	138	
	139	
	140	
	141	
	142	
	143	
	144	
	145	
	146	
	147	
	148	
	149	
	150	

Search Notes

Application/Control No.	Applicant(s)/Patent under Reexamination	
10/972,588	MICHAELIS ET AL.	
Examiner	**Art Unit**	
Dieu-Minh Le	2114	

SEARCHED

Class	Subclass	Date	Examiner
714	10	8/27/2007	DML
	11		
	12		
✓	13	✓	✓
712	31	8/27/2007	DML

INTERFERENCE SEARCHED

Class	Subclass	Date	Examiner

SEARCH NOTES (INCLUDING SEARCH STRATEGY)

	DATE	EXMR
EAST search files USPGPUB, USPAT, USOCR, EPO, JPO, DERWENT, IBM-TDB.	8/27/2007	DML
Inventor names search.	8/27/2007	DML

PALM Intranet

Application Number Submit

IDS Flag Clearance for Application 10972588

IDS Information

Content	Mailroom Date	Entry Number	IDS Review	Last Modified	Reviewer
WIDS	2004-10-25	14	Y ☑	2007-09-07 12:09:30.0	MLe
Update					

10/972,588

EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	2222	lockstep lock-step	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 15:21
L3	25008	(loss adj 1) LOL	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 15:22
L4	51	(detect$3 or track$4) adj 3	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 15:23
L5	1547	(boot$4 or initializ$4) adj processor	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:41
L6	246	spare adj processor	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 15:27
L7	12	4 and 5 and 6	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 15:30
L8	25	(michaelis-scott$ or myer-sylvia$ or mchardy-william$).in.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:14
L10	632	(714/10).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 15:33

EAST Search History

L11	95	(712/31).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:08
L12	52536	(switch$4 chang$4 failover fail-over "fail over") same (spare slave redundan$4 backup back-up "back up" second$4) same processor	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:27
L13	81	12 same 5	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:28
L14	168219	without same ("shut$4 down" stop$4 idle interrupt$4) same (OS "operating system")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:33
L15	6	13 and 14	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:33
L16	20342	multi-processor "multi processor" and 3 and 5	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:45
L17	176	16 and 12 and 14	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:46
L18	7	17 and 6	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 16:46

EAST Search History

S1	14	(("7085959") or ("20050172164") or ("5675794") or ("6438687") or ("6920581") or ("5915082") or ("6625749")).PN.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/26 17:54
S2	2	("7085959").PN.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/08/27 15:05
S3	730	(714/11).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/27 15:06
S4	337	(714/12).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/29 15:32
S5	606	(714/13).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/27 15:07
S6	36	("20020144177"|"20030070050"|"20040006722"|"20040019771"|"20040153857"|"5226152"|"5249188"|"5751932"|"5758058"|"5764660"|"6065135"|"6148348"|"6473869"|"6560682"|"6604177"|"6615366"|"6625749"|"6687851"|"6754787").PN.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/27 15:08
S7	44	("20020144177"|"20030070050"|"20040006722"|"20040019771"|"20040153857"|"6625749"|"6687851"|"6754787")".PN"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/08/27 15:09

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P. O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 200404973-1

IN THE U.S. PATENT AND TRADEMARK OFFICE
Patent Application Transmittal Letter

22264 U.S. PTO
10/972588

Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Sir:

Transmitted herewith for filing under 37 CFR 1.53(b) is a(n): (X) Utility ( ) Design
(X) original patent application,
( ) continuation-in-part application

INVENTOR(S): **Scott L. Michaelis et al.**

TITLE: **System And Method For Switching The Role Of Boot Processor To A Spare Processor Responsive To Detection Of Loss Of Lockstep In A Boot Processor**

Enclosed are:

(X) The Declaration and Power of Attorney. (X) signed ( ) unsigned or partially signed

(X) 4 sheets of drawings (one set) ( ) Associate Power of Attorney

( ) Form PTO-1449 (X) Information Disclosure Statement and Form PTO-1449

( ) Priority document(s) ( ) (Other) _____ (fee $_____)

CLAIMS AS FILED BY OTHER THAN A SMALL ENTITY				
(1) FOR	(2) NUMBER FILED	(3) NUMBER EXTRA	(4) RATE	(5) TOTALS
TOTAL CLAIMS	30 — 20	10	X $18	$ 180
INDEPENDENT CLAIMS	4 — 3	1	X $88	$ 88
ANY MULTIPLE DEPENDENT CLAIMS	0		$300	$ 0
BASIC FEE: Design ($350.00); Utility ($790.00)				$ 790
TOTAL FILING FEE				$ 1,058
OTHER FEES				$
TOTAL CHARGES TO DEPOSIT ACCOUNT				$ 1,058

Charge $ 1,058 to Deposit Account 08-2025. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees. A duplicate copy of this sheet is enclosed.

"Express Mail" label no. EV 375345140US

Date of Deposit Oct. 25, 2004

I hereby certify that this is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to: Commissioner for Patents, Alexandria, VA 22313-1450.

By _____

Typed Name: Phyllis Ewing

Respectfully submitted,

Scott L. Michaelis et al.

By _____

Jody C. Bishop

Attorney/Agent for Applicant(s)
Reg. No. 44,034

Date: Oct. 25, 2004

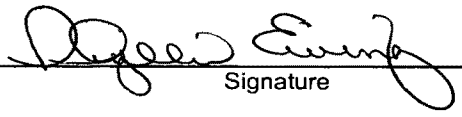
Telephone No.: (214) 855-8007

Certificate of Express Mailing Under 37 CFR 1.10

I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail, Airbill No. EV 375345140US in an envelope addressed to:

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

on October 25, 2004
Date



Signature

Phyllis Ewing
Typed or printed name of person signing Certificate

Registration Number, if applicable

(214) 855-8000
Telephone Number

Note: Each paper must have its own certificate of mailing, or this certificate must identify each submitted paper.

Assignment (copy) 2 pages
Assignment Recordation Cover Sheet - 2 pages (1 orig. + 1 copy)
Certificate of Express Mail - 1 page

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR U.S. LETTERS PATENT

Title:

SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

Inventors:

Scott L. Michaelis
1524 Bamburgh Drive
Plano, TX 75075
Citizenship: U.S.A.

Anurupa Rajkumari
2511 Echo Wood Place
Round Rock, TX 78681
Citizenship: India

William B. McHardy
4023 Sunstone Drive
Fort Collins, CO 80525
Citizenship: U.S.A.

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] The present application is related to the following concurrently filed and commonly assigned U.S. Patent Applications: Serial No. _____ [Attorney Docket No. 200404944-1] titled "SYSTEM AND METHOD FOR ESTABLISHING A SPARE PROCESSOR FOR RECOVERING FROM LOSS OF LOCKSTEP IN A BOOT PROCESSOR"; Serial No. _____ [Attorney Docket No. 200404943-1] titled "SYSTEM AND METHOD FOR CONFIGURING LOCKSTEP MODE OF A PROCESSOR MODULE"; Serial No. _____ [Attorney Docket No. 200404946-1] titled "SYSTEM AND METHOD FOR USING INFORMATION RELATING TO A DETECTED LOSS OF LOCKSTEP FOR DETERMINING A RESPONSIVE ACTION"; Serial No. _____ [Attorney Docket No. 200404973-1 titled "SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR"; Serial No. _____ [Attorney Docket No. 200404971-1] titled "SYSTEM AND METHOD FOR REESTABLISHING LOCKSTEP FOR A PROCESSOR MODULE FOR WHICH LOSS OF LOCKSTEP IS DETECTED"; Serial No. _____ [Attorney Docket No. 200404970-1] titled "SYSTEM AND METHOD FOR SYSTEM FIRMWARE CAUSING AN OPERATING SYSTEM TO IDLE A PROCESSOR"; Serial No. _____ [Attorney Docket No. 200404972-1] titled "SYSTEM AND METHOD FOR REINTRODUCING A PROCESSOR MODULE TO AN OPERATING SYSTEM AFTER LOCKSTEP RECOVERY"; and Serial No. _____ [Attorney Docket No. 200404974-1] titled "SYSTEM AND METHOD FOR MAINTAINING IN A MULTI-PROCESSOR SYSTEM A SPARE PROCESSOR THAT IS IN LOCKSTEP FOR USE IN RECOVERING FROM LOSS OF LOCKSTEP FOR ANOTHER PROCESSOR", the disclosures of which are hereby incorporated herein by reference.

DESCRIPTION OF RELATED ART

[0002] Silent Data Corruption ("SDC") is a difficult problem in the computing industry. In general, SDC refers to data that is corrupt, but which the system does not detect as being corrupt. SDCs primarily occur due to one of two factors: a) a broken hardware unit or b) a

"cosmic" event that causes values to change somewhere in the system. Broken hardware means that a "trusted" piece of hardware is silently giving wrong answers. For example, the arithmetic unit in a processor is instructed to add 1+1 and it returns the incorrect answer 3 instead of the correct answer 2. An example of a cosmic event is when a charged particle (e.g., alpha particle or cosmic ray) strikes a region of a computing system and causes some bits to change value (e.g., from a 0 to a 1 or from a 1 to a 0).

[0003] Numerous techniques have been developed for detecting SDC to prevent the SDC from remaining "silent" or "undetected" within a system, as well as preventing such SDC from propagating through the system. Examples of these techniques include parity-based mechanisms and error correcting codes (ECCs) on buses and memory locations, as well as checksums and/or cyclic redundancy checks (CRC) over regions of memory. Parity-based mechanisms are often employed in processors, wherein a parity bit is associated with each block of data when it is stored. The parity bit is set to one or zero according to whether there is an odd or even number of ones in the data block. When the data block is read out of its storage location, the number of ones in the block is compared with the parity bit. A discrepancy between the values indicates that the data block has been corrupted. ECCs are parity-based mechanisms that track additional information for each data block. The additional information allows the corrupted bit(s) to be identified and corrected.

[0004] Parity/ECC mechanisms have been employed extensively for caches, memories, and similar data storage arrays. In the remaining circuitry on a processor, such as data paths, control logic, execution logic, and registers (the "execution core"), it is more difficult to apply parity/ECC mechanisms for SDC detection. Thus, there is typically some unprotected area on a processor in which data corruption may occur and the parity/ECC mechanisms do not prevent the corrupted data from actually making it out onto the system bus. One approach to SDC detection in an execution core (or other unprotected area of the processor chip) is to employ "lockstep processing." Generally, in lockstep processing two processors are paired together, and the two processors perform exactly the same operations and the results are compared (e.g., with an XOR gate). If there is ever a discrepancy between the results of the lockstep processors, an error is signaled. The odds of two processors experiencing the exact same error at the exact same moment (e.g., due to a cosmic event occurring in both processors at exactly the same time or due to a mechanical failure occurring in each processor at exactly the same time) is nearly zero.

[0005] A pair of lockstep processors may, from time to time, lose their lockstep. "Loss of lockstep" (or "LOL") is used broadly herein to refer to any error in the pair of lockstep processors. One example of LOL is detection of data corruption (e.g., data cache error) in one of the processors by a parity-based mechanism and/or ECC mechanism. Another example of LOL is detection of the output of the paired processors not matching, which is referred to herein as a "lockstep mismatch." It should be recognized that in some cases the data in the cache of a processor may become corrupt (e.g., due to a cosmic event), which once detected (e.g., by a parity-based mechanism or ECC mechanism of the processor) results in LOL. Of course, unless such corrupt data is acted upon by the processor, the output of that processor will not fail to match the output of its paired processor and thus a "lockstep mismatch" will not occur. For example, suppose that a value of "1" is stored to first location of cache in each of a pair of lockstep processors and a value of "1" is also stored to a second location of cache in each of the pair of lockstep processors. Further suppose that a cosmic event occurs for a first one of the processors, resulting in the first location of its cache being changed from "1" to "0", and thus corrupted. This data corruption in the first processor is a LOL for the pair. An error detection mechanism of this first processor may detect the data corruption, thus detecting the LOL. If the processors are instructed to act on the data of their first cache locations, then a lockstep mismatch will occur as the output of each of the processors will not match. For instance, if the processors each add the data stored to the first location of their respective cache with the data stored to the second location of their respective cache, the first processor (having the corrupt data) will output a result of "1" (0+1=1) while the second processor outputs a result of "2" (1+1=2), and thus their respective outputs will not match.

[0006] By employing such techniques as parity-based error detection mechanisms and output comparisons for lockstep paired processors, SDC detection can be enhanced such that practically no SDC occurring in a processor goes undetected (and thus such SDC does not remain "silent") but instead results in detection of LOL. However, the issue then becomes how best for the system to respond to detected LOL. The traditional response to detected LOL has been to crash the system to ensure that the detected error is not propagated through the system. That is, LOL in one pair of lockstep processors in a system halts processing of the system even if other processors that have not encountered an error are present in the system. However, with the increased desire for many systems to maintain high availability, crashing the system each time LOL is detected is not an attractive proposition. This is particularly unattractive for large

systems having many processors because cosmic events typically occur more frequently as the processor count goes up, which would result in much more frequent system crashes in those large systems. High availability is a major desire for many customers having large, multi-processor systems, and thus having their system crash every few weeks is not an attractive option. Of course, permitting corrupt data to propagate through the system is also not a viable option.

[0007] Prior solutions attempting to resolve at least some detected SDCs without requiring the system to be crashed have been Operating System ("OS") centric. That is, in certain solutions the OS has been implemented in a manner to recover from a detected LOL without necessarily crashing the system. This OS-centric type of solution requires a lot of processor and platform specific knowledge to be embedded in the OS, and thus requires that the OS provider maintain the OS up-to-date as changes occur in later versions of the processors and platforms in which the OS is to be used. This is such a large burden that most commonly used OSs do not support lockstep recovery.

[0008] Certain solutions have attempted to recover from a LOL without involving the OS in such recovery procedure. For instance, in one technique upon LOL being detected, firmware is used to save the state of one of the processors in a lockstep pair (the processor that is considered "good") to memory, and then both processors of the pair are reset and reinitialized. Thereafter, the state is copied from the memory to each of the processors in the lockstep pair. This technique makes the processors unavailable for an amount of time without the OS having any knowledge regarding this unavailability, and if the amount of time required for recovery is too long, the system may crash. That is, typically, if a processor is unresponsive for X amount of time, the OS will assume that the processor is hung and will crashdump the system so that the problem can be diagnosed. Further, in the event that a processor in the pair cannot be reset and reinitialized (e.g., the processor has a physical problem and fails to pass its self-test), this technique results in crashing the system.

BRIEF SUMMARY OF THE INVENTION

[0009] According to one embodiment, a method comprises detecting loss of lockstep (LOL) for a processor in a multi-processor system. The method further comprises determining that the processor for which the LOL is detected is assigned the role of boot

processor, and switching the role of boot processor to a spare processor without shutting down the system's operating system.

[0010] According to one embodiment, a method comprises system firmware determining that loss of lockstep (LOL) is detected for a lockstep pair of processors that are assigned the role of boot processor in a system. The method further comprises determining one of the lockstep pair of processors that is not the cause of the LOL, and copying the state of the determined one of the lockstep pair of processors that is not the cause of the LOL to a spare processor. The method further comprises switching the role of boot processor to the spare processor.

[0011] According to one embodiment, a system comprises a plurality of processor modules that each include a master processor and a slave processor that operate in lockstep. The system further comprises an operating system, and error detection logic operable to detect loss of lockstep (LOL) for at least one of the processor modules. The system further comprises system firmware operable, responsive to detection of LOL for a first of the processor modules, to determine whether the first processor module is assigned a role of system boot processor. If determined that the first processor module is assigned the role of system boot processor, the system firmware is further operable to cause the operating system to recognize another processor module as system boot processor without shutting down the operating system.

[0012] According to one embodiment, a system comprises means for detecting loss of lockstep (LOL) for a processor in a multi-processor system. The system further comprises means for determining whether the processor for which the LOL is detected is assigned the role of boot processor, and means for switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which the LOL is detected is assigned the role of boot processor.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] FIGURE 1 shows an example multi-processor system in which firmware utilizes a spare processor for recovering from LOL for the system's boot processor;

[0014] FIGURE 2 shows a block diagram of one embodiment implemented for the IA-64 processor architecture;

[0015] FIGURE 3 shows an example system having multi-processor cells in which an embodiment for switching the role of boot processor to a spare processor responsive to detected LOL in the boot processor may be employed;

[0016] FIGURE 4 shows an exemplary operational flow diagram for switching the role of boot processor to a hot spare processor according to one embodiment;

[0017] FIGURE 5 shows an exemplary operational flow diagram for switching the role of boot processor to an active spare processor according to one embodiment; and

[0018] FIGURE 6 shows a more general operational flow diagram for certain embodiments.

DETAILED DESCRIPTION OF THE INVENTION

[0019] As described further herein and in concurrently filed and commonly assigned U.S. Patent Application Serial No. _____ [Attorney Docket No. 200404941-1] titled "SYSTEM AND METHOD FOR PROVIDING FIRMWARE RECOVERABLE LOCKSTEP PROTECTION," the disclosure of which is incorporated herein by reference, certain techniques are provided for recovering from LOL detected for a boot processor in a multi-processor system, in which a spare processor is utilized for such recovery. For instance, upon LOL being detected for the system's boot processor, the spare processor assumes the role of boot processor, and then recovery of lockstep for the original boot processor is attempted. Accordingly, for such lockstep recovery technique, a spare processor is used for attempting to recover from LOL detected for the boot processor. Exemplary techniques are described in concurrently filed and commonly assigned U.S. Patent Application Serial No. _____ [Attorney Docket No. 200404944-1] titled "SYSTEM AND METHOD FOR ESTABLISHING A SPARE PROCESSOR FOR RECOVERING FROM LOSS OF LOCKSTEP IN A BOOT PROCESSOR," the disclosure of which is incorporated herein by reference, for establishing a spare processor for the boot processor. Such exemplary techniques include techniques for establishing a hot spare processor that is held in reserve until such time as it is used to assume the role of boot processor responsive to a detected LOL in the boot processor. The exemplary techniques further include techniques for establishing an active spare processor that is not held in reserve, but is instead made available to the OS for executing instructions during normal system

operation and is dynamically transformed into a hot spare (i.e., it is idled) such that it can assume the role of the boot processor responsive to detection of LOL in the boot processor.

[0020] Embodiments are provided herein for switching the role of boot processor to an established spare processor responsive to detection of LOL in the boot processor. For instance, exemplary actions by system firmware for switching the role of boot processor to an established spare processor and thus causing the OS to recognize the established spare processor as the system's boot processor are provided in accordance with certain embodiments hereof.

[0021] As described further in concurrently filed and commonly assigned U.S. Patent Application Serial No. _____ [Attorney Docket No. 200404944-1] titled "SYSTEM AND METHOD FOR ESTABLISHING A SPARE PROCESSOR FOR RECOVERING FROM LOSS OF LOCKSTEP IN A BOOT PROCESSOR," in certain embodiments, information (referred to herein as a "device tree") is stored to a multi-processor system, wherein such information identifies the processor in the system that is the boot processor and identifies the processor that is the spare for the boot processor. The information is stored such that it is accessible by the system's firmware, and the firmware, upon detecting LOL for the boot processor, determines the spare processor from the device tree and assigns the spare processor the role of boot processor. The firmware then attempts to recover lockstep for the processor for which LOL was detected. If the recovery of lockstep is successful, the processor for which lockstep was recovered (i.e., the former boot processor) may be assigned the role of spare for the current boot processor. Thus, the multi-processor system may recover from LOL detected for the boot processor without requiring a crash of the system (at least for certain types of "recoverable" LOLs).

[0022] In some implementations described herein, the spare processor for the boot processor is a "hot" spare, in which it is idled and not made available to the OS for processing instructions during normal system operation. That is, the hot spare processor is held in reserve, idling until such time as it assumes the role of boot processor responsive to detected LOL for the boot processor. In other implementations, the spare processor is not held in reserve, but is instead an "active" processor that is made available to the OS for executing instructions during normal system operation. Upon LOL being detected for the boot processor, the designated spare processor is dynamically transformed into a hot spare (i.e., it is idled) such that it can assume the role of the boot processor. For instance, upon the system's firmware detecting a LOL for the

boot processor, the firmware accesses the device tree and determines the designated spare processor, and then the firmware instructs the OS to idle the designated spare processor (i.e., stop scheduling instructions for the processor) to make the "active" spare a "hot" spare, which then assumes the role of boot processor.

[0023] Concurrently filed and commonly assigned U.S. Patent Application Serial No. _____ [Attorney Docket No. 200404941-1] titled "SYSTEM AND METHOD FOR PROVIDING FIRMWARE RECOVERABLE LOCKSTEP PROTECTION," discloses a technique in which lockstep can be recovered for processors (particularly, non-boot processors) without requiring use of a spare processor. Rather, upon LOL being detected for a lockstep pair of processors, the firmware instructs the OS to idle (or "eject") the lockstep pair of processors for which LOL was detected and then the firmware attempts to reestablish lockstep (e.g., by resetting the processors). If successful in reestablishing lockstep, the firmware triggers the OS to recognize the lockstep pair of processors as again being available for servicing instructions.

[0024] However, for various reasons, in certain system architectures problems arise in attempting to idle (or eject) the boot processor from the system. Thus, in certain embodiments, a spare processor is used for recovering from LOL for the system's boot processor. Again, embodiments are provided herein for switching the role of boot processor to an established spare processor responsive to detection of LOL for the boot processor. In certain embodiments, the spare processor is used for recovering from LOL only in the case in which LOL is detected for the boot processor. For instances in which LOL is detected for non-boot processors, the lockstep may be recovered without the use of a spare by using the recovery technique described in concurrently filed and commonly assigned U.S. Patent Application Serial No. _____ [Attorney Docket No. 200404941-1] titled "SYSTEM AND METHOD FOR PROVIDING FIRMWARE RECOVERABLE LOCKSTEP PROTECTION," for example.

[0025] Turning to FIGURE 1, an example embodiment of a system 10 in which a spare processor is used in recovering from detected loss of lockstep (LOL) for a system boot processor is shown. System 10 includes OS 11, as well as master processor 12A and slave processor 12B (collectively referred to as a lockstep processor pair 12). In certain implementations the lockstep processor pair 12 may be implemented on a single silicon chip, which is referred to as a "dual core processor" in which master processor 12A is a first core and slave processor 12B is a second core. Further, lockstep processor pair 12 may be referred to as a

processor or CPU "module" because it includes a plurality of processors (12A and 12B) in such module. As described further herein, to certain portions of system 10, such as its OS 11, the processor module 12 may appear as a single processor because slave processor 12B is merely used for redundant processing for checking the output of master processor 12A. In this example, lockstep processor pair 12 is designated as the boot processor for system 10. As described further herein, in certain embodiments lockstep processor pair 12 is designated as the system boot processor by storing information to the system that specifies such designation.

[0026] Master processor 12A includes cache 14A, and slave processor 12B includes cache 14B. OS 11 and lockstep processor pair 12 are communicatively coupled to bus 16. Typically, master processor 12A and slave processor 12B are coupled to bus 16 via an interface that allows each of such processors to receive the same instructions to process, but such interface only communicates the output of master processor 12A back onto bus 16. The output of slave processor 12B is used solely for checking the output of mater processor 12A.

[0027] In this example, master processor 12A includes error detect logic 13A, and slave processor 12B includes error detect logic 13B. While shown as included in each of the processors 12A and 12B in this example, in certain embodiments the error detect logic 13A and 13B may be implemented external to processors 12A and 12B. Error detect logic 13A and 13B include logic for detecting errors, such as data cache errors, present in their respective processors 12A and 12B. Examples of error detect logic 13A and 13B include known parity-based mechanisms and ECC mechanisms. Error detect logic 13C is also included, which may include an XOR (exclusive OR) gate, for detecting a lockstep mismatch between master processor 12A and slave processor 12B. As mentioned above, a lockstep mismatch refers to the output of master processor 12A and slave processor 12B failing to match. While shown as external to the lockstep processor pair 12 in this example, in certain embodiments error detect logic 13C may be implemented on a common silicon chip with processors 12A and 12B.

[0028] Lockstep mismatch is one way of detecting a LOL between the master processor 12A and slave processor 12B. A detection of an error by either of error detect logic 13A and 13B also provides detection of LOL in the processors 12A and 12B. Because the detection of LOL by error detect logic 13A and 13B may occur before an actual lockstep mismatch occurs, the detection of LOL by error detect logic 13A and 13B may be referred to as a detection of a "precursor to lockstep mismatch". In other words, once an error (e.g., corrupt

data) is detected by error detect logic 13A or 13B, such error may eventually propagate to a lockstep mismatch error that is detectable by error detect logic 13C.

[0029] System 10 further includes a second lockstep processor pair 15 that is communicatively coupled to bus 16. Lockstep processor pair 15 includes master processor 15A and slave processor 15B. Master processor 15A includes cache 17A, and slave processor 15B includes cache 17B. As with error detect logic 13A-13C implemented for lockstep processor pair 12, lockstep processor pair 15 has error detect logic 16A-16C. While only two lockstep processor pairs 12 and 15 are shown for simplicity in the example of FIGURE 1, system 10 may include any number of such lockstep processor pairs. As one specific example, in one implementation system 10 includes 64 lockstep processor pairs, wherein the master processors of the pairs perform parallel processing for the system. In the specific example of FIGURE 1, lockstep processor pair 15 is designated as a spare for boot processor 12, as described further herein.

[0030] In certain embodiments described herein, techniques are provided for switching the role of boot processor to the designated spare processor responsive to detection of LOL for the boot processor. Thus, the spare processor is used for recovery when LOL is detected for the system's boot processor. That is, the designated spare processor assumes the role of boot processor in the event that LOL is detected for the boot processor. In accordance with certain embodiments, upon LOL being detected for a boot processor, a hot spare processor (i.e., an idling processor that is available in the system) is transferred the role of boot processor, and then the old boot processor having LOL is reset to reestablish its lockstep. In other embodiments, a processor is designated as a spare processor for the boot processor, but such designated spare processor is not held in reserve (idled) during normal system operation. Rather, the designated spare is utilized as an active processing resource made available to OS 11 for processing instructions that OS 11 schedules for such "active" spare processor. Upon LOL being detected for the boot processor, the designated spare processor is idled (thereby, transforming such active processor into a hot spare), and is transferred the role of boot processor. Thereafter, the former boot processor for which the LOL was detected may be reset to reestablish its lockstep. Thus, in certain embodiments, a designated spare processor is not required to be held in reserve, but is instead an available resource for use within the system and is only transformed into an actual spare processor when needed for recovering from LOL for the boot processor.

[0031] Firmware 18 is also included in system 10, which in this embodiment is invoked upon an error being detected by any of the error detect logics 13A-13C and 16A-16C. In certain embodiments, processors 12A, 12B, 15A, and 15B are processors from the Itanium Processor Family (IPF). IPF is a 64-bit processor architecture co-developed by Hewlett-Packard Company and Intel Corporation, which is based on Explicitly Parallel Instruction Computing (EPIC). IPF is a well-known family of processors. IPF includes processors such as those having the code names of MERCED, MCKINLEY, and MADISON. In addition to supporting a 64-bit processor bus and a set of 128 registers, the 64-bit design of IPF allows access to a very large memory (VLM) and exploits features in EPIC. While a specific example implementation of one embodiment is described below for the IPF architecture, embodiments described herein for establishing a spare processor for recovering from LOL detected for a boot processor are not limited in application to an IPF architecture, but may be applied as well to other architectures (e.g., 32-bit processor architectures, etc.).

[0032] Processor architecture generally comprises corresponding supporting firmware, such as firmware 18 of system 10. For example, as described further below in conjunction with the specific example of FIGURE 2, the IPF processor architecture comprises such supporting firmware as Processor Abstraction Layer (PAL), System Abstraction Layer (SAL), and Extended Firmware Interface (EFI). Such supporting firmware may enable, for example, the OS to access a particular function implemented for the processor. For instance, the OS may query the PAL as to the size of the cache implemented for the processor, etc. Other well-known functions provided by the supporting firmware (SAL, EFI) include, for example: (a) performing I/O configuration accesses to discover and program the I/O Hardware (SAL_PCI_CONFIG_READ and SAL_PCI_CONFIG_WRITE); (b) retrieving error log data from the platform following a Machine Check Abort (MCA) event (SAL_GET_STATE_INFO); (c) accessing persistent store configuration data stored in non-volatile memory (EFI variable services: GetNextVariableName, GetVariable and SetVariable); and accessing the battery-backed real-time clock/calendar (EFI GetTime and SetTime). Accordingly, the supporting firmware, such as the PAL, is implemented to provide an interface to the processor(s) for accessing the functionality provided by such processor(s). Each of those interfaces provide standard, published procedure calls that are supported.

[0033] In the example system 10 of FIGURE 1, firmware 18 utilizes the above-mentioned hot spare technique for recovering from LOL for a boot processor. That is, processor

15 is designated as a hot spare for boot processor 12, and such processor 15 is held as an idle, spare processor and used in the manner described further herein for assuming the role of boot processor in the event that a LOL is detected for boot processor 12. Of course, as described further herein, the above-mentioned technique in which a designated spare is not held in reserve but is instead dynamically transformed into a spare processor may be implemented by firmware 18 in alternative embodiments. As described further herein, in certain embodiments, a device tree is stored to a data storage device (e.g., SRAM) such that it is accessible to the firmware 18, wherein the device tree includes information identifying processor 12 as the system's boot processor and identifying processor 15 as the spare. The information specifying the boot processor and/or spare may be received as input from a user in certain embodiments. For instance, in certain embodiments a user interface is provided that enables a system administrator to designate the boot processor and/or spare processor in the multi-processor system. In other embodiments, the firmware 18 selects the boot processor and/or spare.

[0034] Suppose that during the system boot-up procedure, master processor 12A assumes the role of boot processor, as shown in the example of FIGURE 1. In this example embodiment, lockstep processor pair 15 is held as a hot spare for recovering from a LOL that may be detected for the boot processor 12A. Of course, additional lockstep processor pairs may be included in system 10 (not specifically shown in the example of FIGURE 1), and those additional lockstep processor pairs, which are not the system boot processor, may recover from LOL in a different manner that does not utilize a hot spare processor. Thus, in accordance with certain embodiments hereof, hot spare processor pair 15 is not needed for recovering from LOL detected for any non-boot processor, but may instead be used for recovery of LOL only if such LOL is detected for the boot processor (processor 12A in this example).

[0035] Firmware 18 is included in this example, and upon detection of LOL by any of error detect logics 13A-13C, firmware 18 determines, in operational block 101, whether the detected LOL is a recoverable LOL. That is, firmware 18 determines in block 101 whether the detected LOL is of a type from which the firmware can recover lockstep for the lockstep processor pair 12 without crashing the system. As described further herein, lockstep is recoverable for certain detected LOLs (which may be referred to as "recoverable LOLs"), while lockstep is not recoverable for other detected LOLs (which may be referred to as "non-recoverable LOLs"). If the lockstep is not recoverable from the detected LOL, then in the example of FIGURE 1 firmware 18 crashes the system in block 102.

[0036] If the lockstep is recoverable, then operation advances to block 103 whereat the firmware determines whether the processor for which the LOL was detected is the system's boot processor. Because the LOL is detected for the lockstep processor pair 12 in this example, firmware 18 determines whether master processor 12A is the system's boot processor. This can be determined, for example, by accessing the device tree 203 of FIGURE 2 (described further below). The device tree may, in certain implementations, include a field for each processor indicating whether such processor is the system's boot processor. This field in the device tree may be set by the firmware during the boot-up process to identify the corresponding processor that is used as the system's boot processor or it may be pre-set by a system administrator (e.g., via a user interface), as examples. If determined in block 103 that the master processor 12A is not the system's boot processor, then operation advances in this example to block 104, and a lockstep recovery process is performed that does not utilize hot spare 15.

[0037] In the specific example shown in FIGURE 1, if the lockstep is determined to be recoverable in block 101 and the processor for which the LOL is detected is determined in block 103 to not be the system boot processor, firmware 18 cooperates with OS 11 via standard OS methods to recover the lockstep. For instance, in the example embodiment of FIGURE 1, Advanced Configuration and Power Interface (ACPI) methods are used by firmware 18 to cooperate with OS 11. Accordingly, no processor or platform specific knowledge is required to be embedded in OS 11, but instead any ACPI-compatible OS may be used, including without limitation HP-UX and Open VMS operating systems. Example techniques that may be used for recovering from LOL for non-boot processors of a system are described further in concurrently filed and commonly assigned U.S. Patent Application Serial No. _____ [Attorney Docket No. 200404941-1] titled "SYSTEM AND METHOD FOR PROVIDING FIRMWARE RECOVERABLE LOCKSTEP PROTECTION," the disclosure of which is incorporated herein by reference.

[0038] In the example embodiment of FIGURE 1, if determined in block 103 that the processor for which the LOL is detected is not the system boot processor, firmware 18 triggers OS 11 to idle the master processor 12A in operational block 104. In this embodiment, firmware 18 utilizes an ACPI method 105 to "eject" master processor 12A, thereby triggering OS 11 to idle the master processor 12A (i.e., stop scheduling tasks for the processor). Of course, by idling master processor 12A, slave processor 12B will in turn be idled. Thus, idling master processor 12A results in idling the lockstep processor pair 12. In this example embodiment, OS

11 is not aware of the presence of slave processor 12B, but is instead aware of master processor 12A. The interface of lockstep processor pair 12 to bus 16 manages copying to slave processor 12B the instructions that are directed by OS 11 to master processor 12A. Thus, firmware 18 need not direct OS 11 to eject slave processor 12B, as OS 11 is not aware of such slave processor 12B in this example implementation. Again, by idling master processor 12A, slave processor 12B is also idled as it merely receives copies of the instructions directed to master processor 12A. Of course, if in a given implementation OS 11 is aware of slave 12B, firmware 18 may be implemented to also direct OS 11 to idle such slave processor 12B in a manner similar to that described for idling master processor 12A.

[0039] Firmware 18 then attempts to recover lockstep for the lockstep processor pair 12 in block 106. For instance, firmware 18 resets the processor pair 12. During such reset of processor pair 12, system 10 can continue to operate on its remaining available processors. Once the processor pair 12 is reset and lockstep is recovered, firmware 18 reintroduces master processor 12A to OS 11 in operational block 107. In this embodiment, firmware 18 updates the ACPI device table information for master processor 12A to indicate that such master processor 12A is "present, functioning and enabled." As discussed in the ACPI 2.0 specification for the _STA status method of a device, the _STA (status) object returns the status of a device, which can be one of the following: enabled, disabled, or removed. In this respect, in the result code (bitmap) bit 0 is set if the device is present; bit 1 is set if the device is enabled and decoding its resources; bit 2 is set if the device should be shown in the UI; bit 3 is set if the device is functioning properly (cleared if the device failed its diagnostics); bit 4 is set if the battery is present; and bits 5-31 are reserved. A device can only decode its hardware resources if both bits 0 and 1 are set. If the device is not present (bit 0 cleared) or not enabled (bit 1 cleared), then the device must not decode its resources. Bits 0, 1 and 3 are the "present, enabled and functioning" bits mentioned above. Firmware 18 utilizes an ACPI method 108 to trigger OS 11 to "check for" master processor 12A, thereby reintroducing the master processor 12A to OS 11. As a result of checking for master processor 12A, OS 11 will recognize that such master processor 12A is again available and will thus begin scheduling tasks for master processor 12A once again.

[0040] In view of the above, a recovery technique is provided that does not require utilizing a spare processor for recovering from LOL detected for non-boot processors. For various reasons, in certain system architectures problems arise in attempting to idle (or eject) the boot processor from the system. Thus, in certain embodiments, a hot spare processor is used for

recovering from LOL for the system's boot processor. For instance, in the example of FIGURE 1, if determined in block 103 that the master processor 12A is the system's boot processor, operation advances to block 109 whereat the state of the "good" processor in the lockstep processor pair 12 is copied over to each of processors 15A and 15B in the spare lockstep processor pair 15. Exemplary techniques that may be utilized by firmware 18 for switching the role of boot processor to the hot spare are described further herein below.

[0041] Thus, operational block 109 essentially makes the spare processor pair 15 the system's boot processor, and then in certain embodiments firmware 18 resets the lockstep processor pair 12 and reestablishes its lockstep in block 110. Once lockstep is reestablished for lockstep processor pair 12, that pair is held in operational block 111 as a hot spare for the new boot processor pair 15. That is, firmware 18 updates the device tree 203 of FIGURE 2 to reflect that lockstep processor pair 12 is a hot spare for the boot processor pair 15. Thus, should a recoverable LOL be detected for the new boot processor 15, the above process may be used by firmware 18 to make the spare lockstep processor pair 12 the boot processor and then recover lockstep for pair 15. An example technique for resetting the lockstep processor pair 12 and reestablishing its lockstep in which the lockstep processor pair 12 is then held as a hot spare is described further in concurrently filed and commonly assigned U.S. Patent Application Serial No. _____ [Attorney Docket No. 200404974-1] titled "SYSTEM AND METHOD FOR MAINTAINING IN A MULTI-PROCESSOR SYSTEM A SPARE PROCESSOR THAT IS IN LOCKSTEP FOR USE IN RECOVERING FROM LOSS OF LOCKSTEP FOR ANOTHER PROCESSOR", the disclosure of which is hereby incorporated herein by reference.

[0042] FIGURE 2 shows a block diagram of one embodiment of the above system 10, which is implemented for the IPF processor architecture and is labeled as system 10_A. The quintessential model of the traditional IPF architecture is given in the Intel IA-64 Architecture Software Developer's Manual, Volume 2: IA-64 System Architecture, in section 11.1 *Firmware Model*, the disclosure of which is hereby incorporated herein by reference. Accordingly, in this example embodiment of system 10_A, firmware 18, labeled as firmware 18_A, includes processor abstraction layer (PAL) 201 and platform/system abstraction layer (SAL) 202. In general, PAL 201 is firmware provided by Intel for its processors, and SAL 202 is developed by an original equipment manufacturer (OEM) for the specific system/platform in which the processors are to be employed. PAL 201, SAL 202, as well as an extended firmware interface (EFI) layer (not

shown), together provide, among other things, the processor and system initialization for an OS boot in an IPF system.

[0043] The boot-up process of a traditional IPF system, for example, proceeds as follows: When the system is first powered on, there are some sanity checks (e.g., power on self-test) that are performed by microprocessors included in the system platform, which are not the main system processors that run applications. After those checks have passed, power and clocks are given to a boot processor (which may, for example, be master processor 12A). The boot processor begins executing code out of the system's Read-Only Memory (ROM) (not specifically shown in FIGURE 2). The code that executes is the PAL 201, which gets control of system 10. PAL 201 executes to acquire all of the processors in system 10_A (recall that there may be many lockstep processor pairs, such as pairs 12 and 15) such that the processors begin executing concurrently through the same firmware.

[0044] After it has performed its duty of initializing the processor(s), PAL 201 passes control of system 10_A to SAL 202. It is the responsibility of SAL 202 to discover what hardware is present on the system platform, and initialize it to make it available for the OS 11. When main memory is initialized and functional, the firmware 18_A is copied into the main memory. Then, control is passed to EFI (not shown), which is responsible for activating boot devices, which typically includes the disk. The EFI reads the disk to load a program into memory, typically referred to as an operating system loader. The EFI loads the OS loader into memory, and then passes it control of system 10_A by branching the boot processor into the entry point of such OS loader program.

[0045] The OS loader program then uses the standard firmware interfaces to discover and initialize system 10_A further for control. One of the things that the OS loader typically has to do in a multi-processor system is to retrieve control of the other processors (those processors other than the boot processor). For instance, at this point in a multi-processor system, the other processors may be executing in do-nothing loops. In an ACPI-compatible system, OS 11 makes ACPI calls to parse the ACPI tables to discover the other processors of a multi-processor system in a manner as is well-known in the art. Then OS 11 uses the firmware interfaces to cause those discovered processors to branch into the operating system code. At that point, OS 11 controls all of the processors and the firmware 18_A is no longer in control of system 10_A.

[0046] As OS 11 is initializing, it has to discover from the firmware 18_A what hardware is present at boot time. And in the ACPI standards, it also discovers what hardware is present or added or removed at run-time. Further, the supporting firmware (PAL, SAL, and EFI) are also used during system runtime to support the processor. For example, OS 11 may access a particular function of master processor 12A via the supporting firmware 18_A, such as querying PAL 201 for the number, size, etc., of the processor's cache 14A. Some other well-known firmware functions that OS 11 may employ during runtime include: (a) PAL 201 may be invoked to configure or change processor features such as disabling transaction queuing (PAL_BUS_SET_FEATURES); (b) PAL 201 may be invoked to flush processor caches (PAL_CACHE_FLUSH); (c) SAL 202 may be invoked to retrieve error logs following a system error (SAL_GET_STATE_INFO, SAL_CLEAR_STATE_INFO); (d) SAL 202 may be invoked as part of hot-plug sequences in which new I/O cards are installed into the hardware (SAL_PCI_CONFIG_READ, SAL_PCI_CONFIG_WRIT); (e) EFI may be invoked to change the boot device path for the next time the system reboots (SetVariable); (f) EFI may be invoked to change the clock/calendar hardware settings; and (g) EFI may be invoked to shutdown the system (ResetSystem).

[0047] A "device tree" is provided, which is shown as device tree 203 in this example. Device tree 203 is stored in SRAM (Scratch RAM) on the cell, which is RAM that is reinitialized. Firmware 18A builds the device tree 203 as it discovers what hardware is installed in the system. Firmware then converts this information to the ACPI tables format and presents it to OS 11 so that OS 11 can know what is installed in the system. The ACPI device tables (not shown) are only consumed by OS 11 at boot time, so they are never updated as things change. For OS 11 to find the current status, it calls an ACPI "method" to discover the "current status". The _STA method described above is an example of such an ACPI method. When _STA is called, the AML can look for properties on the device specified in the firmware device tree and convert that into the Result Code bitmap described above. So, if lockstep has been lost on a processor, firmware 15A will set the device tree property that indicates loss of lockstep, then when OS 11 calls _STA for that device, the "lockstep lost" property directs the AML code to return to "0" in the "functioning properly" bit so that OS 11 can know there is a problem with that processor.

[0048] If a lost lockstep is recovered in accordance with the recovery technique described herein, firmware 18A can indicate that lockstep has been recovered in the device tree

203. Then when _STA is called on that device responsive to the OS receiving the "check for device" ACPI method, the present, enabled and functioning bits will all be set and OS 11 will know the CPU is safe to use.

[0049] Further, device tree 203 includes information designating which processor is the boot processor, as well as a spare processor for the boot processor. A simple example of device tree 203 is shown below in Table 1:

Device	Status	Lockstep Enabled	Boot Processor?	Spare for Boot Processor?
Processor A	Present, Enabled, and Functioning	Yes	Yes	No
Processor B	N/A	Yes	No	Yes
Processor C	Present, Enabled, and Functioning	Yes	No	No
...	...	...	...	...

Table 1

[0050] In the above example of Table 1, 3 processors "Processor A," "Processor B," and "Processor C" are identified in the device tree as being included in the multi-processor system. The status of each of processors Processor A and Processor B is identified as "Present, Enabled, and Functioning." In this example Processor B is a hot spare, so it is held in reserve and does not show up in the static ACPI tables that are built when the system boots. Accordingly the OS does not know about Processor B and does not have a device handle to call the _STA method on. As described above, the _STA method is a known ACPI method for checking the status of the corresponding device. Therefore, the status of Processor B is designated N/A (not applicable) in this example because the _STA method will not be called for this processor as long as it is the hot spare. Of course, upon such spare assuming the role of boot processor, its status is updated to "Present, Enabled, and Functioning." In other embodiments in which Processor B is an active spare, the OS will be aware of such processor and accordingly its status may be also indicated as "Present, Enabled, and Functioning."

[0051] Further, lockstep is identified in this example as enabled for these processors. In certain embodiments, the system administrator can designate, via user interface 20, whether lockstep mode is to be enabled for the system's processors. Further, Processor A is designated as the system's boot processor in the example device tree shown in Table 1, and Processor B is designated as the spare processor. Thus, upon detection of LOL for one of the

processors in the system, the firmware can access this device tree to determine if the processor for which LOL is detected is the system's boot processor; and if determined that the processor for which LOL is detected is the system's boot processor, the firmware can further determine from this device tree the designated spare processor to use in the lockstep recovery process.

[0052] In the example of FIGURE 2, a user interface 20 is included, which allows a user (e.g., system administrator) to designate the system's boot processor and spare processor. In certain embodiments, the user interface 20 further allows the user to designate other information, such as whether lockstep mode is enabled for the processors.

[0053] One example system in which firmware 18_A of FIGURE 2 may be employed is shown in FIGURE 3. FIGURE 3 shows system 30 that includes multi-processor cells A-D, labeled 31_A-31_D, respectively. Cell A 31_A is shown in more detail, and should be understood that cells B-D 31_B-31_D have substantially the same architecture as that of cell A 31_A in this example. The architecture of cell A 31_A described hereafter corresponds to that of Hewlett-Packard's Superdome™ systems. As shown, multi-processor cell A 31_A includes coherency controller 304 that is communicatively coupled to two buses, labeled 305 and 306, respectively. A plurality of processor modules are included within cell A 31_A, shown as CPU 0, CPU 1, CPU 2 and CPU 3 (and labeled 300-303, respectively). More particularly, a plurality of processors modules are communicatively coupled to each of the buses 305 and 306. In this example, CPU 0 (300) and CPU 1 (301) are each communicatively coupled to the first bus 305, and CPU 2 (302) and CPU 3 (303) are each communicatively coupled to the second bus 306. It should be understood that each of CPU 0 (300), CPU 1 (301), CPU 2 (302), and CPU 3 (303) is in actuality a lockstep processor pair, such as the lockstep processor pair 12 of FIGURES 1-3. Thus, for instance, CPU 0 (300) includes both a master processor and a slave processor, as with processors 12A and 12B included in the lockstep processor pair 12 of FIGURES 1-2.

[0054] Coherency controller 304 decodes the address an interrupt is targeted towards and determines which bus (305 or 306) the interrupt packet should be delivered to. Each CPU on the destination bus sees the interrupt and compares the interrupt target address with its internal "LID" register to determine if it is the targeted CPU. If the match is correct, the CPU responds to the interrupt. If the match is not correct, the CPU ignores the interrupt packet.

[0055] As shown, the example firmware 18_A described above with FIGURES 1-2 may be employed to manage the recovery from detected LOLs for the processors of cells A-D

(31_A-31_D). In this example architecture, when designating a spare for the boot processor, such spare is selected to be another processor communicatively coupled to the same bus as the boot processor. For instance, suppose that during the boot-up of system 30, CPU 2 (302) is designated as the boot processor. Accordingly, firmware 18_A will identify another processor that is communicatively coupled to bus 306 to designate as a spare for such boot processor, such as CPU 3 (303). During runtime if a LOL is detected for CPU 2 (302), then firmware 18_A may operate according to the process described above in FIGURE 1 to switch the designation of boot processor to a designated spare, CPU 3 (303). In performing this switch the LID register of the CPU that lost lockstep is copied into the LID register of the spare CPU and the spare CPU pair will then assume the ID of the failed CPU and start responding to interrupts directed to that ID. The LID of the failed CPU is quickly re-written to a different value so that it will no longer respond to interrupts directed to that ID. CPU 2 (302) can then be reset to recover its lockstep, and such CPU 2 (302) may then be designated as a spare for the new boot processor CPU 3 (303).

[0056] It should be recognized that in the example system of FIGURE 3, if a hot spare were used for recovering from LOL for all processors, rather than just for the boot processor, much of the potential processing resources would be required to be held in reserve as spare resources. For instance, a hot spare would be required on each of the buses of each cell. Because there are two processors coupled to each bus in this example (i.e., CPU 0 and CPU 1 coupled to bus 305, and CPU 2 and CPU 3 coupled to bus 306), reserving a hot spare processor for each bus would result in holding half of the total processing resources in reserve. This is an undesirably expensive and wasteful solution. Thus, according to the illustrated embodiment, a hot spare processor is reserved only for the boot processor. Thus, continuing with the above example in which CPU 2 is the boot processor and CPU 3 is held as a "hot" spare, both of CPU 0 and CPU 1 are active (i.e., neither are held in spare). Further, none of the processors of cells B-D need to be held in reserve as hot spares. In the event of a recoverable LOL occurring for any of non-boot processors of the cells, the lockstep recovery process of FIGURE 1 (i.e., operational blocks 104-108) can be utilized without requiring a spare for lockstep recovery for those non-boot processors.

[0057] While in the above example embodiment, a hot spare CPU is maintained for recovery from LOL for the system's boot processor, in an alternative embodiment a hot spare need not be held as such, but rather a processor can be dynamically made a "spare" when needed

for recovery from LOL encountered for the system boot processor. That is, assuming as in the above example of FIGURE 3 that CPU 2 is the system's boot processor, CPU 3 does not have to be held in reserve (idled) as a hot spare for CPU 2, but instead CPU 3 can be active and dynamically idled and made a "spare" upon detection of LOL in CPU 2. Accordingly, CPU 3 may be designated as a spare for the boot processor, but such CPU 3 need not be held in reserve during normal system operation. Instead, the CPU 3 may be dynamically (during system run-time) transformed into a spare processor (e.g., by causing the OS to idle such processor) in the event that a LOL is detected for the boot processor.

[0058] Thus, the resource of this spare CPU need not be wasted during normal runtime, but can be dynamically turned into a spare processor that is available for recovering from LOL encountered on CPU 2 in the manner described above. As an example of this embodiment, the firmware may send an "eject request" on a healthy processor pair (e.g., CPU 3), and lie to the OS by indicating (e.g., in response to an ACPI _STA method for the healthy processor pair) that such processor pair is not functioning. This would cause the OS to eject that processor pair as though it had encountered a LOL. This processor would become the "hot spare" and be used to replace the boot processor. The system boot processor role would be assumed by the idled healthy processor pair, just as described above in the example in which the healthy CPU 3 is held idle for the system boot processor, CPU 2. The only difference being that now the firmware had to take some initial action to turn the healthy processor pair into a "hot spare" by idling it and making it appear to the OS as though it was not functioning. Thus, the LIDs would be swapped. That is, the LID register of the boot processor that lost lockstep is copied into the LID register of the now spare processor pair (CPU 3 in this example), and the LID register of this spare processor pair would be copied into the LID register of the original boot processor. Lockstep is then reestablished on the original boot processor, and it can be reintroduced to the OS with the ID of the original non-boot processor that replaced it as the system boot processor. The processor that was formerly the boot processor may be designated, in device tree 203, as the spare for the now boot processor.

[0059] Turning to FIGURE 4, an exemplary operational flow diagram for switching the role of boot processor to a hot spare processor according to one embodiment is shown. In this case, an LOL has been detected for the boot processor. In operational block 401, the system firmware allocates space to save the status of the good processor of the lockstep pair of processors for which LOL was detected. Exemplary techniques for determining which of the

lockstep pair of processors had the error are disclosed in concurrently filed and commonly assigned U.S. Patent Application Serial No. _____ [Attorney Docket No. 200404946-1] titled "SYSTEM AND METHOD FOR USING INFORMATION RELATING TO A DETECTED LOSS OF LOCKSTEP FOR DETERMINING A RESPONSIVE ACTION," the disclosure of which is hereby incorporated herein by reference. Accordingly, an appropriate amount of data storage space for storing the cache of the "good" processor in the lockstep pair of processors is allocated in block 401. For instance, an amount of SRAM may be allocated for storing the registers from the good processor.

[0060] In operational block 402, the state of the good processor is stored to the allocated storage space. For instance, in this illustrative example, PAL is called to dump the processor state of the good processor of the current boot processor (CBP) module (for which LOL was detected) to the allocated storage space.

[0061] In operational block 403, the LID of the current boot processor (for which LOL was detected) is reprogrammed to a different value. For instance, the current boot processor may be assigned a LID that is reserved just for this situation (i.e., for switching the role of boot processor). Accordingly, the boot processor for which LOL was detected will not respond to the interrupts addressed for its original LID because the boot processor's LID value is reprogrammed to a different value.

[0062] In operational block 404, the state of the good processor that was stored in block 402 to the allocated storage space is stored to the hot spare processor. In this example, the PAL is called to "Restore Processor State" on the hot spare processor, wherein the PAL is provided a pointer to the address to which the good processor's state was stored in block 402. Accordingly, all the values from that storage space will be read into the registers of the spare processor. As a result, the LID value of the spare processor is programmed with the value of the boot processor that was saved to the allocated storage space. Accordingly, the spare processor automatically, as a result of receiving this stored state, receives the Local ID corresponding to the boot processor for interrupt handling.

[0063] In operational block 405, the system firmware updates its device tree information to reflect that the hot spare is assuming the role of boot processor. The system firmware further updates its device tree, in this example, to indicate that the former boot

processor (for which LOL was detected) is assuming the role of hot spare for the new boot processor.

[0064] In operational block 406, PAL_MC_RESUME (which is a standard PAL call for resuming normal execution after a machine check (MC)) is called for the hot spare (which is assuming the role of boot processor) to cause the OS to recognize the hot spare processor as available for processing instructions, wherein the OS recognizes the hot spare as the boot processor. Thus, the hot spare has now assumed the role of the boot processor, taking the place of the former boot processor for which LOL was detected. Accordingly, in operational block 407 the former hot spare (which has assumed the role of boot processor) resumes normal operation as the system's boot processor with its lockstep protection enabled. Thus, in this exemplary embodiment, upon detection of LOL for a boot processor, the role of boot processor is switched to a hot spare that has lockstep protection, wherein the system effectively recovers lockstep for its boot processor without requiring a system shut down (i.e., without shutting down the OS). In certain implementations, the firmware may go on to take action to attempt to reestablish lockstep for the former boot processor (e.g., by resetting the former boot processor), and if successful in reestablishing its lockstep, such former boot processor may be held in reserve as a hot spare processor for the new boot processor in the event that LOL is detected for this new boot processor.

[0065] Turning to FIGURE 5, an exemplary operational flow diagram for switching the role of boot processor to an active spare processor according to one embodiment is shown. Again, an LOL has been detected for the boot processor. In operational block 501, the system firmware, responsive to such detection of LOL for the boot processor, examines the device tree to determine which processor module is assigned the role of active spare. In operational block 502 the system firmware (e.g., SAL) asserts a General Purpose Event Interrupt (GPE) by writing to the appropriate GPE register in the ACPI register space. This causes the OS to execute its interrupt handler, which will run an AML Method associated with this particular interrupt. In this example, this AML method that is executed responsive to the interrupt generated by the GPE includes an AML instruction called "Notify". This "Notify" operation indicates the determined active spare processor object and that an Eject is being requested as arguments. Accordingly, an AML method is used to request the OS to eject the active spare that was determined in block 501. In response, the OS idles the active spare processor and returns control over the active spare processor to the system firmware, in block 503.

[0066] In operational block 504, the system firmware allocates space to save the status of the good processor of the lockstep pair of processors for which LOL was detected (i.e., the lockstep pair of processors of the CBP). Exemplary techniques for determining which of the lockstep pair of processors had the error are disclosed in concurrently filed and commonly assigned U.S. Patent Application Serial No. _____ [Attorney Docket No. 200404946-1] titled "SYSTEM AND METHOD FOR USING INFORMATION RELATING TO A DETECTED LOSS OF LOCKSTEP FOR DETERMINING A RESPONSIVE ACTION," the disclosure of which is hereby incorporated herein by reference. Accordingly, an appropriate amount of data storage space for storing the cache of the "good" processor in the lockstep pair of processors is allocated in block 504. For instance, an amount of SRAM may be allocated for storing the registers from the good processor.

[0067] In operational block 505, the state of the good processor is stored to the allocated storage space. For instance, in this illustrative example, PAL is called to dump the processor state of the good processor of the current boot processor (CBP) module (for which LOL was detected) to the allocated storage space.

[0068] In operational block 506, the LID of the current boot processor (for which LOL was detected) is reprogrammed to a different value. For instance, the current boot processor may be assigned a LID that is reserved just for this situation (i.e., for switching the role of boot processor). Accordingly, the boot processor for which LOL was detected will not respond to the interrupts addressed for its original LID because the boot processor's LID value is reprogrammed to a different value.

[0069] In operational block 507, the state of the good processor that was stored in block 505 to the allocated storage space is stored to the idled spare processor. In this example, the PAL is called to "Restore Processor State" on the active spare processor, wherein the PAL is provided a pointer to the address to which the good processor's state was stored in block 505. Accordingly, all the values from that storage space will be read into the registers of the spare processor. As a result, the LID value of the spare processor is programmed with the value of the boot processor that was saved to the allocated storage space. Accordingly, the spare processor automatically, as a result of receiving this stored state, receives the Local ID (LID) corresponding to the boot processor for interrupt handling.

[0070] In operational block 508, the system firmware updates its device tree information to reflect that the active spare is assuming the role of boot processor. The system firmware further updates its device tree, in this example, to indicate that the former boot processor (for which LOL was detected) is assuming the role of active spare for the new boot processor.

[0071] In operational block 509, PAL_MC_RESUME (which is a standard PAL call for resuming normal execution after a machine check (MC)) is called for the active spare (which is assuming the role of boot processor) to cause the OS to recognize the active spare processor as available for processing instructions, wherein the OS recognizes the active spare as the boot processor. Thus, the active spare has now assumed the role of the boot processor, taking the place of the former boot processor for which LOL was detected. Accordingly, in operational block 510 the former active spare (which has assumed the role of boot processor) resumes normal operation as the system's boot processor with its lockstep protection enabled. Thus, in this exemplary embodiment, upon detection of LOL for a boot processor, an active spare processor is idled, and the role of boot processor is switched to such idled spare that has lockstep protection, wherein the system effectively recovers lockstep for its boot processor without requiring a system shut down (i.e., without shutting down the OS). In certain implementations, the firmware may go on to take action to attempt to reestablish lockstep for the former boot processor (e.g., by resetting the former boot processor), and if successful in reestablishing its lockstep, such former boot processor may be reintroduced to the OS as an active spare processor for the new boot processor in the event that LOL is detected for this new boot processor.

[0072] The exemplary procedures of FIGURES 4 and 5 switch the role of boot processor to a spare processor in accordance with certain embodiments. While exemplary operational flows are provided in FIGURES 4 and 5, embodiments hereof are not limited to those illustrative examples. For instance, FIGURE 6 shows a more general operational flow diagram for certain embodiments. In operational block 601, LOL is detected for a processor in a multi-processor system. In operational block 602, the system (e.g., system firmware) determines whether the processor for which LOL is detected is the system's boot processor. As described above, this is determined in certain embodiments by accessing the node of the device tree corresponding to the processor for which LOL is detected and determining if the information in that node of the device tree indicates that the processor is the system's boot processor.

[0073] In operational block 603, responsive to determining that the processor for which LOL is detected is the system's boot processor, the system's firmware switches the role of boot processor to a spare processor, which may be either a hot spare or an active spare, as described above. Accordingly, the system's OS recognizes the spare processor as the system boot processor. In view of the above, embodiments hereof enable the role of boot processor to be dynamically switched, during system runtime, to a spare processor responsive to a detected LOL for the boot processor, thus enabling recovery from the LOL without requiring that the system be crashed or re-booted.

CLAIMS

What is claimed is:

1. A method comprising:

detecting loss of lockstep (LOL) for a processor in a multi-processor system;

determining that the processor for which said LOL is detected is assigned a role of boot processor; and

switching the role of boot processor to a spare processor without shutting down the system's operating system.

2. The method of claim 1 wherein the spare processor is a hot spare that is held in idle until switched the role of boot processor.

3. The method of claim 1 wherein the spare processor is an active spare.

4. The method of claim 3 wherein said switching further comprises:

causing the system's operating system to idle the active spare.

5. The method of claim 1 wherein the spare processor is a processor module that includes a master and slave processors that operate in lockstep.

6. The method of claim 1 wherein said determining that the processor for which said LOL is detected is assigned the role of boot processor comprises:

accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor.

7. The method of claim 1 further comprising:

determining the processor that is assigned the role of spare processor.

8. The method of claim 7 wherein said determining the processor that is assigned the role of spare processor comprises:

accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of spare processor.

9. The method of claim 1 wherein said determining step is performed by system firmware.

10. The method of claim 1 wherein said switching step is performed by system firmware.

11. The method of claim 1 wherein the processor for which said LOL is detected includes a lockstep pair of processors, and wherein said switching comprises:

determining one of the lockstep pair of processors that is not the cause of said LOL; and

copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to said spare processor.

12. The method of claim 11 wherein said spare processor includes a lockstep pair of processors, and wherein said copying comprises:

copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to each of said lockstep pair of processors of said spare processor.

13. The method of claim 1 wherein said switching the role of boot processor to said spare processor comprises:

changing a system identifier of the spare processor to that of the boot processor.

14. The method of claim 13 wherein the system identifier is used by the system's operating system to recognize the boot processor.

15. The method of claim 13 further comprising:

changing a system identifier of said boot processor for which said LOL is detected to a different value, which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor.

16. A method comprising:

system firmware determining that loss of lockstep (LOL) is detected for a lockstep pair of processors that are assigned the role of boot processor in a system;

determining one of said lockstep pair of processors that is not the cause of said LOL;

copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to a spare processor; and

switching the role of boot processor to said spare processor.

17. The method of claim 16 wherein the spare processor is a hot spare that is held in idle until switched the role of boot processor.

18. The method of claim 16 wherein the spare processor is an active spare.

19. The method of claim 18 further comprising:

causing the system's operating system to idle the active spare.

20. The method of claim 16 wherein said switching causes the system's operating system to recognize said spare processor as said boot processor without shutting down said operating system.

21. The method of claim 16 wherein said determining step is performed by system firmware.

22. The method of claim 16 wherein said copying step is performed by system firmware.

23. The method of claim 16 wherein said switching step is performed by system firmware.

24. A system comprising:

a plurality of processor modules that each include a master processor and a slave processor that operate in lockstep;

an operating system;

error detection logic operable to detect loss of lockstep (LOL) for at least one of said processor modules; and

system firmware operable, responsive to the detection of LOL for a first of said processor modules, to determine whether said first processor module is assigned a role of system boot processor;

wherein if determined that said first processor module is assigned the role of system boot processor, said system firmware is further operable to cause said operating system to recognize another processor module as system boot processor without shutting down said operating system.

25. The system of claim 24 wherein the another processor module is a processor module assigned a role of spare processor.

26. The system of claim 24 wherein the another processor module is held in idle until said system firmware causes said operating system to recognize said another processor module as said system boot processor.

27. The system of claim 24 wherein the system firmware causes the operating system to idle the another processor module before causing said operating system to recognize said another processor module as said system boot processor.

28. The system of claim 24 further comprising:

non-volatile data storage storing information that assigns the role of system boot processor to one of said plurality of processor modules.

29. A system comprising:

means for detecting loss of lockstep (LOL) for a processor in a multi-processor system;

means for determining whether the processor for which said LOL is detected is assigned the role of boot processor; and

means for switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which said LOL is detected is assigned the role of boot processor.

30. The system of claim 29 wherein said processor in said multi-processor system is an Itanium Processor Family (IPF) processor.

ABSTRACT

According to one embodiment, a method comprises detecting loss of lockstep (LOL) for a processor in a multi-processor system. The method further comprises determining that the processor for which the LOL is detected is assigned the role of boot processor, and switching the role of boot processor to a spare processor without shutting down the system's operating system. In another embodiment, a method comprises system firmware determining that an LOL is detected for a lockstep pair of processors that are assigned the role of boot processor in a system. The method further comprises determining one of the lockstep pair of processors that is not the cause of the LOL, and copying the state of the determined one of the lockstep pair of processors that is not the cause of the LOL to a spare processor. The method further comprises switching the role of boot processor to the spare processor.

FIG. 1

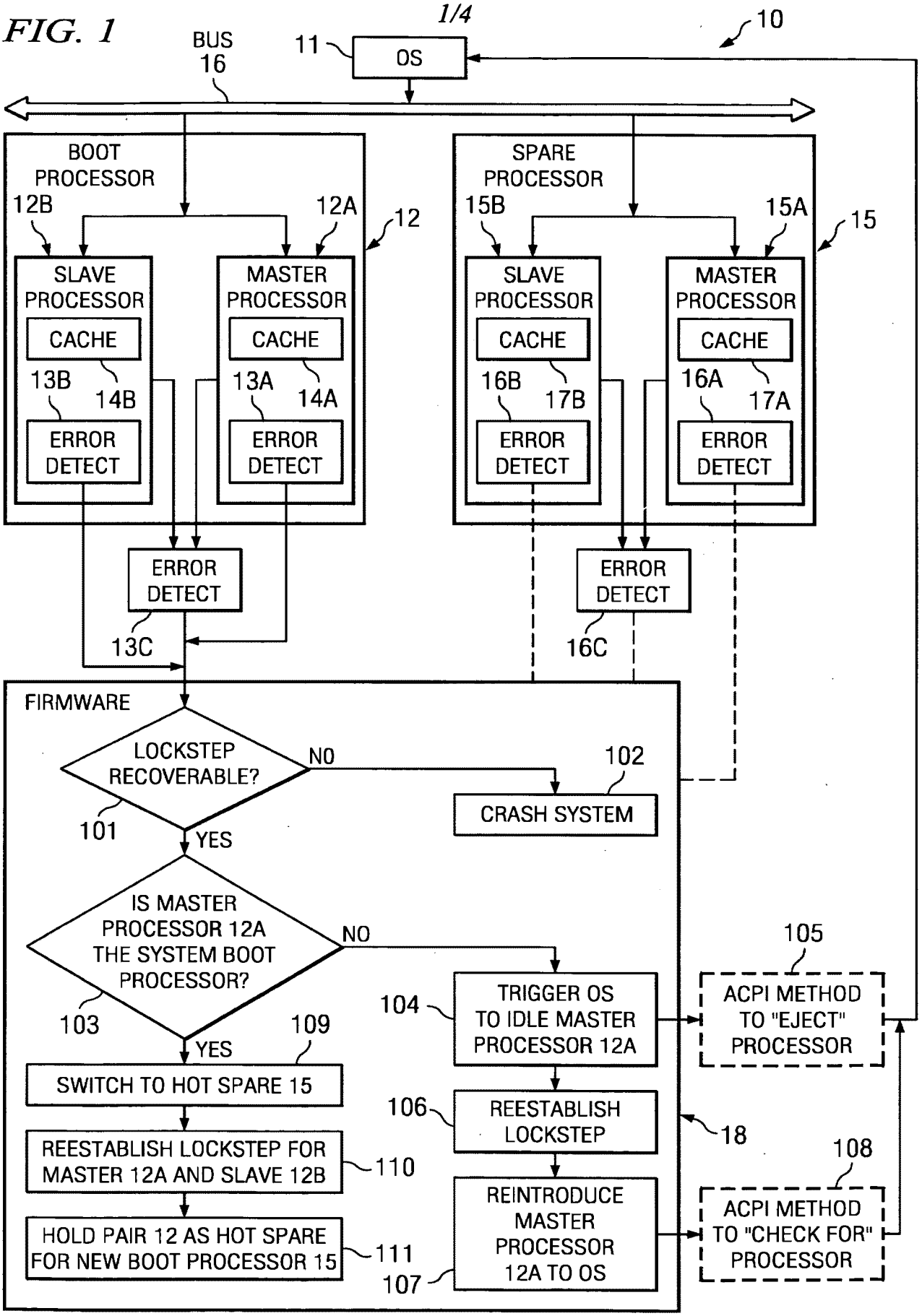

10
BUS
16
11
OS
BOOT PROCESSOR
12B
12A
12
SLAVE PROCESSOR
MASTER PROCESSOR
CACHE
13B
14B
13A
14A
ERROR DETECT
SPARE PROCESSOR
15B
15A
15
16B
17B
16A
17A
ERROR DETECT
13C
16C
FIRMWARE
LOCKSTEP RECOVERABLE?
NO
102
CRASH SYSTEM
101
YES
IS MASTER PROCESSOR 12A THE SYSTEM BOOT PROCESSOR?
NO
105
103
YES
109
104
TRIGGER OS TO IDLE MASTER PROCESSOR 12A
ACPI METHOD TO "EJECT" PROCESSOR
SWITCH TO HOT SPARE 15
106
REESTABLISH LOCKSTEP
18
REESTABLISH LOCKSTEP FOR MASTER 12A AND SLAVE 12B
110
108
REINTRODUCE MASTER PROCESSOR 12A TO OS
ACPI METHOD TO "CHECK FOR" PROCESSOR
HOLD PAIR 12 AS HOT SPARE FOR NEW BOOT PROCESSOR 15
111
107

FIG. 2

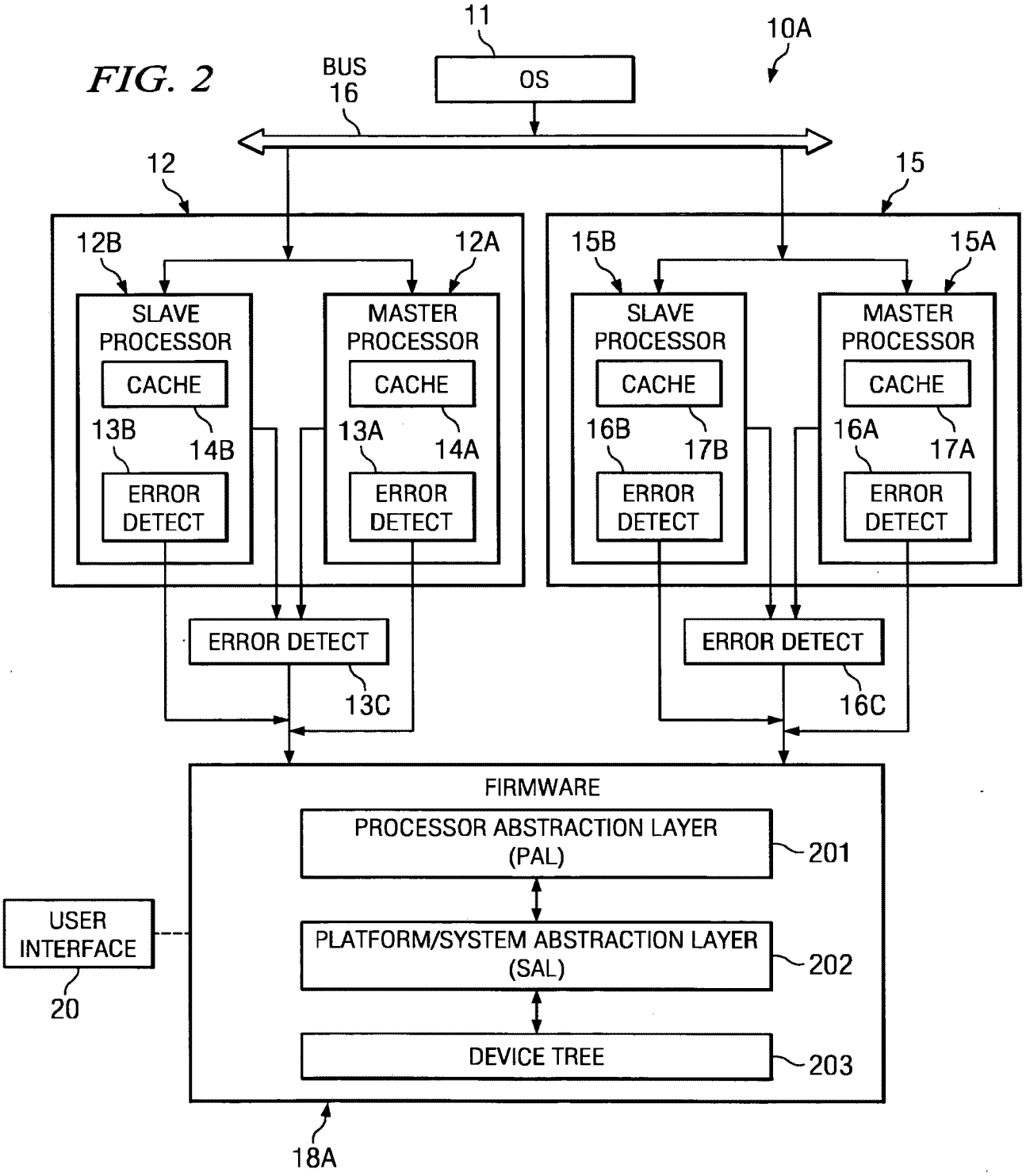

11
10A
BUS
16
OS
12
15
12B
12A
15B
15A
SLAVE PROCESSOR
CACHE
13B
14B
ERROR DETECT
MASTER PROCESSOR
CACHE
13A
14A
ERROR DETECT
SLAVE PROCESSOR
CACHE
16B
17B
ERROR DETECT
MASTER PROCESSOR
CACHE
16A
17A
ERROR DETECT
ERROR DETECT
13C
ERROR DETECT
16C
FIRMWARE
PROCESSOR ABSTRACTION LAYER (PAL)
201
PLATFORM/SYSTEM ABSTRACTION LAYER (SAL)
202
DEVICE TREE
203
USER INTERFACE
20
18A

FIG. 3

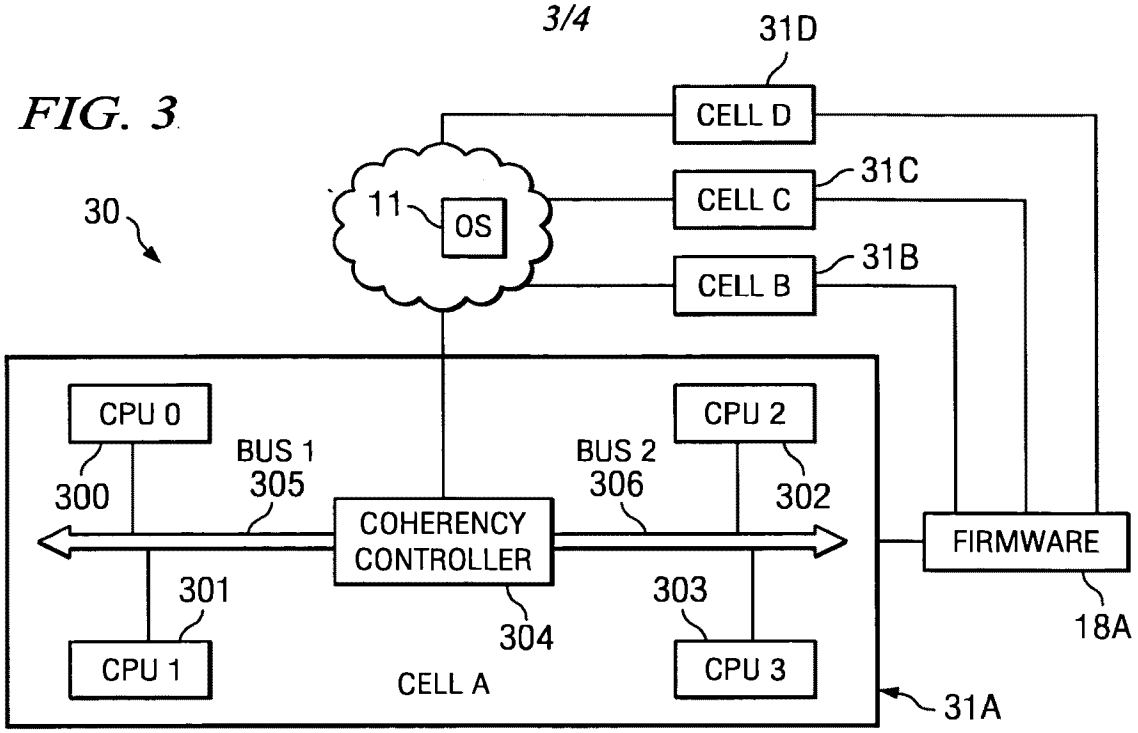

31D
CELL D
31C
CELL C
31B
CELL B
11
OS
30
CPU 0
BUS 1
305
300
CPU 2
BUS 2
306
302
COHERENCY
CONTROLLER
304
FIRMWARE
18A
301
CPU 1
303
CPU 3
CELL A
31A

FIG. 4

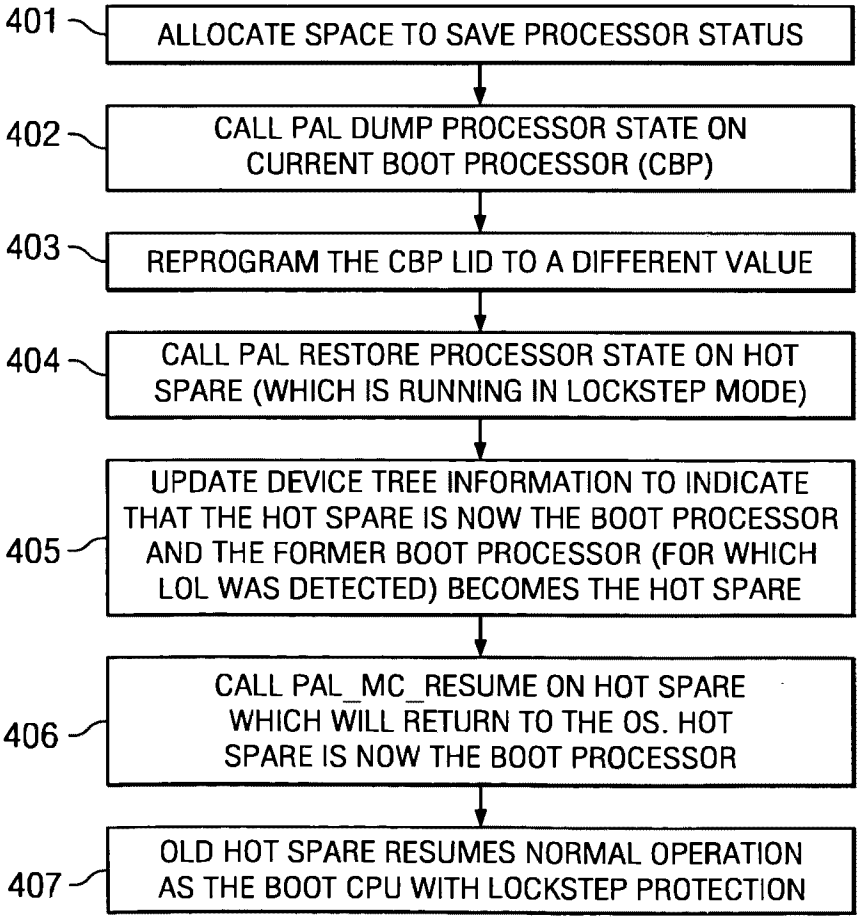

401
ALLOCATE SPACE TO SAVE PROCESSOR STATUS
402
CALL PAL DUMP PROCESSOR STATE ON CURRENT BOOT PROCESSOR (CBP)
403
REPROGRAM THE CBP LID TO A DIFFERENT VALUE
404
CALL PAL RESTORE PROCESSOR STATE ON HOT SPARE (WHICH IS RUNNING IN LOCKSTEP MODE)
405
UPDATE DEVICE TREE INFORMATION TO INDICATE THAT THE HOT SPARE IS NOW THE BOOT PROCESSOR AND THE FORMER BOOT PROCESSOR (FOR WHICH LOL WAS DETECTED) BECOMES THE HOT SPARE
406
CALL PAL_MC_RESUME ON HOT SPARE WHICH WILL RETURN TO THE OS. HOT SPARE IS NOW THE BOOT PROCESSOR
407
OLD HOT SPARE RESUMES NORMAL OPERATION AS THE BOOT CPU WITH LOCKSTEP PROTECTION

FIG. 5

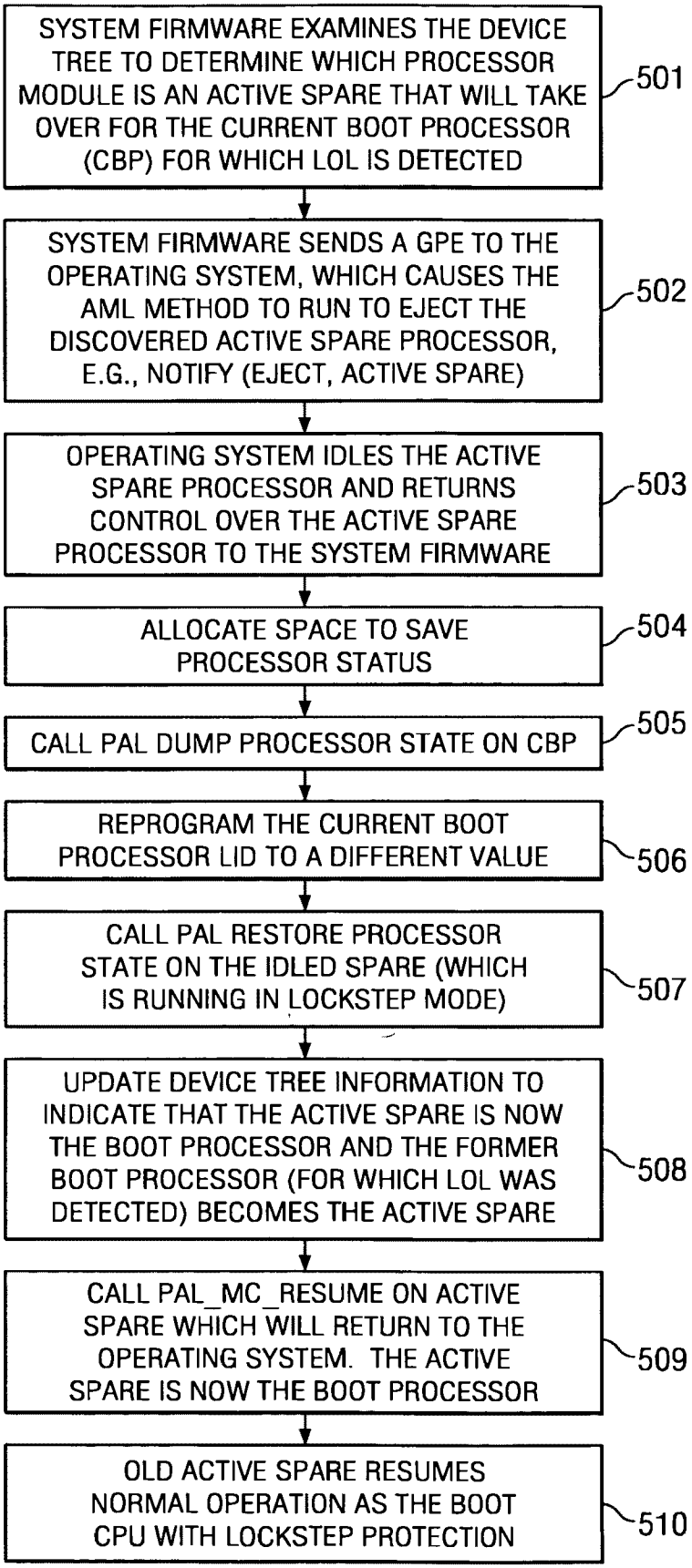
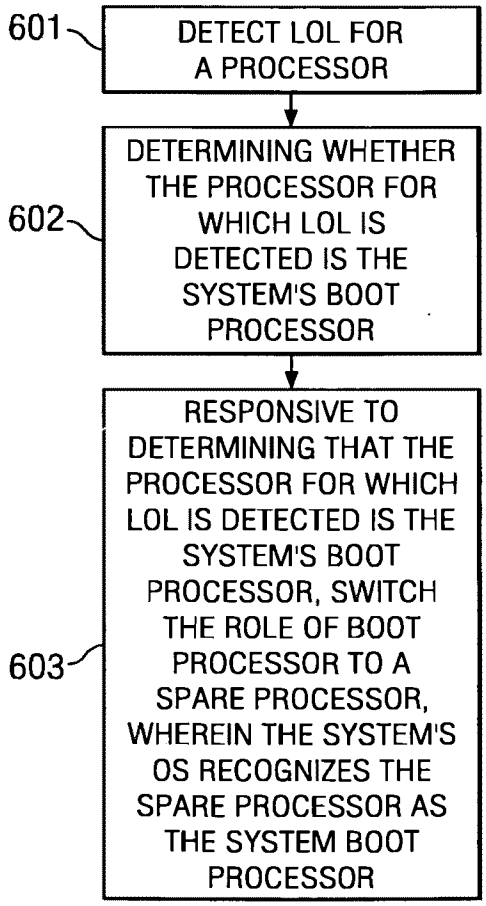

SYSTEM FIRMWARE EXAMINES THE DEVICE TREE TO DETERMINE WHICH PROCESSOR MODULE IS AN ACTIVE SPARE THAT WILL TAKE OVER FOR THE CURRENT BOOT PROCESSOR (CBP) FOR WHICH LOL IS DETECTED
501
SYSTEM FIRMWARE SENDS A GPE TO THE OPERATING SYSTEM, WHICH CAUSES THE AML METHOD TO RUN TO EJECT THE DISCOVERED ACTIVE SPARE PROCESSOR, E.G., NOTIFY (EJECT, ACTIVE SPARE)
502
OPERATING SYSTEM IDLES THE ACTIVE SPARE PROCESSOR AND RETURNS CONTROL OVER THE ACTIVE SPARE PROCESSOR TO THE SYSTEM FIRMWARE
503
ALLOCATE SPACE TO SAVE PROCESSOR STATUS
504
CALL PAL DUMP PROCESSOR STATE ON CBP
505
REPROGRAM THE CURRENT BOOT PROCESSOR LID TO A DIFFERENT VALUE
506
CALL PAL RESTORE PROCESSOR STATE ON THE IDLED SPARE (WHICH IS RUNNING IN LOCKSTEP MODE)
507
UPDATE DEVICE TREE INFORMATION TO INDICATE THAT THE ACTIVE SPARE IS NOW THE BOOT PROCESSOR AND THE FORMER BOOT PROCESSOR (FOR WHICH LOL WAS DETECTED) BECOMES THE ACTIVE SPARE
508
CALL PAL_MC_RESUME ON ACTIVE SPARE WHICH WILL RETURN TO THE OPERATING SYSTEM. THE ACTIVE SPARE IS NOW THE BOOT PROCESSOR
509
OLD ACTIVE SPARE RESUMES NORMAL OPERATION AS THE BOOT CPU WITH LOCKSTEP PROTECTION
510

FIG. 6


601
DETECT LOL FOR A PROCESSOR
602
DETERMINING WHETHER THE PROCESSOR FOR WHICH LOL IS DETECTED IS THE SYSTEM'S BOOT PROCESSOR
603
RESPONSIVE TO DETERMINING THAT THE PROCESSOR FOR WHICH LOL IS DETECTED IS THE SYSTEM'S BOOT PROCESSOR, SWITCH THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR, WHEREIN THE SYSTEM'S OS RECOGNIZES THE SPARE PROCESSOR AS THE SYSTEM BOOT PROCESSOR

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

ATTORNEY DOCKET NO. 200404973-1

As a below named inventor, I hereby declare that:

My residence/post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

System And Method For Switching The Role Of Boot Processor To A Spare Processor Responsive To Detection Of Loss Of Lockstep In A Boot Processor

the specification of which is attached hereto unless the following box is checked:

( ) was filed on _____ as US Application No. or PCT International Application Number _____ and was amended on _____ (if applicable).

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment(s) referred to above. I acknowledge the duty to disclose all information which is material to patentability as defined in 37 CFR 1.56.

Foreign Application(s) and/or Claim of Foreign Priority

I hereby claim foreign priority benefits under Title 35, United States Code Section 119 of any foreign application(s) for patent or inventor(s) certificate listed below and have also identified below any foreign application for patent or inventor(s) certificate having a filing date before that of the application on which priority is claimed:

COUNTRY	APPLICATION NUMBER	DATE FILED	PRIORITY CLAIMED UNDER 35 U.S.C. 119
			YES: ___ NO: ___
			YES: ___ NO: ___

Provisional Application

I hereby claim the benefit under Title 35, United States Code Section 119(e) of any United States provisional application(s) listed below:

APPLICATION NUMBER	FILING DATE

U. S. Priority Claim

I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code Section 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, Section 1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

APPLICATION NUMBER	FILING DATE	STATUS (patented/pending/abandoned)

POWER OF ATTORNEY:

As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith:

Send Correspondence to:

Customer Number 022879

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

Direct Telephone Calls to:

Richard Stokey

(281) 518-3184

Even though the attorney is listed, please associate HP's Customer Number 022879 with this case.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Inventor: Scott L. Michaelis **Citizenship:** U.S.A.

Residence: 1524 Bamburgh Drive, Plano, TX 75075

Post Office Address: 1524 Bamburgh Drive, Plano, TX 75075

Scott L Michaelis 10/20/2004

Inventor's Signature **Date**

DECLARATION AND POWER OF ATTORNEY
FOR PATENT APPLICATION (continued)

ATTORNEY DOCKET NO. 200404973-1

Full Name of # 2 joint inventor: Anurupa Rajkumari Citizenship: India

Residence: 2511 Echo Wood Place, Round Rock, TX 78681

Post Office Address: 2511 Echo Wood Place, Round Rock, TX 78681

Anurupa Rajkumari
Inventor's Signature

10/20/04
Date

Full Name of # 3 joint inventor: William B. McHardy Citizenship: U.S.A.

Residence: 4023 Sunstone Drive, Fort Collins, CO 80525

Post Office Address: 4023 Sunstone Drive, Fort Collins, CO 80525

William B. McHardy
Inventor's Signature

10/19/04
Date

Full Name of # 4 joint inventor: _____ Citizenship: _____

Residence: _____

Post Office Address: _____

Inventor's Signature

Date

Full Name of # 5 joint inventor: _____ Citizenship: _____

Residence: _____

Post Office Address: _____

Inventor's Signature

Date

Full Name of # 6 joint inventor: _____ Citizenship: _____

Residence: _____

Post Office Address: _____

Inventor's Signature

Date

Full Name of # 7 joint inventor: _____ Citizenship: _____

Residence: _____

Post Office Address: _____

Inventor's Signature

Date

Full Name of # 8 joint inventor: _____ Citizenship: _____

Residence: _____

Post Office Address: _____

Inventor's Signature

Date

PATENT APPLICATION SERIAL NO. _____

U.S. DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE
FEE RECORD SHEET

10/27/2004 GWORDOF1 00000043 082025 10972588

01 FC:1001 790.00 DA
02 FC:1201 88.00 DA
03 FC:1202 180.00 DA

PTO-1556
(5/87)

*U.S. Government Printing Office: 2002 — 489-267/69033

PATENT APPLICATION FEE DETERMINATION RECORD
Effective October 1, 2004

Application or Docket Number: 10/972,588

CLAIMS AS FILED - PART I

	(Column 1)	(Column 2)
TOTAL CLAIMS 30		
FOR	NUMBER FILED	NUMBER EXTRA
TOTAL CHARGEABLE CLAIMS	30 minus 20=	* 10
INDEPENDENT CLAIMS	4 minus 3 =	* 1
MULTIPLE DEPENDENT CLAIM PRESENT		☐

* If the difference in column 1 is less than zero, enter "0" in column 2

SMALL ENTITY TYPE ☐		OR	OTHER THAN SMALL ENTITY	
RATE	FEE		RATE	FEE
BASIC FEE	395.00	OR	BASIC FEE	790.00
X$ 9=		OR	X$18=	180
X44=		OR	X88=	88
+150=		OR	+300=	
TOTAL		OR	TOTAL	1,058

CLAIMS AS AMENDED - PART II

AMENDMENT A

	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA
Total	*	Minus	**	=
Independent	*	Minus	***	=
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM				☐

1, 16, 24, 29

SMALL ENTITY		OR	OTHER THAN SMALL ENTITY	
RATE	ADDI-TIONAL FEE		RATE	ADDI-TIONAL FEE
X$ 9=		OR	X$18=	
X44=		OR	X88=	
+150=		OR	+300=	
TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

AMENDMENT B

	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA
Total	*	Minus	**	=
Independent	*	Minus	***	=
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM				☐

RATE	ADDI-TIONAL FEE		RATE	ADDI-TIONAL FEE
X$ 9=		OR	X$18=	
X44=		OR	X88=	
+150=		OR	+300=	
TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

AMENDMENT C

	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA
Total	*	Minus	**	=
Independent	*	Minus	***	=
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM				☐

RATE	ADDI-TIONAL FEE		RATE	ADDI-TIONAL FEE
X$ 9=		OR	X$18=	
X44=		OR	X88=	
+150=		OR	+300=	
TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20."
***If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3."
The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P. O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 200404973-1

IN THE
UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): Scott L. Michaelis et al. **Confirmation No.:**

Application No.: **Examiner:**

Filing Date: **Group Art Unit:**

Title: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

Mail Stop Amendment
Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

INFORMATION DISCLOSURE STATEMENT

Sir:

This Information Disclosure Statement is submitted:

(X) under 37 CFR 1.97(b), or
(Within three months of filing national application; or date of entry of national application; or before mailing date of first office action on the merits; whichever occurs last)

( ) under 37 CFR 1.97(c) together with either a:
( ) Statement under 37 CFR 1.97(e), or
( ) a $180.00 fee under 37 CFR 1.17(p), or
(After the CFR 1.97 (b) time period, but before final action or notice of allowance, whichever occurs first)

( ) under 37 CFR 1.97 (d) together with a:
( ) Statement under 37 CFR 1.97(e)(1) or (2), and
( ) a $180.00 fee set forth in 37 CFR 1.17(p).
(Filed after final action, a notice of allowance, on or before payment of the issue fee)

Please charge to Deposit Account **08-2025** the sum of $0.00 . At any time during the pendency of this application, please charge any fees required or credit any overpayment to Deposit Account **08-2025** pursuant to 37 CFR 1.25.

(X) Applicant(s) submit herewith Form PTO 1449 - Information Disclosure Statement together with any required copies of patents, publications or other information of which applicant(s) are aware, which applicant(s) believe(s) may be material to the examination of this application and for which there may be a duty to disclose in accordance with 37 CFR 1.56.

( ) A concise explanation of the relevance of foreign language patents, foreign language publications and other foreign language information listed on PTO Form 1449, as presently understood by the individuals(s) designated in 37 CFR 1.56 (c) most knowledgeable about the content is given on the attached sheet, or where a foreign language patent is cited in a search report or other action by a foreign patent office in a counterpart foreign application, an English language version of the search report or action which indicates the degree of relevance found by the foreign office is listed on form PTO 1449 and is enclosed herewith.

It is requested that the information disclosed herein be made of record in this application.

"Express Mail" label no. **EV 375345140US**

Date of Deposit **Oct. 25, 2004**

I hereby certify that this is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to: Commissioner for Patents, Alexandria, VA 22313-1450

By _____

Typed Name: **Phyllis Ewing**

Respectfully submitted,

Scott L. Michaelis et al.

By _____

Jody C. Bishop

Attorney/Agent for Applicant(s)
Reg. No. **44,034**

Date: **Oct. 25, 2004**

Telephone No.: **(214) 855-8007**

FORM PTO-1449 LIST OF PATENTS AND PUBLICATIONS FOR APPLICANT'S INFORMATION DISCLOSURE STATEMENT (Use several sheets if necessary)	ATTY. DOCKET NO. 200404973-1	APPLICATION NO.	CONFIRMATION NO.
	APPLICANT Scott L. Michaelis et al.		
	FILING DATE	GROUP	

REFERENCE DESIGNATION **U.S. PATENT DOCUMENTS**

EXAMINER INITIAL		DOCUMENT NUMBER	PUBLICATION DATE	NAME	Pages, Columns, Lines Where Relevant Passages or Figures Appear
	1A	5,226,152	07/06/1993	Klug et al.	
	1B	5,249,188	09/28/1993	McDonald	
	1C	5,751,932	05/12/1998	Horst et al.	
	1D	5,758,058	05/26/1998	Milburn	
	1E	5,764,660	06/09/1998	Mohat	
	1F	6,065,135	05/16/2000	Marshall et al.	
	1G	6,148,348	11/14/2000	Garnett et al.	
	1H	6,473,869	10/29/2002	Bissett et al.	
	1I	6,560,682	05/06/2003	Miller et al.	
	1J	6,604,177	08/05/2003	Kondo et al.	
	1K	6,615,366	09/02/2003	Grochowski et al.	

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	PUBLICATION DATE	NAME OF PATENTEE OR APPLICANT	Pages/Columns/Lines Where Relevant Passages/Figures Appear	Check if Translation attached
	1L					
	1M					
	1N					
	1O					
	1P					

OTHER REFERENCES (including Author, Title, Date, Pertinent Pages, etc.)

	1Q	Attorney Docket No. 200404974-1, filed concurrently herewith, entitled "SYSTEM AND METHOD FOR MAINTAINING IN A MULTI-PROCESSOR SYSTEM A SPARE PROCESSOR THAT IS IN LOCKSTEP FOR USE IN RECOVERING FROM LOSS OF LOCKSTEP FOR ANOTHER PROCESSOR"
	1R	Attorney Docket No. 200404944-1, filed concurrently herewith, entitled "SYSTEM AND METHOD FOR ESTABLISHING A SPARE PROCESSOR FOR RECOVERING FROM LOSS OF LOCKSTEP IN A BOOT PROCESSOR"
	1S	Attorney Docket No. 200404946-1, filed concurrently herewith, entitled "SYSTEM AND METHOD FOR USING INFORMATION RELATING TO A DETECTED LOSS OF LOCKSTEP FOR DETERMINING A RESPONSIVE ACTION"

EXAMINER	DATE CONSIDERED

FORM PTO-1449 LIST OF PATENTS AND PUBLICATIONS FOR APPLICANT'S INFORMATION DISCLOSURE STATEMENT (Use several sheets if necessary)	ATTY. DOCKET NO. 200404973-1	APPLICATION NO.	CONFIRMATION NO.
	APPLICANT Scott L. Michaelis et al.		
	FILING DATE	GROUP	

REFERENCE DESIGNATION **U.S. PATENT DOCUMENTS**

EXAMINER INITIAL		DOCUMENT NUMBER	PUBLICATION DATE	NAME	Pages, Columns, Lines Where Relevant Passages or Figures Appear
	2A	6,625,749	09/23/2003	Quach	
	2B	6,687,851	02/03/2004	Somers et al.	
	2C	6,754,787	06/22/2004	Miller et al.	
	2D	2002/0144177	10/03/2002	Kondo et al.	
	2E	2003/0070050	04/10/2003	Miller et al.	
	2F	2004/0006722	01/08/2004	Safford	
	2G	2004/0019771	01/29/2004	Quach	
	2H	2004/0153857	08/05/2004	Yamazaki et al.	
	2I				
	2J				
	2K				

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	PUBLICATION DATE	NAME OF PATENTEE OR APPLICANT	Pages/Columns/Lines Where Relevant Passages/Figures Appear	Check if Translation attached
	2L					
	2M					
	2N					
	2O					
	2P					

OTHER REFERENCES (including Author, Title, Date, Pertinent Pages, etc.)

	2Q	Attorney Docket No. 200404941-1, filed concurrently herewith, entitled "SYSTEM AND METHOD FOR PROVIDING FIRMWARE RECOVERABLE LOCKSTEP PROTECTION"
	2R	Attorney Docket No. 200404943-1, filed concurrently herewith, entitled "SYSTEM AND METHOD FOR CONFIGURING LOCKSTEP MODE OF A PRCESSOR MODULE"
	2S	Attorney Docket No. 200404971-1, filed concurrently herewith, entitled "SYSTEM AND METHOD FOR REESTABLISHING LOCKSTEP FOR A PROCESSOR MODULE FOR WHICH LOSS OF LOCKSTEP IS DETECTED"

EXAMINER	DATE CONSIDERED

FORM PTO-1449 LIST OF PATENTS AND PUBLICATIONS FOR APPLICANT'S INFORMATION DISCLOSURE STATEMENT (Use several sheets if necessary)	ATTY. DOCKET NO. 200404973-1	APPLICATION NO.	CONFIRMATION NO.
	APPLICANT Scott L. Michaelis et al.		
	FILING DATE	GROUP	

REFERENCE DESIGNATION **U.S. PATENT DOCUMENTS**

EXAMINER INITIAL		DOCUMENT NUMBER	PUBLICATION DATE	NAME	Pages, Columns, Lines Where Relevant Passages or Figures Appear
	3A				
	3B				
	3C				
	3D				
	3E				
	3F				
	3G				
	3H				
	3I				
	3J				
	3K				

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	PUBLICATION DATE	NAME OF PATENTEE OR APPLICANT	Pages/Columns/Lines Where Relevant Passages/Figures Appear	Check if Translation attached
	3L					
	3M					
	3N					
	3O					
	3P					

OTHER REFERENCES (including Author, Title, Date, Pertinent Pages, etc.)

	3Q	Attorney Docket No. 200404970-1, filed concurrently herewith, entitled "SYSTEM AND METHOD FOR SYSTEM FIRMWARE CAUSING AN OPERATING SYSTEM TO IDLE A PROCESSOR"
	3R	Attorney Docket No. 200404972-1, filed concurrently herewith, entitled "SYSTEM AND METHOD FOR REINTRODUCING A PROCESSOR MODULE TO AN OPERATING SYSTEM AFTER LOCKSTEP RECOVERY"
	3S	

EXAMINER	DATE CONSIDERED

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P. O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 200404973-1

IN THE U.S. PATENT AND TRADEMARK OFFICE
Patent Application Transmittal Letter

22264 U.S. PTO
10/972588

Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Sir:

Transmitted herewith for filing under 37 CFR 1.53(b) is a(n): (X) Utility ( ) Design
(X) original patent application,
( ) continuation-in-part application

INVENTOR(S): Scott L. Michaelis et al.

TITLE: System And Method For Switching The Role Of Boot Processor To A Spare Processor Responsive To Detection Of Loss Of Lockstep In A Boot Processor

Enclosed are:

(X) The Declaration and Power of Attorney. (X) signed ( ) unsigned or partially signed

(X) 4 sheets of drawings (one set) ( ) Associate Power of Attorney

( ) Form PTO-1449 (X) Information Disclosure Statement and Form PTO-1449

( ) Priority document(s) ( ) (Other) _____ (fee $_____)

CLAIMS AS FILED BY OTHER THAN A SMALL ENTITY				
(1) FOR	(2) NUMBER FILED	(3) NUMBER EXTRA	(4) RATE	(5) TOTALS
TOTAL CLAIMS	30 — 20	10	X $18	$ 180
INDEPENDENT CLAIMS	4 — 3	1	X $88	$ 88
ANY MULTIPLE DEPENDENT CLAIMS	0		$300	$ 0
BASIC FEE: Design ($350.00); Utility ($790.00)				$ 790
TOTAL FILING FEE				$ 1,058
OTHER FEES				$
TOTAL CHARGES TO DEPOSIT ACCOUNT				$ 1,058

Charge $ 1,058 to Deposit Account 08-2025. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees. A duplicate copy of this sheet is enclosed.

"Express Mail" label no. EV 375345140US

Date of Deposit Oct. 25, 2004

I hereby certify that this is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to: Commissioner for Patents, Alexandria, VA 22313-1450.

By _____

Typed Name: Phyllis Ewing

Respectfully submitted,

Scott L. Michaelis et al.

By _____

Jody C. Bishop

Attorney/Agent for Applicant(s)
Reg. No. 44,034

Date: Oct. 25, 2004

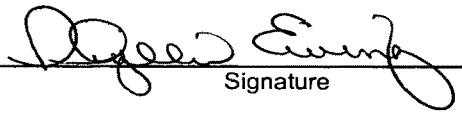
Telephone No.: (214) 855-8007

Certificate of Express Mailing Under 37 CFR 1.10

I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail, Airbill No. EV 375345140US in an envelope addressed to:

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

on October 25, 2004
Date



Signature

Phyllis Ewing
Typed or printed name of person signing Certificate

Registration Number, if applicable

(214) 855-8000
Telephone Number

Note: Each paper must have its own certificate of mailing, or this certificate must identify each submitted paper.

Assignment (copy) 2 pages
Assignment Recordation Cover Sheet - 2 pages (1 orig. + 1 copy)
Certificate of Express Mail - 1 page



Intel® IA-64 Architecture Software Developer's Manual

Volume 2: IA-64 System Architecture

Revision 1.1

July 2000

Document Number: 245318-002

Firmware Overview 24

The IA-64 architecture defines three firmware layers: Processor Abstraction Layer (PAL), System Abstraction Layer (SAL), and Extensible Firmware Interface (EFI).

The PAL, SAL and EFI layers work together to handle the reset abort event. The reset abort handling performs processor and system initialization for operating system (OS) boot and provides a legacy-free API to the operating system loader. The PAL and SAL firmware layers work together to handle machine check aborts (MCA), initialization events (INIT), and platform management interrupt (PMI) handling. All three firmware layers also provide runtime procedure calls to abstract processor and platform functions that may vary across implementations.

This chapter will provide an overview of the firmware layers and how the firmware layers interact with each other as well as with the operating system. For the full architecture specifications of the PAL firmware please refer to Section 11, "IA-64 Processor Abstraction Layer". For full architecture specifications on SAL and EFI firmware layers please refer to Section 12.2, ".Related Documents" on page 12-3.

The PAL layer is developed by Intel Corporation and delivered with the processor. The SAL and EFI firmware is developed by the platform manufacturer and provide a means of supporting value added platform features from different vendors.

The interaction of the various functional firmware blocks with the processor, platform and operating system is shown in Figure 24-1, "Firmware Model" on page 24-2.

24.1 Processor Boot Flow Overview

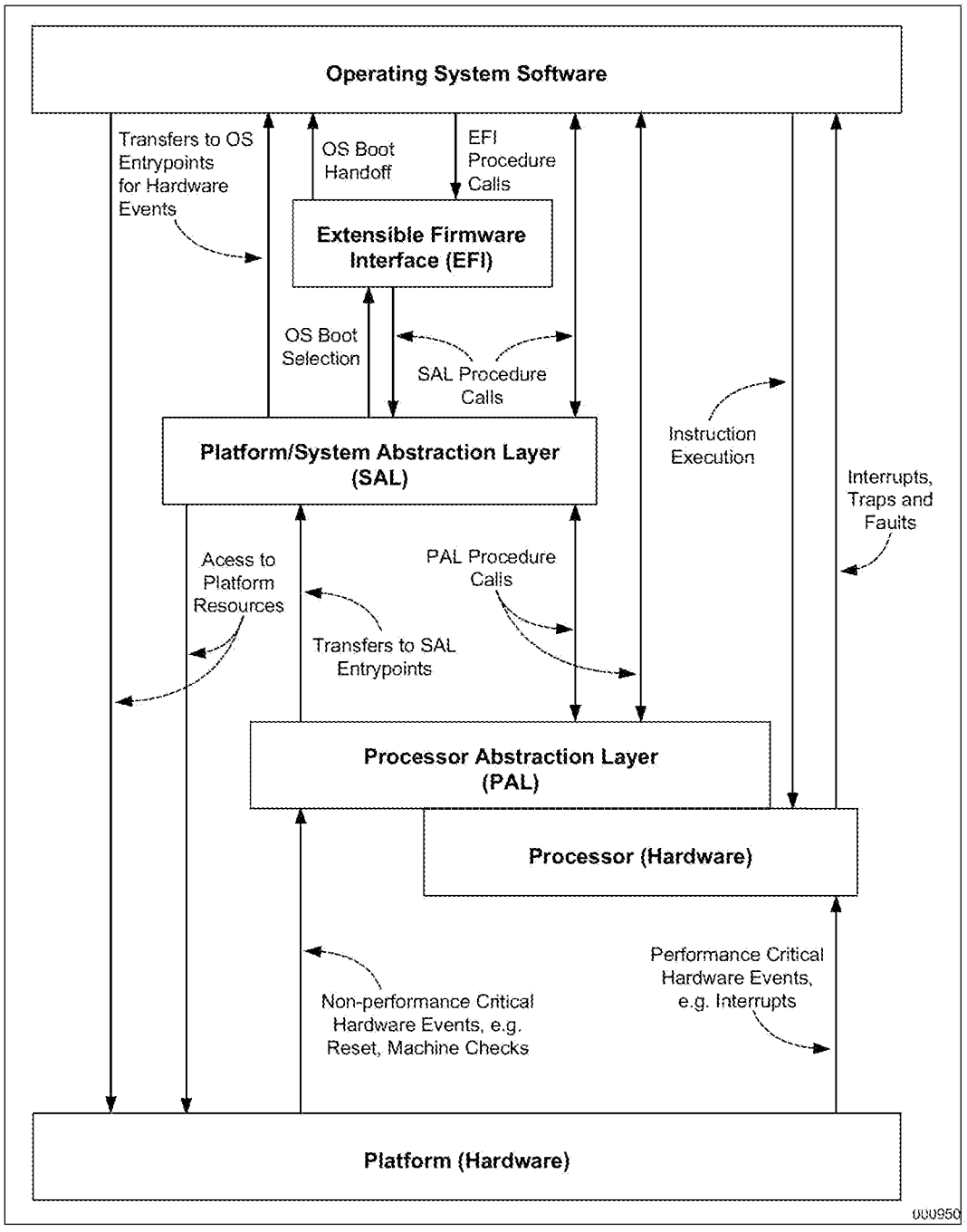
24.1.1 Firmware Boot Flow

Upon detection of a reset event on an IA-64 processor, execution begins at an architected entry point inside of PAL. This PAL code will verify the integrity of the PAL code and may perform some basic processor testing. PAL will then branch to an entry point within the SAL firmware. This first branch to SAL is to determine if a firmware update is needed requiring re-programming of the firmware code. If no firmware update is needed SAL will branch back to PAL.

PAL now performs additional processor testing and initialization. These first processor tests are performed without platform memory. PAL indicates the outcome of the testing and branches to an entry point within SAL firmware for the second time. SAL will now begin platform testing and initialization.

The order of steps within the SAL firmware is platform implementation dependent and may vary. In general, the SAL firmware selects a Bootstrap processor (BSP) in multi-processor (MP) configurations early in the boot sequence. Next, SAL will find and initialize memory and invoke PAL procedures to conduct additional processor tests to ensure the health of the processors. SAL then initializes the system fabric and platform devices. SAL will display the progress of the boot on the video output device and permit the user to change the system configuration.

Figure 24-1. Firmware Model



The SAL firmware layer hands off control to the EFI firmware layer which incorporates a Boot Manager. The EFI firmware specification [EFI] enables booting from a variety of mass storage devices such as hard disk, CD, DVD as well as remote boot via a network. At a minimum, one of the mass storage devices contains an EFI system partition.

The EFI Boot Manager displays the list of operating system choices and permits the user to select the operating system for booting. To support this functionality, the OS setup program stores the

boot paths of the OS loaders and boot options in non-volatile storage managed by the EFI. The EFI reserves the environment variables Boot#### (#### represents values 0000 to 0xFFFF) for this purpose. The OS setup program must also store the OS loader binary images within the EFI System Partition. The EFI Boot Manager will also allow the user to add boot options, delete boot options, launch an EFI application, and set the auto-boot time out value.

The EFI System Partition also contains EFI drivers that will be loaded by the EFI firmware prior to transfer of control to an IA-64 OS loader. The floating-point software assist (FPSWA) library is included in these EFI drivers. The FPSWA library may be invoked by the OS during floating-point exception faults and traps. Please see Section 19.1.1, "The Software Assistance Exceptions (Faults and Traps)" on page 19-1 for more information on the usage of this library.

If the user elects to boot an IA-32 operating system, the EFI will load 512 bytes of the first level boot code (Master Boot Record in the case of disk devices) at location 0x7C00. Next, EFI will remove its memory footprint and returns to the SAL firmware. The SAL will then invoke a PAL procedure to set up the IA-32 System environment and jump to the boot code at 0x7C00. The boot code will load an IA-32 OS loader which, in turn, loads and transfers control to the IA-32 OS kernel.

If the user elects to boot an IA-64 operating system, the EFI loads the appropriate OS loader from the EFI System Partition and passes control to it. The OS loader will load other files including the OS kernel from an OS partition using the EFI boot services which provides a legacy free API interface to the OS loader. EFI uses SAL to access low level platform resources. The interfaces between EFI and SAL are platform firmware implementation dependent and not relevant for the OS loader developers.

The OS loader can obtain information about the memory map usage of the firmware by making the EFI procedure call GetMemoryMap(). This procedure provides information related to the size and attributes of the memory regions currently used by firmware.

The OS loader will then jump to the OS kernel that takes control of the system. Until this point, SAL retained control of key system resources such as the Interrupt Vector Table and provided the necessary interrupt, trap and fault handlers.

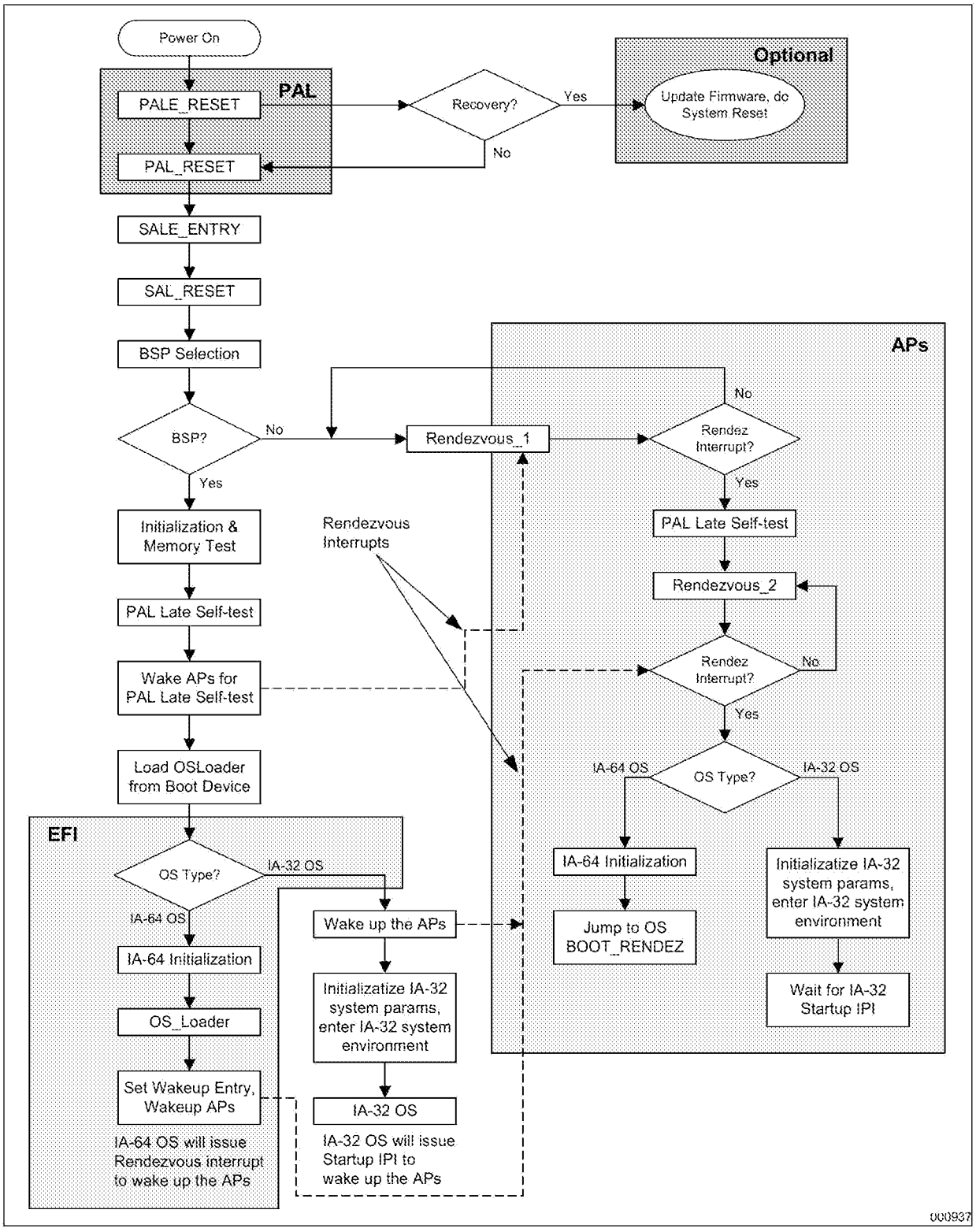
Figure 24-2, "Control Flow of Boot Process in a Multi-processor Configuration" on page 24-4 depicts the booting steps in a MP configuration.

24.1.2 Operating System Boot Steps

The firmware will initialize the processor(s) and platform to a specific state before handing off to the operating system boot loader. The boot loader is then responsible for copying the operating system from some storage medium into memory for running. Once this is done the operating system will need to initialize some key registers before entering into a higher level language code such as C. This section will describe code that an OS will need to execute in order to initialize system registers for preparing an OS to run in virtual mode and handle interrupts. Appendix B, "Code Examples" provides the IA-64 sample assembly code described in this section.

Assuming the specific operating system boot loader hands off to the OS kernel in physical mode, the operating system should first disable interrupts and interrupt collection via the PSR. This is done to avoid taking external interrupts from timers, etc and also prepares for writing specific system registers that require PSR.ic to be 0 when written.

Figure 24-2. Control Flow of Boot Process in a Multi-processor Configuration



Next the operating system startup code invalidates the ALAT via the `invala` instruction. The invala in complete form will invalidate all entries in the ALAT.

The register stack should be invalidated. This can be done by setting the Register Stack Configuration Register (RSC) to zero followed by a loadrs instruction. Setting the RSC to zero will put the register stack in enforced lazy mode and set the RSC.loadrs, load distance to tear point, to zero. The loadrs will invalidate all stacked registers outside current frame.

The region registers and protection key registers are then initialized with operating system implementation dependent values. For example, the OS will initialize the region register with a preferred page size. It would also disable the VHPT until it was ready for it. In the example, all region registers are initialized with an 8-KB page size.

An OS must setup a kernel stack pointer and backing store pointer for the register stack. The stack pointer (GR12) is set to the OS kernel stack area with scratch space to cover calling conventions. AR.RSC must be set to enforced lazy mode before writing to the bspstore register. Initializing the bspstore has effects on all 3 RSE pointers (BSP, BSPSTORE, and RSE.BspLoad).

In order for the operating systems to handle interruptions, the operating system interrupt vector table base address must be set up. The size of the vector table is 32K bytes and is 32K byte aligned. Setting the location of the table is accomplished by moving the address into CR.IVA.

Operating systems setup system address translations for the kernel text and data by using the translation insertion format described in Section 4.1.1.5, "Translation Insertion Format". A combination of a general register, Interruption TLB Insertion Register (ITIR), and the Interruption Faulting Address register (IFA) are used to insert entries into the TLB. To void TLB faults on specific text and data areas the operating system can lock critical virtual memory translations in the TLB by use of Translation Register (TR) section of the TLB. The entries are placed into a TR via the Insert Translation Register (itr) instruction. The translation will remain unless the software issues the Purge Translation (ptr) instruction. Other important areas might be locked also, such as entries for memory mapped I/O, etc.

After the initial translations have been entered, the OS can make final preparations for enabling virtual addressing. The OS needs to set several important bits in the IPSR, such as data address translation (dt), register stack translation (rt), instruction address translation (it), enabling interruption collection (ic), and setting the specific register bank (bn).

The Default Control Register (DCR) specifies the default parameters for PSR values on interruption, some additional global controls, and whether speculative load faults can be deferred. The example defers all speculation faults. Also, if the operating system is utilizing the performance monitors then the DCR.pp bit should be set so that on interruption the PSR.pp bit will be set.

The global pointer (GR1) should point to the global data area. It must be setup properly before using higher level languages such as C. The startup code should also set the following registers to zero, the Interruption Function State (CR.IFS, to set frame marker to zero), and AR.RNAT (to make sure no NaT bits are set before OS kernel begins using the RSE.

Before enabling virtual addressing, the Interruption Instruction Bundle Pointer (IIP) is set to point a virtual address. This is done so when the return from interruption instruction (`rfi`) is executed the instruction fetched will have a virtual address. The rfi will switch modes based on IPSR values which are moved into the PSR. The IIP value becomes the new IP.

PTO/SB/08a (01-10)
Approved for use through 07/31/2012. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Not for submission under 37 CFR 1.99)

Application Number	
Filing Date	2023-09-01
First Named Inventor	Scott L. Michaelis
Art Unit	
Examiner Name	Unassigned
Attorney Docket Number	185945.012600

U.S.PATENTS Remove

Examiner Initial*	Cite No	Patent Number	Kind Code[1]	Issue Date	Name of Patentee or Applicant of cited Document	Pages,Columns,Lines where Relevant Passages or Relevant Figures Appear
	1	7624302		2009-11-24	Michaelis et al.	
	2	7085959		2022-06-17	Safford	
	3	7328371		2008-02-05	Kalyanasundharam et al.	
	4	5450576		1995-09-12	Kennedy et al.	
	5	6625749		2019-12-25	Quach	
	6	7191328		2007-03-13	Hobson	
	7	5491788		2014-05-14	Cepulis et al.	
	8	5915082		2016-05-11	Marshall	

If you wish to add additional U.S. Patent citation information please click the Add button. Add

INFORMATION DISCLOSURE STATEMENT BY APPLICANT (Not for submission under 37 CFR 1.99)		
	Application Number	
	Filing Date	2023-09-01
	First Named Inventor	Scott L. Michaelis
	Art Unit	
	Examiner Name	Unassigned
	Attorney Docket Number	185945.012600

U.S.PATENT APPLICATION PUBLICATIONS							Remove
Examiner Initial*	Cite No	Publication Number	Kind Code[1]	Publication Date	Name of Patentee or Applicant of cited Document	Pages,Columns,Lines where Relevant Passages or Relevant Figures Appear	
	1	20040123201		2006-05-30	Nguyen et al.		
	2	20050172164		2007-07-31	Fox et al.		
	3						

If you wish to add additional U.S. Published Application citation information please click the Add button. Add

FOREIGN PATENT DOCUMENTS								Remove
Examiner Initial*	Cite No	Foreign Document Number[3]	Country Code[2] i	Kind Code[4]	Publication Date	Name of Patentee or Applicant of cited Document	Pages,Columns,Lines where Relevant Passages or Relevant Figures Appear	T[5]
	1							☐

If you wish to add additional Foreign Patent Document citation information please click the Add button Add

NON-PATENT LITERATURE DOCUMENTS			Remove
Examiner Initials*	Cite No	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc), date, pages(s), volume-issue number(s), publisher, city and/or country where published.	T[5]
	1	U.S. File History of the '302 Patent ("U.S. File History")	☐
	2	Declaration of Andrew Wolfe ("Wolfe Decl.")	☐

INFORMATION DISCLOSURE STATEMENT BY APPLICANT (Not for submission under 37 CFR 1.99)		
	Application Number	
	Filing Date	2023-09-01
	First Named Inventor	Scott L. Michaelis
	Art Unit	
	Examiner Name	Unassigned
	Attorney Docket Number	185945.012600

	3	Wendy Bartlett, et. al., Commercial Fault Tolerance: A Tale of Two Systems, IEEE Transactions on Dependable and Secure Computing, Vol. 1, No. 1 (January-March 2004) ("Bartlett")	☐
	4	Intel, Extensible Firmware Interface Specification, Version 1.10 (Dec. 1, 2002) ("Intel EFI Specification")	☐
	5	Bozidar Levi, UNIX Administration, A Comprehensive Sourcebook for Effective Systems and Network Management (2002) ("Levi")	☐
	6	Intel IA-64 Architecture Software Developer's Manual, Revision 1.1 (July 2000) ("Intel Itanium Manual")	☐
	7	Intel Multiprocessor Specification, Version 1.4 (May 1997) ("Intel Multiprocessor Specification")	☐

If you wish to add additional non-patent literature document citation information please click the Add button Add

EXAMINER SIGNATURE

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through a citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

[1] See Kind Codes of USPTO Patent Documents at www.USPTO.GOV or MPEP 901.04. [2] Enter office that issued the document, by the two-letter code (WIPO Standard ST.3). [3] For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [4] Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [5] Applicant is to place a check mark here if English language translation is attached.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT (**Not for submission under 37 CFR 1.99)**		
	Application Number	
	Filing Date	2023-09-01
	First Named Inventor	Scott L. Michaelis
	Art Unit	
	Examiner Name	Unassigned
	Attorney Docket Number	185945.012600

CERTIFICATION STATEMENT

Please see 37 CFR 1.97 and 1.98 to make the appropriate selection(s):

☐ That each item of information contained in the information disclosure statement was first cited in any communication from a foreign patent office in a counterpart foreign application not more than three months prior to the filing of the information disclosure statement. See 37 CFR 1.97(e)(1).

OR

☐ That no item of information contained in the information disclosure statement was cited in a communication from a foreign patent office in a counterpart foreign application, and, to the knowledge of the person signing the certification after making reasonable inquiry, no item of information contained in the information disclosure statement was known to any individual designated in 37 CFR 1.56(c) more than three months prior to the filing of the information disclosure statement. See 37 CFR 1.97(e)(2).

☐ See attached certification statement.

☐ The fee set forth in 37 CFR 1.17 (p) has been submitted herewith.

☒ A certification statement is not submitted herewith.

SIGNATURE

A signature of the applicant or representative is required in accordance with CFR 1.33, 10.18. Please see CFR 1.4(d) for the form of the signature.

Signature	/Greg Lunt/	Date (YYYY-MM-DD)	2023-09-01
Name/Print	Greg Lunt	Registration Number	57354

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 1 hour to complete, including gathering, preparing and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C. 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether the Freedom of Information Act requires disclosure of these record s.

2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.

3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.

4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).

5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.

6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).

7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (i.e., GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.

8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspections or an issued patent.

9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

UNITED STATES
PATENT AND TRADEMARK OFFICE

P.O. Box 1450
Alexandria, VA 22313 - 1450
www.uspto.gov

ELECTRONIC PAYMENT RECEIPT

APPLICATION #	RECEIPT DATE / TIME	ATTORNEY DOCKET #
90/019,243	**09/01/2023 06:38:04 PM ET**	**185945.012600**

Title of Invention

SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

Application Information

APPLICATION TYPE	Utility / ex parte reexam	PATENT #	7624302
CONFIRMATION #	1079	FILED BY	Judi Cordo
PATENT CENTER #	62727465	AUTHORIZED BY	Andrew Sommer
CUSTOMER #	56436	FILING DATE	-
CORRESPONDENCE ADDRESS	-	FIRST NAMED INVENTOR	Scott L. Michaelis

Payment Information

PAYMENT METHOD	PAYMENT TRANSACTION ID	PAYMENT AUTHORIZED BY
DA / 506990	E202391I40338183	Judi Cordo

FEE CODE	DESCRIPTION	ITEM PRICE($)	QUANTITY	ITEM TOTAL($)
1812	REQUEST FOR EX PARTE REEXAM-NON-STREAMLINED	12600.00	1	12600.00
			TOTAL AMOUNT:	**$12,600.00**

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

US006625749B1

(12) **United States Patent**
Quach

(10) **Patent No.: US 6,625,749 B1**
(45) **Date of Patent: Sep. 23, 2003**

(54) **FIRMWARE MECHANISM FOR CORRECTING SOFT ERRORS**

(75) Inventor: **Nhon Quach**, San Jose, CA (US)

(73) Assignee: **Intel Corporation**, Santa Clara, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/469,963**

(22) Filed: **Dec. 21, 1999**

(51) **Int. Cl.**[7] **G06F 11/00**
(52) **U.S. Cl.** **714/10**; 712/220
(58) **Field of Search** 714/12, 11, 10; 712/229, 20, 43; 700/3, 4, 5

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,453,215	A		6/1984	Reid
4,912,707	A	*	3/1990	Kogge et al. 712/228
5,475,856	A	*	12/1995	Kogge 700/12
5,504,859	A		4/1996	Gustafson et al.
5,530,802	A	*	6/1996	Fuchs et al. 714/17
5,664,214	A	*	9/1997	Taylor et al. 712/20
5,748,873	A	*	5/1998	Ohguro et al. 714/10
6,279,119	B1	*	8/2001	Bissett et al. 703/23
6,393,582	B1	*	5/2002	Klecka et al. 714/10

FOREIGN PATENT DOCUMENTS

EP	0315303	5/1989
EP	0411805 A2	2/1991

OTHER PUBLICATIONS

Keith Diefendorff, Microprocessor Report,Nov. 15, 1999, pp. 8, vol. 13, No. 15.
Keith Diefendorff, Power4 Focuses on Memory Bandwidth, Oct. 6, 1999, pp. 11–17.

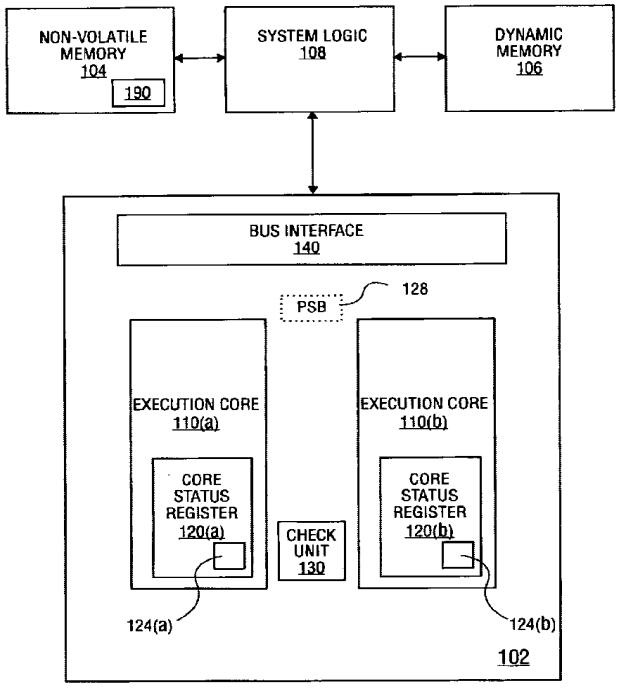
* cited by examiner

Primary Examiner—Robert Beausoliel
Assistant Examiner—Christopher S. McCarthy
(74) *Attorney, Agent, or Firm*—Blakely, Sokoloff, Taylor & Zafman LLP

(57) **ABSTRACT**

A computer system includes processor having dual execution cores and a non-volatile memory that stores an error recovery routine. The processor's execution cores operate in lock step when the processor is in a redundant execution mode, and they operate independently when the processor is in a split execution mode. The error recovery routine is invoked when the processor detects a soft error while operating in the redundant execution mode. The error recovery routine switches the processor to split execution mode. In split mode, each execution core saves uncorrupted processor state data to a designated memory location and updates any corrupted data with corresponding processor state data from the other execution core. The error recovery routine returns the processor to redundant mode, initializes each execution core with the recovered processor state data, and returns control of the processor to the program thread that was executing when the soft error was detected.

19 Claims, 9 Drawing Sheets



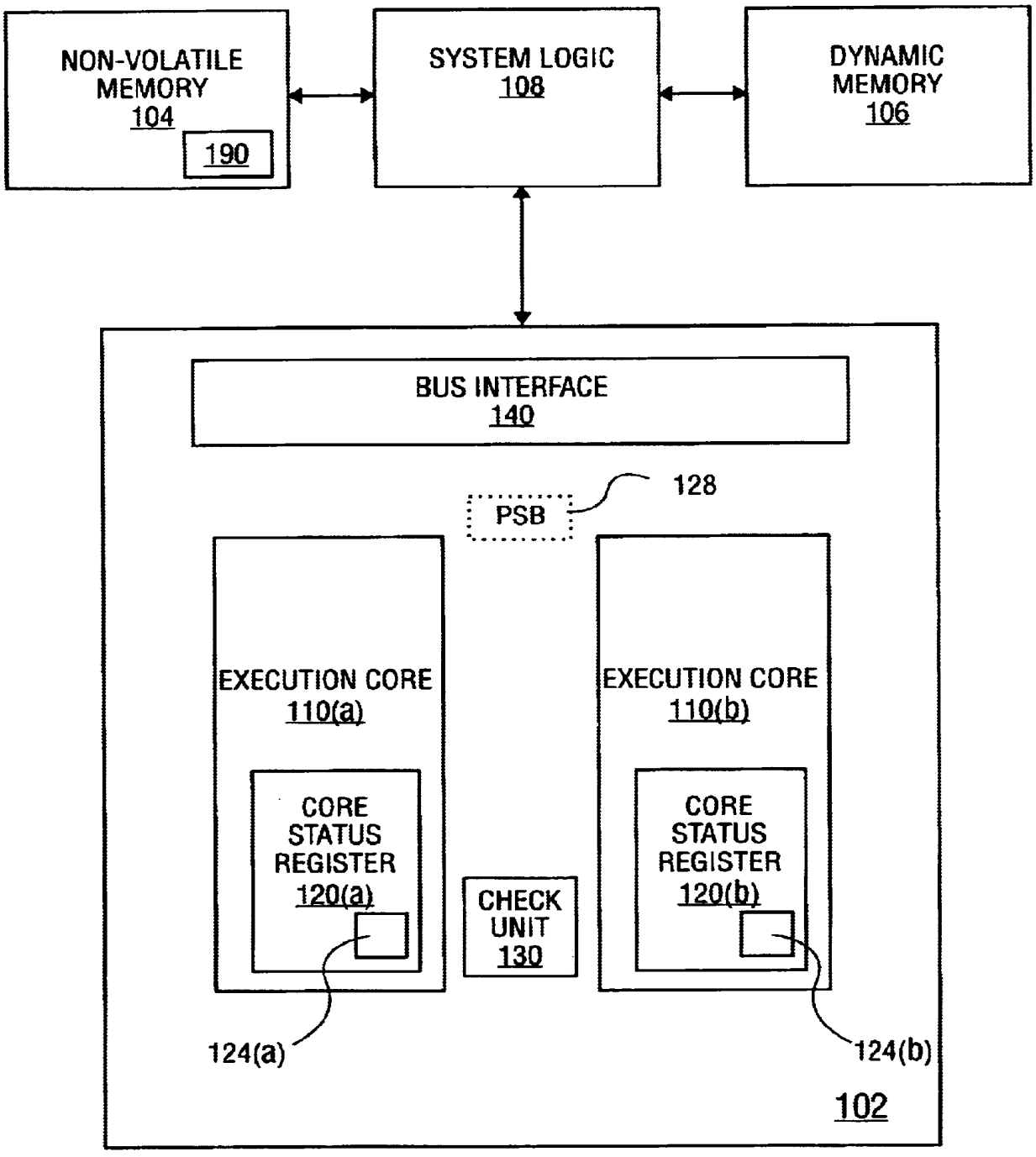

NON-VOLATILE MEMORY
104
190
SYSTEM LOGIC
108
DYNAMIC MEMORY
106
BUS INTERFACE
140
128
PSB
EXECUTION CORE
110(a)
EXECUTION CORE
110(b)
CORE STATUS REGISTER
120(a)
CORE STATUS REGISTER
120(b)
CHECK UNIT
130
124(a)
124(b)
102

FIG. 1

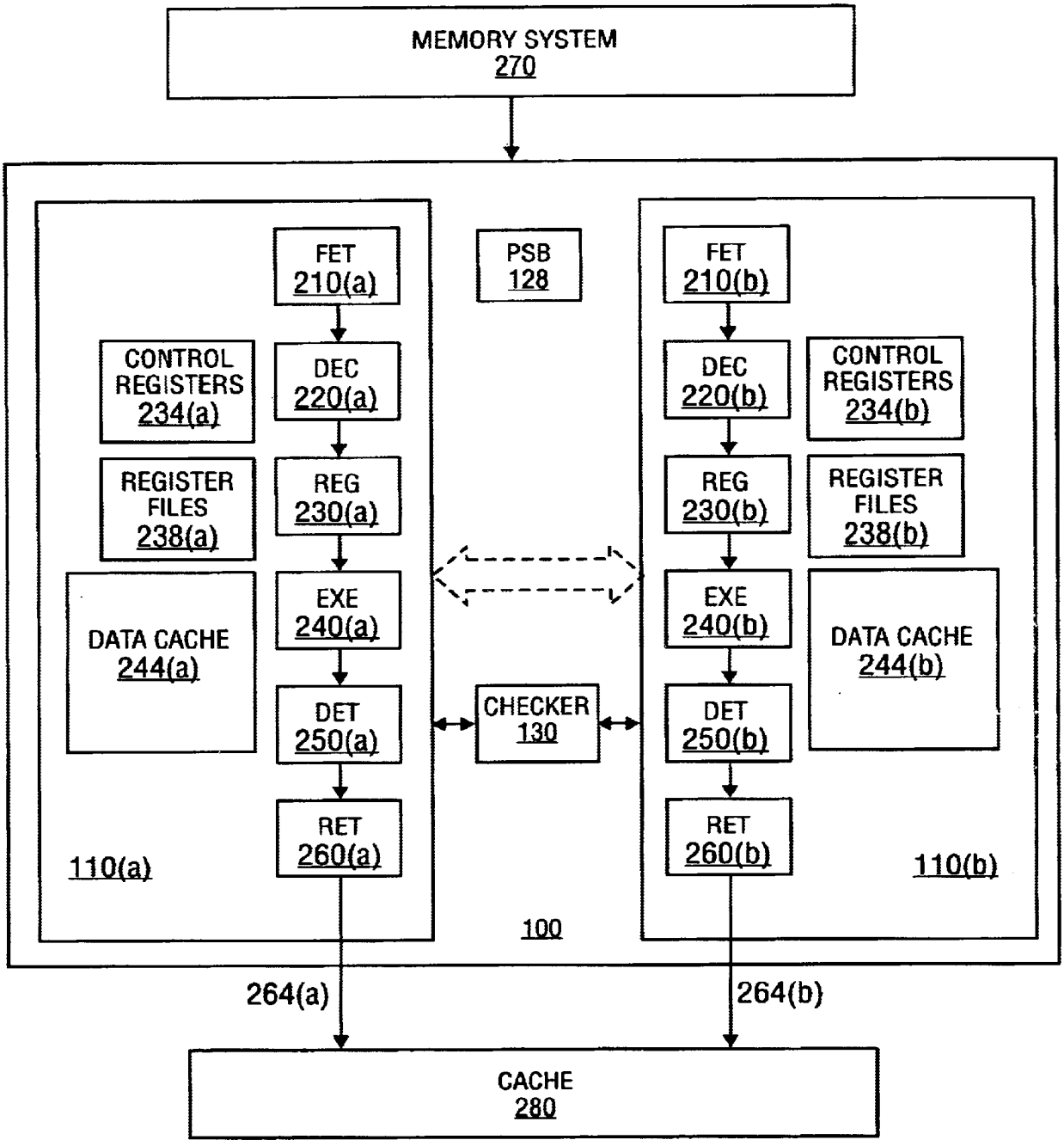
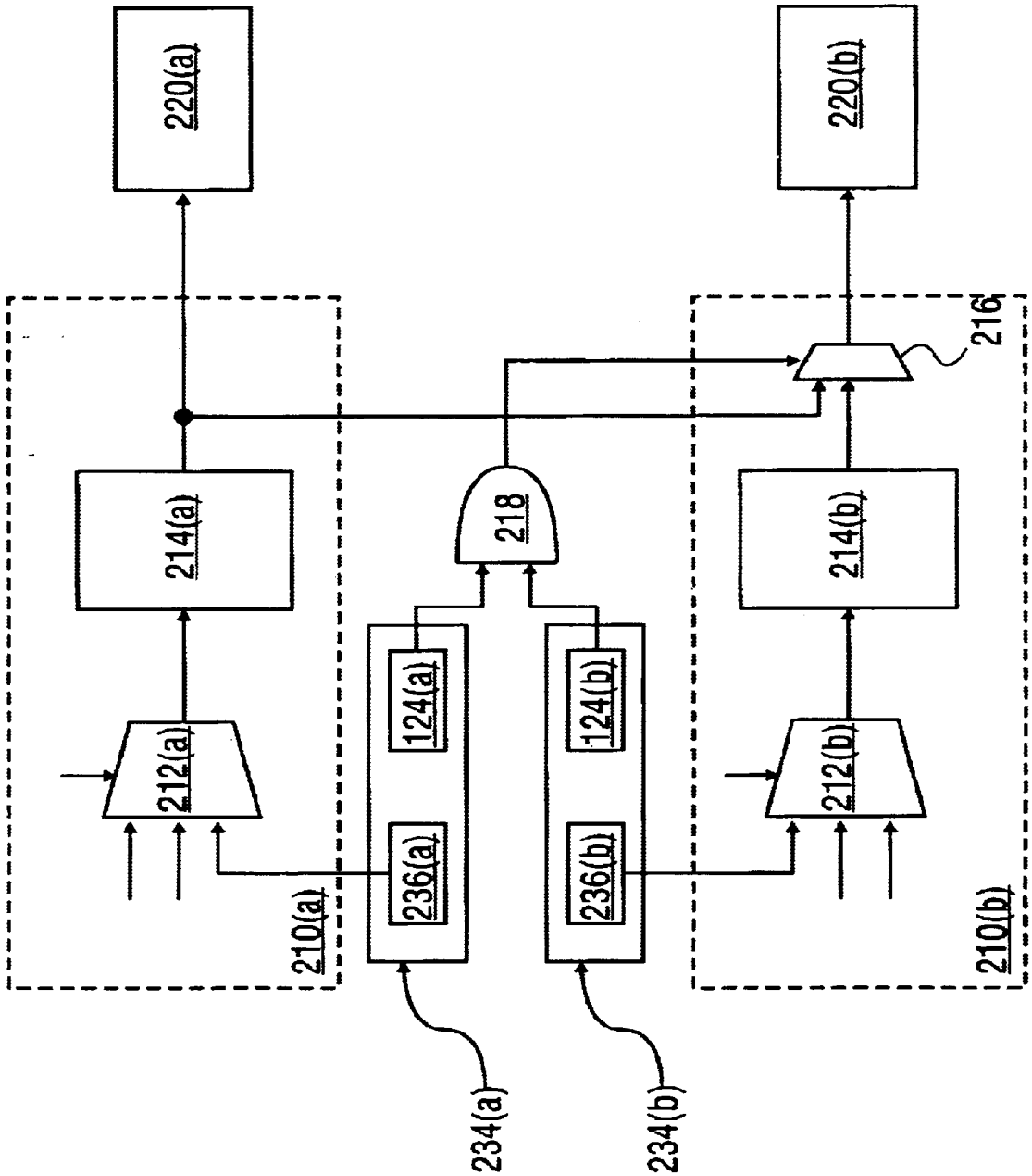

MEMORY SYSTEM
270
FET
210(a)
PSB
128
FET
210(b)
CONTROL
REGISTERS
234(a)
DEC
220(a)
DEC
220(b)
CONTROL
REGISTERS
234(b)
REGISTER
FILES
238(a)
REG
230(a)
REG
230(b)
REGISTER
FILES
238(b)
DATA CACHE
244(a)
EXE
240(a)
EXE
240(b)
DATA CACHE
244(b)
DET
250(a)
CHECKER
130
DET
250(b)
RET
260(a)
RET
260(b)
110(a)
110(b)
100
264(a)
264(b)
CACHE
280

FIG. 2A

FIG. 2B


220(a)
220(b)
214(a)
214(b)
218
216
212(a)
212(b)
124(a)
124(b)
236(a)
236(b)
210(a)
210(b)
234(a)
234(b)

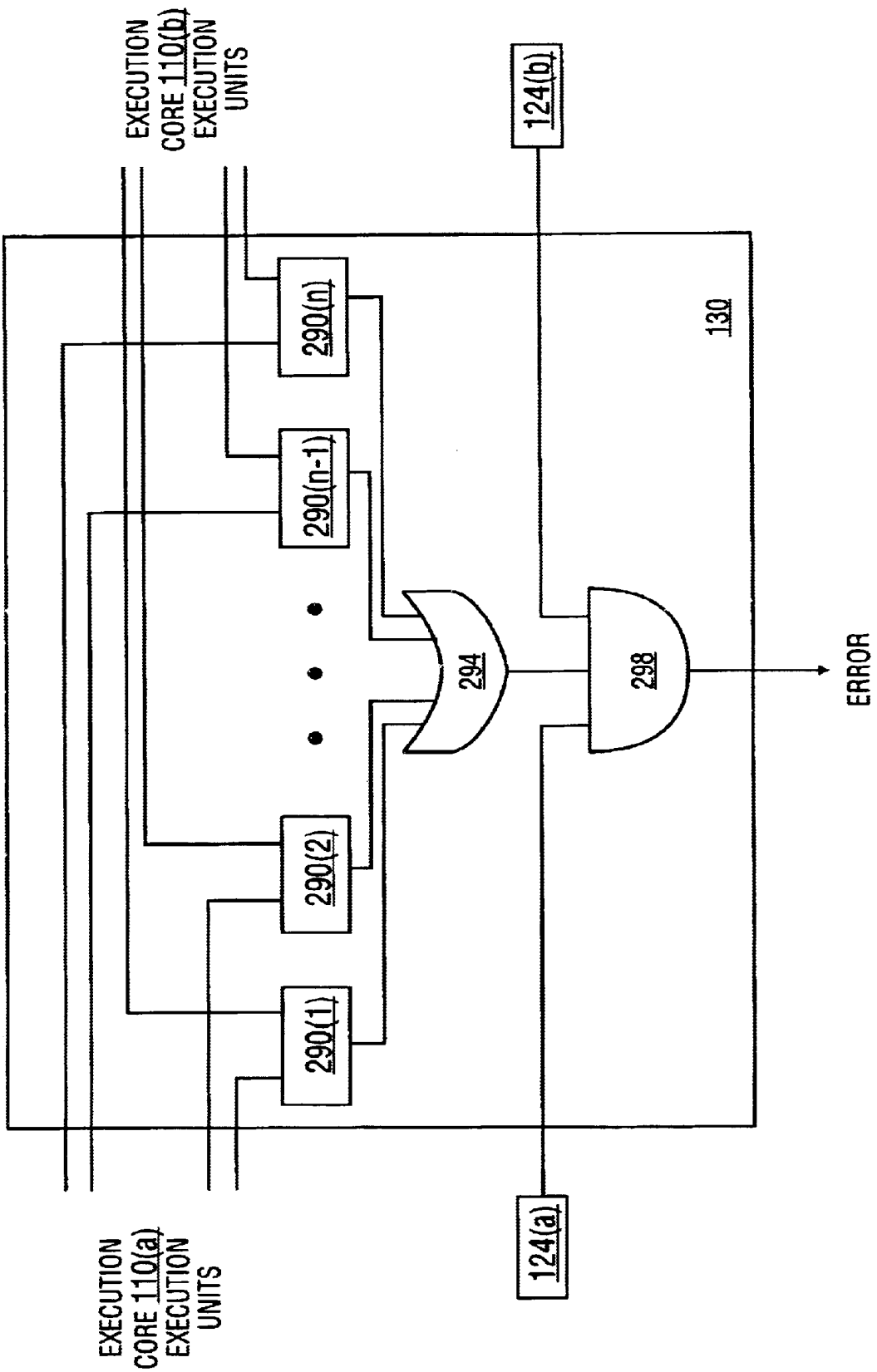

EXECUTION CORE 110(b) EXECUTION UNITS
124(b)
290(n)
290(n-1)
130
294
298
ERROR
290(2)
290(1)
EXECUTION CORE 110(a) EXECUTION UNITS
124(a)

FIG. 2C

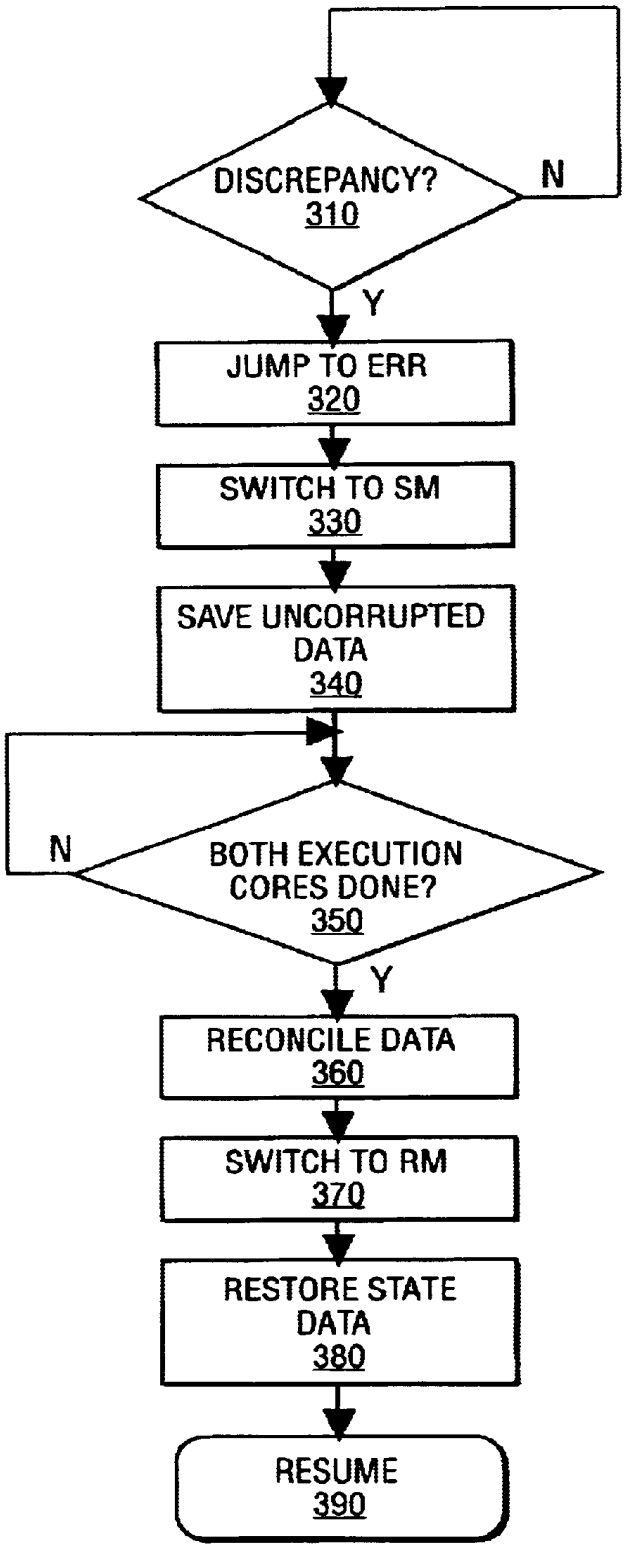

DISCREPANCY?
310
N
Y
JUMP TO ERR
320
SWITCH TO SM
330
SAVE UNCORRUPTED DATA
340
N
BOTH EXECUTION CORES DONE?
350
Y
RECONCILE DATA
360
SWITCH TO RM
370
RESTORE STATE DATA
380
RESUME
390

FIG. 3

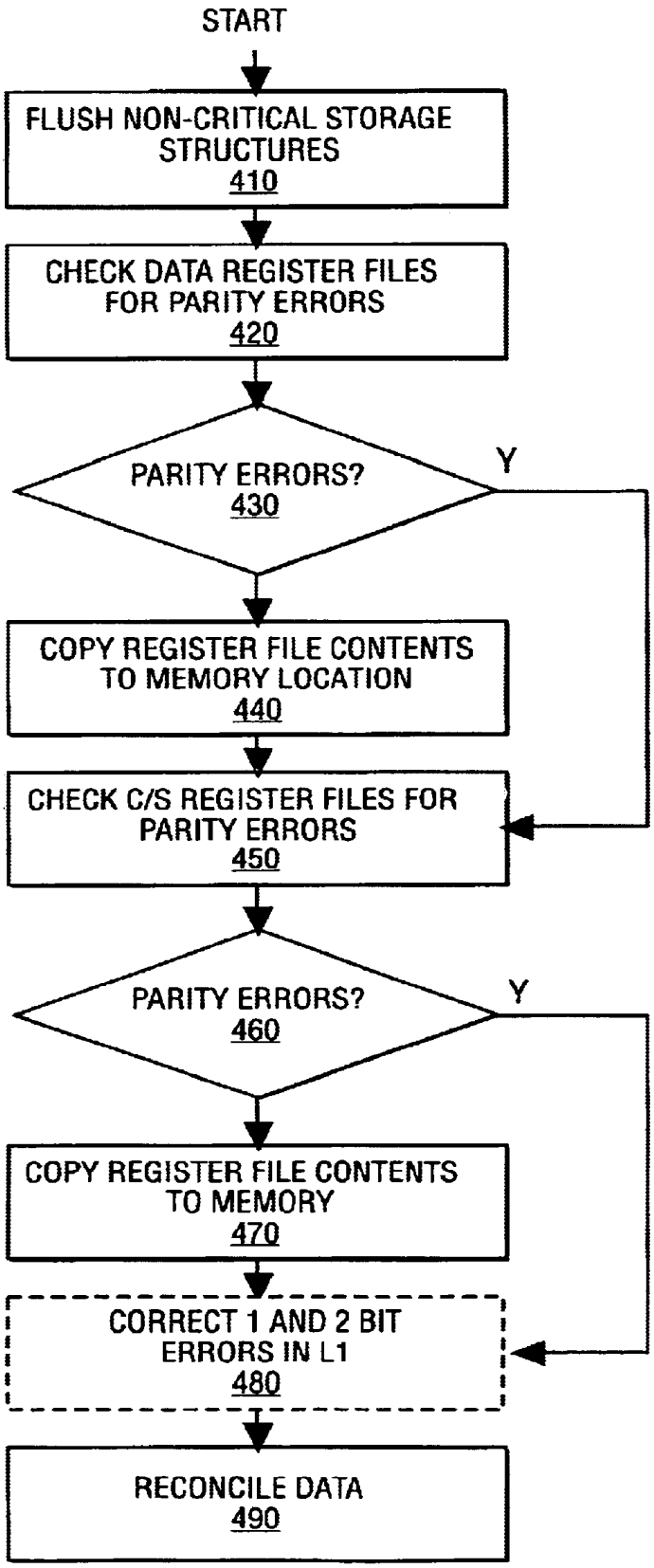

START
FLUSH NON-CRITICAL STORAGE STRUCTURES
410
CHECK DATA REGISTER FILES FOR PARITY ERRORS
420
PARITY ERRORS?
430
Y
COPY REGISTER FILE CONTENTS TO MEMORY LOCATION
440
CHECK C/S REGISTER FILES FOR PARITY ERRORS
450
PARITY ERRORS?
460
Y
COPY REGISTER FILE CONTENTS TO MEMORY
470
CORRECT 1 AND 2 BIT ERRORS IN L1
480
RECONCILE DATA
490

FIG. 4

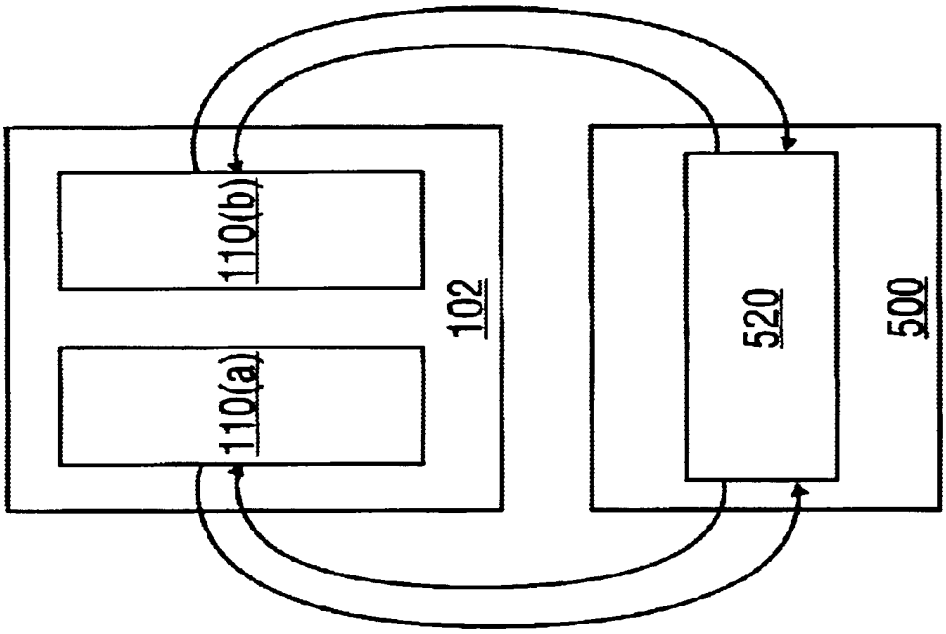


FIG. 5B

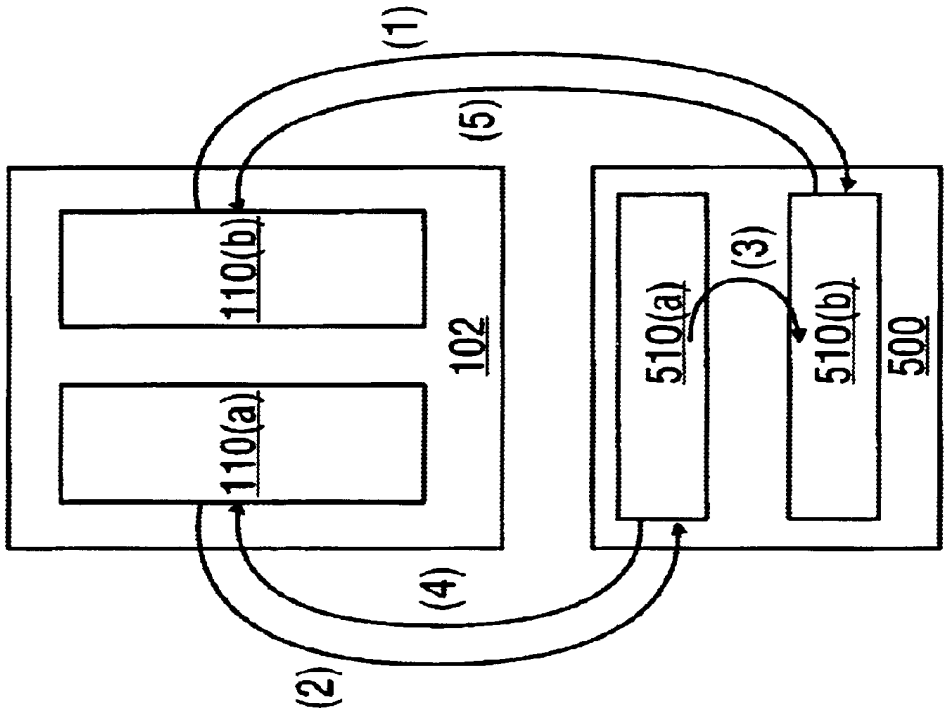


FIG. 5A

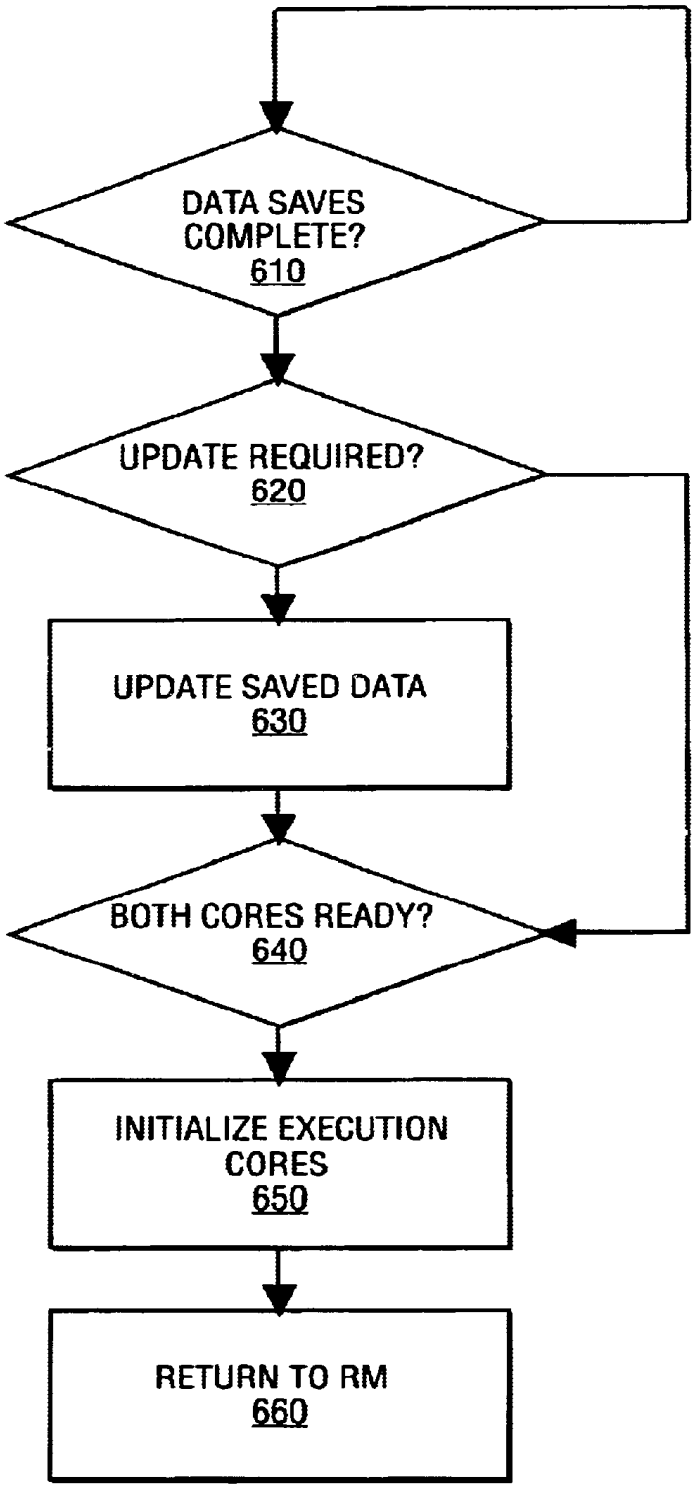

DATA SAVES COMPLETE? 610
UPDATE REQUIRED? 620
UPDATE SAVED DATA 630
BOTH CORES READY? 640
INITIALIZE EXECUTION CORES 650
RETURN TO RM 660

FIG. 6

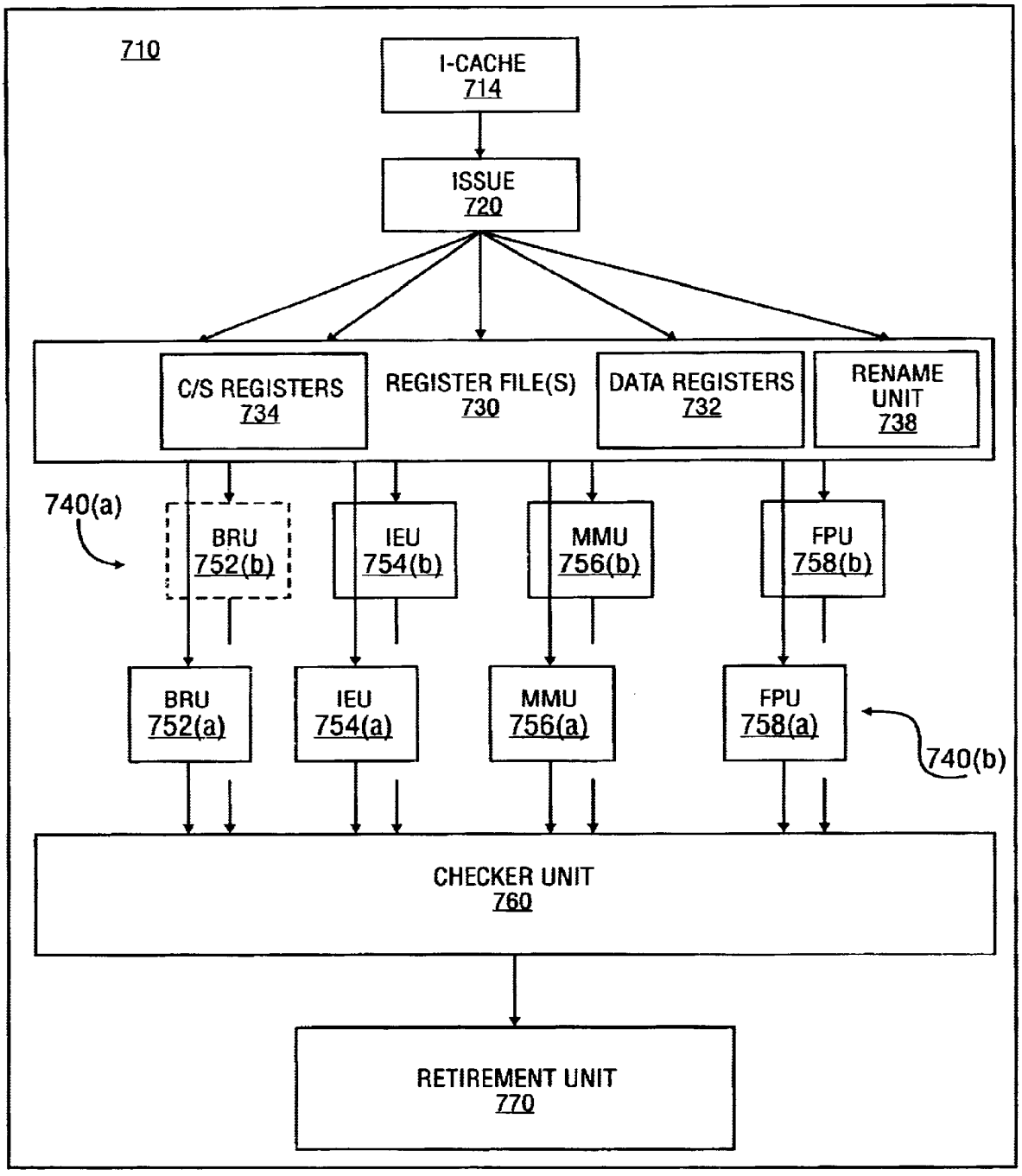

710
I-CACHE
714
ISSUE
720
C/S REGISTERS
734
REGISTER FILE(S)
730
DATA REGISTERS
732
RENAME
UNIT
738
740(a)
BRU
752(b)
IEU
754(b)
MMU
756(b)
FPU
758(b)
BRU
752(a)
IEU
754(a)
MMU
756(a)
FPU
758(a)
740(b)
CHECKER UNIT
760
RETIREMENT UNIT
770

FIG. 7

FIRMWARE MECHANISM FOR CORRECTING SOFT ERRORS

BACKGROUND OF THE INVENTION

1. Technical Field

The present invention relates to microprocessors and, in particular, to microprocessors capable of operating in high-reliability modes.

2. Background Art

Soft errors arise when alpha particles or cosmic rays strike an integrated circuit and alter the charges stored on the voltage nodes of the circuit. If the charge alteration is sufficiently large, a voltage representing one logic state may be changed to a voltage representing a different logic state. For example, a voltage representing a logic true state may be altered to a voltage representing a logic false state, and any data that incorporates the logic state will be corrupted.

Soft error rates (SERs) for integrated circuits, such as microprocessors ("processors"), increase as semiconductor process technologies scale to smaller dimensions and lower operating voltages. Smaller process dimensions allow greater device densities to be achieved on the processor die. This increases the likelihood that an alpha particle or cosmic ray will strike one of the processor's voltage nodes. Lower operating voltages mean that smaller charge disruptions are sufficient to alter the logic state represented by the node voltages. Both trends point to higher SERs in the future. Soft errors may be corrected in a processor if they are detected before any corrupted results are used to update the processor's architectural state.

Processors frequently employ parity-based mechanisms to detect data corruption due to soft errors. A parity bit is associated with each block of data when it is stored. The bit is set to one or zero according to whether there is an odd or even number of ones in the data block. When the data block is read out of its storage location, the number of ones in the block is compared with the parity bit. A discrepancy between the values indicates that the data block has been corrupted. Agreement between the values indicates that either no corruption has occurred or two (or four . . .) bits have been altered. Since the latter events have very low probabilities of occurrence, parity provides a reliable indication of whether data has been corrupted. Error correcting codes (ECCs) are parity-based mechanisms that track additional information for each data block. The additional information allows the corrupted bit(s) to be identified and corrected.

Parity/ECC mechanisms have been applied extensively to caches, memories, and similar data storage arrays. These structures have relatively high densities of data storing nodes and are susceptible to soft errors even at current device dimensions. Their localized array structures make it relatively easy to implement parity/ECC mechanisms. The remaining circuitry on a processor includes data paths, control logic, execution logic and registers ("execution core"). The varied structures of these circuits and their distribution over the processor die make it more difficult to apply parity/ECC mechanisms.

One approach to detecting soft errors in an execution core is to process instructions on duplicate execution cores and compare results determined by each on an instruction by instruction basis ("redundant execution"). For example, one computer system includes two separate processors that may be booted to run in either a symmetric multi-processing ("SMP") mode or a [Functionalredundant redundant] Functional Redundant Check ("FRC") mode. In SMP mode, instruction execution is distributed between the processors to provide higher overall performance than single processor systems. In FRC mode, one processor executes code normally and a second processor executes identical instructions on the same data that is provided to the first processor. If the second processor detects a discrepancy between its operations and those of the first processor, it signals an error. The operating mode can only be switched between SMP and FRC modes by resetting the computer system.

The dual processor approach is costly (in terms of silicon). In addition, the inter-processor signaling through which results are compared is too slow to detect corrupted data before it updates the processors' architectural states. Consequently, this approach is not suitable for correcting the detected soft errors.

Another computer system provides execution redundancy using dual execution cores on a single processor chip. The two execution cores operate in FRC mode, and ECC protected check point registers store information on intermediate states of the processor. When an error is detected in a code segment, the processor implements a micro-code routine to restore the processor to the last uncorrupted processor state, using the check point registers. Control is then returned to the code segment, beginning with the instruction (s) that encountered the error. The error recovery micro-code routine is stored on the processor chip, making it difficult to update or modify. In addition, routines sufficiently flexible to correct a broad range of errors tend to be relatively complex. Micro-code implementations of these routines occupy significant area on the processor die.

The present invention addresses these and other deficiencies of available high reliability computer systems.

SUMMARY OF THE INVENTION

The present invention provides a firmware mechanism for recovering from soft errors in a dual execution core processor that is capable of operating the execution cores in redundant mode and split mode.

A method in accordance with the present invention detects a soft error when the processor is operating the execution cores in redundant mode. An error recovery routine is executed on each execution core to save uncorrupted data from storage structures associated with the execution core. Processor state data is recovered from the saved data, and the first and second execution cores are initialized using processor state data.

A computer system in accordance with the present invention includes a dual execution core processor and a non-volatile memory that stores the error recovery routine. The error recovery routine is invoked when the processor detects a soft error while operating in the redundant mode. The routine switches the processor to the split mode, in which mode each execution core saves uncorrupted data in its associated storage structures to a designated memory location. The routine recovers processor state data from the saved data.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention may be understood with reference to the following drawings, in which like elements are indicated by like numbers. These drawings are provided to illustrate selected embodiments of the present invention and are not intended to limit the scope of the invention.

FIGS. **1** is a block diagram of one embodiment of a computer system suitable for implementing the present invention.

FIG. **2A** is a block diagram of one embodiment of the dual execution core processor in FIG. **1**.

FIG. **2B** is a block diagram of one embodiment of the FET stage of the processor in FIG. **2A**.

FIG. **2C** is a block diagram of one embodiment of the check unit of the processor in FIG. **2A**.

FIG. **3** is a flow chart representing one embodiment of a method in accordance with the present invention for recovering from soft errors.

FIG. **4** is a flowchart representing one embodiment of an error recovery routine implemented by an execution core.

FIGS. **5A** and **5B** are block diagrams representing embodiments of different mechanisms for reconciling saved data for soft error recoveries.

FIG. **6** is a flowchart representing one embodiment of the data reconciliation mechanism of FIG. **5A**.

FIG. **7** is a block diagram of one embodiment of an execution core that provides cluster level redundancy when a dual core processor is operated in split mode.

DETAILED DESCRIPTION OF THE INVENTION

The following discussion sets forth numerous specific details to provide a thorough understanding of the invention. However, those of ordinary skill in the art, having the benefit of this disclosure, will appreciate that the invention may be practiced without these specific details. In addition, various well-known methods, procedures, components, and circuits have not been described in detail in order to focus attention on the features of the present invention.

The present invention provides a flexible approach to correcting soft errors in a dual core processor which may be switched dynamically between a redundant mode and a split mode. In redundant mode, the processor operates the execution cores in lock step on identical code segments and compares the results to identify errors. In split mode, the execution cores may process instructions independently, e.g. the execution cores may operate on different instructions in a given clock cycle.

For one embodiment of the invention, the redundant mode may be a high reliability (HR) processor execution mode that reduces the risk of soft errors when executing critical code segments. The split mode may be a high performance (HP) processor execution mode that is available to process selected code faster, by increasing the execution resources available to the selected code. The split mode may also be available for only selected purposes, such as error recovery or [bootstrapping] bootstrapping the processor. The significant aspect of split mode is that it allows the execution cores to operate "independently", i.e. each may execute different instructions on a given clock cycle.

In accordance with the present invention, an error recovery routine is stored in a non-volatile memory. The recovery routine is accessed when the processor, implementing a program thread in redundant mode, detects a soft error. The error may be indicated by a discrepancy between results in the two execution cores. The routine switches the processor to split mode, in which mode each execution core examines its associated storage locations for corrupted data. Uncorrupted data is copied to a designated memory location and sufficient processor state data is recovered from the uncorrupted data to resume the interrupted program thread. The recovery routine initializes the execution cores with the recovered processor state data.

For one embodiment of the invention, various resources associated with each execution core are parity-protected, and the recovery routine implements a parity check to identify any corrupted data in an execution core. Parity-protected resources may include, for example, general registers, floating point registers, control and status registers, low level caches, and the like. In general, any structure used to store processor state data can be parity-protected. The extent of protection depends on the system's reliability requirements.

For another embodiment of the invention, the error recovery routine identifies and saves uncorrupted data from one or both execution cores to a designated memory location. The execution cores are initialized with processor state data recovered from the saved data. For example, both execution cores may be initialized with the processor state data saved from the execution core that did not experience the soft error. Alternatively, uncorrupted data from each execution core may be saved to different memory locations, and data from one memory location may be used to replace the corrupted data in the other memory location. This alternate approach initializes each execution core with a copy of the processor state data from its designated memory location. By maintaining separate sets of processor state data, this alternative reduces the risk that a soft error generated during the initialization process will escape detection. The recovery routine synchronizes the execution cores for a return to redundant mode following the identification and reconciliation of saved processor state data from the execution cores in split mode.

For another embodiment of the invention, the processor may be configured to retain some level of redundancy within the individual execution cores in split mode. For example, the execution resources of each execution core may be logically organized into redundant execution clusters for split mode operations. The execution clusters operate in lock step on the same instruction when the processor is in split mode, and results from the redundant clusters may be compared to detect soft errors within a single execution core.

FIG. **1** is a block level diagram of one embodiment of a computer system **100** in accordance with the present invention. Computer system **100** comprises a processor **102**, a non-volatile memory **104**, a dynamic memory **106**, and system logic **108**. System logic **108** routes communications among processor **102**, non-volatile memory **104**, and dynamic memory **106**. An error recovery routine **190** is stored in non-volatile memory **104**, although portions of routine **190** may be copied (shadowed) to dynamic memory **106**.

The disclosed embodiment of processor **102** includes a first execution core **110**(*a*), a second execution core **110**(*b*), corresponding core status registers (CSRs) **120**(*a*), **120**(*b*), and a check unit **130**. Each CSR **120**(*a*), **120**(*b*) includes a core status bit (CSB) **124**(*a*), **124**(*b*), respectively. A bus interface **140** to transfer data/instructions to and from processor **102** is also shown. Each execution core **110**(*a*), **110**(*b*) includes resources to fetch, decode, execute, and retire instructions. In the following discussion, references to execution cores **110**(*a*), **110**(*b*) are not indexed unless the comments apply to a specific execution core **110**. References to CSRs **120**, CSBs **124**, and any other resources that may be duplicated in execution cores **110** are treated similarly.

In redundant mode, execution cores **110** execute the same instructions from a code segment in lock step, and the results are compared by check unit **130** to detect errors in either execution core **110**. In split mode, execution cores **110** operate "independently". That is, each may execute different instructions from one or more code segments. As noted

above, split mode may provide a programmable high performance mode for selected programs since the execution resources available in the processor are effectively doubled in split mode.

The level of independence between execution cores **110** in split mode may vary in different embodiments of processor **102**. For one embodiment, processor **102** may operate as an SMP system implemented on a single processor chip, when it is in split mode. In this case, each execution core **110** operates as a separate processor, sharing only the processor die and certain components of a memory system. This embodiment of processor **102** is particularly useful for code that requires a high performance processor. Another embodiment of processor **102** may support some level of coupling between execution cores **110** in split mode by providing appropriate channels to share processor state information or execution core resources.

Switching between redundant and split modes may be accomplished under hardware, software or firmware control. For software and firmware controlled switching, mode switch instructions may be provided through the different program threads being executed, an operating system (OS) scheduler, interrupt handlers, firmware routine or similar sources. For hardware controlled transitions, switching may be triggered in response to detected conditions, e.g. execution of a particular instruction type, detection of a discrepancy between results in different execution cores or resolution of the discrepancy.

For one embodiment of the invention, CSBs **124**(*a*) and **124**(*b*) indicate the states of execution cores **110**(*a*) and **110**(*b*), respectively, and CSBs **124**(*a*) and **124**(*b*) together indicate the mode in which processor **102** is operating. The execution mode of the processor as a whole may be tracked through a single processor status bit (PSB) **128**, indicated by the dashed box in FIG. **1**. For one embodiment, CSB **124** is set to a first value, e.g. one, when processor **102** is to be operated in redundant mode, and it is set to a second value, e.g. zero, when processor **102** is to be operated in split mode. CSB **124** may be adjusted when a mode switch instruction triggers a switch between redundant and split modes. Various resources in processor **102** alter their operations according to its execution mode. For the disclosed embodiment of processor **102**, instruction fetching, instruction retirement, and result checking are handled differently in redundant and split modes, and the corresponding resources adjust their operations according to the states of CSBs **124** (or PSB **128**).

An embodiment of processor **102** having dual execution modes is disclosed in U.S. patent application Ser. No. 09/470,098, entitled "Microprocessor With High Reliability Operating Mode" and filed on even date herewith.

FIG. **2**A represents in greater detail one embodiment of processor **102**. For the disclosed embodiment, each execution core **110** is represented as a series of stages in an instruction execution pipeline. Each stage corresponds to one or more operations implemented by execution cores **110** to execute their instructions. Alternatively, the pipeline stages may be understood to represent the logic that executes the indicated operations. Instructions and data are provided to execution cores **110** from a memory system **270**. Memory system **270** represents dynamic memory **106** and any intervening caches. For example, cache **280** represents one portion of memory system **270** to which results from executed instructions are written. Cache **280** may be located on the same chip as processor **102** or it may be located on a separate chip.

For the disclosed embodiment of processor **102**, each execution core **110** is partitioned into a fetch (FET) stage **210**, a decode (DEC) stage **220**, a register (REG) stage **230**, an execute (EXE) stage **240**, a detect (DET) stage **250**, and a retirement (RET) stage **260**. One or more instructions are retrieved from memory system **270** in FET stage **210**. The retrieved instructions are decoded into pops in DEC stage **220**, and source operands specified by the μop(s) are retrieved in REG stage **230**. The μop(s) are executed on the retrieved operands in EXE stage **240**, and any exceptions raised by the μop(s) are signaled in DET stage **250**. The μop(s) is retired in RET stage **260** if no exceptions are detected. For the disclosed embodiment, results from retired μop(s) are written to cache **280** through retirement channel **264**.

In this discussion, instruction, instruction bundle, and macro-instruction are used interchangeably as are pop and instruction syllable. The latter refer to the instruction recognized by the execution units of the processor. The former refer to instructions in the form in which they are provided to the processor. For some embodiments, there may be little or no difference between these entities.

Embodiments of processor **102** may include a buffer to decouple front-end stage(s) (FET or FET and DEC) from back-end stages (DEC, REG, EXE, DET, and RET or REG, EXE, DET, and RET). The buffer temporarily stores fetched (or fetched and decoded) instructions. This allows front-end operations to continue, even if back-end operations are stalled or otherwise delayed. It also allows back-end operations to proceed if front-end operations are delayed. One embodiment of processor **102** employs a decoupling buffer to correct errors detected in redundant mode.

The present invention does not require partition of processor **102** into a particular set of pipeline stages. For example, a disclosed stage may be subdivided into two or more stages to address timing issues or facilitate higher processor clock speeds. Alternatively, two or more stages may be combined into a single stage. Other embodiments may or may not employ a decoupling buffer. Still other embodiments may include hardware for processing instructions out-of-order. The disclosed pipeline provides only one example of how operations may be partitioned in a processor for use with the present invention.

Also shown for each execution core **110** are status/control (S/C) registers **234**, data registers **238**, and a data cache **244**. S/C registers **234** store information that governs the operation of execution core **110**. For example, S/C registers **234** typically include CSR **120** (and CSB **124**). Data registers **238** store operands for use by various resources in execution core **110**, and data cache **244** buffers operands between memory system **270** and other resources in execution core **110**. Depending on timing constraints, data cache **244** may provide operands to data registers **238**, to execution resources in EXE stage **240**, or to both. For one embodiment of the invention, each execution core **110** provides some form of parity protection for S/C registers **234**, data registers **238**, and cache **244**.

Execution cores **110**(*a*) and **110**(*b*) are synchronized to operate on identical instructions in lock step when processor **102** is in redundant mode. In split mode, execution cores **110**(*a*) and **110**(*b*) may operate independently on different instructions. As noted above, various embodiments of processor **102** may support different levels of coordination between execution cores **110**(*a*) and **110**(*b*) in split mode, as indicated by the dashed arrow in FIG. **2**A. For example, if processor **102** operates as a single chip SMP system in split mode, the need for coordination between execution cores **110**(*a*) and **110**(*b*) arises mainly during mode switches. For

other embodiments of processor **102**, execution cores **110**(*a*) and **110**(*b*) may handle processes that are closely coupled. These embodiments support some sharing of data between S/C registers **234**(*a*) and **234**(*b*), data registers **238**(*a*) and **238**(*b*), and data caches **244**(*a*) and **244**(*b*), as well as some coordination of operations between the different pipe stages.

FIG. **2**B shows one embodiment of FET stages **210**(*a*), **210**(*b*) suitable for providing instructions to execution cores **110**(*a*), **110**(*b*), respectively, in redundant mode and split mode. Each FET stage **210** includes an instruction pointer (IP)-select MUX **212** and an instruction cache **214** that is coupled to DEC stage **220**. S/C registers **234** include an IP register **236** that may be initialized by software to indicate the next instruction to be executed. In addition, FET stage **210**(*b*) includes a MUX **216** at the output of cache **214**(*b*). MUX **216** is controlled by CSBs **124** through AND gate **218**. For one embodiment of the invention, instruction cache **214** may be protected by a parity-based scheme such as ECC.

MUX **212** receives IPs at its data inputs from various sources, including IP register **236**. In response to a signal at its control input, MUX **212** selects one IP to indicate the next instruction to be fetched from cache **214**. In split mode, CSBs **124** are set to zero and MUX **216** transfers the instruction provided by instruction cache **214**(*b*). In this mode, IP registers **236**(*a*) and **236**(*b*) are initialized and updated independently, and caches **214**(*a*) and **214**(*b*) feed corresponding instructions to DEC stages **220**(*a*) and **220**(*b*), respectively. In redundant mode, CSBs **124** are set to one, and MUX **216** provides the instruction from cache **214**(*a*) to DEC stage **220**(*b*).

An alternative embodiment of FET stages **210** does not employ MUX **216**. Instead, IP registers **236**(*a*), **236**(*b*) and caches **214**(*a*), **214**(*b*) are initialized to the same state for redundant mode, and FET stages **210**, including caches **214**, operate in lock step. Persons skilled in the art of processor design and having the benefit of this disclosure will recognize other variations on FET stages **210** that may be used to implement independent and lock-step instruction fetching for execution cores **110**, according to the execution mode in which processor **102** operates.

FIG. **2**C is a block diagram representing one embodiment of check unit **130** in accordance with the present invention. The disclosed embodiment of check unit **130** includes "n" comparators **290**(1)–**290**(n), an OR gate **294**, and an AND gate **298**. A comparator **290** is provided for each execution unit in an execution core **110**. For example, one embodiment of processor **102** may include an integer execution unit (IEU), a floating-point execution unit (FPU), a memory execution unit (MEU), and a branch execution unit BRU in the EXE stage of each execution core **110**. For this embodiment, check unit **130** includes 4 comparators **290**. Comparator **290**(1), **290**(2), **290**(3) and **290**(4) monitor outputs of the IEUs, FPUs, MEUs, and BRUs, respectively, from execution cores **110**(*a*) and **110**(*b*).

For the disclosed embodiment of check unit **130**, each comparator **290** generates a logic value zero when the execution results applied to its inputs match and a logic value one when the execution results do not match. For one embodiment of check unit **130**, comparators **290** are self-check comparators. OR gate **294** generates a logic value one when any of comparators **290** indicates that its corresponding execution results do not match. The output of OR gate **294** serves as an ERROR signal when AND gate **298** is enabled. For the disclosed embodiment, this occurs when CSBs **124** are both set to one, i.e. when processor **102** is in redundant mode.

Persons skilled in the art of processor design and having the benefit of this disclosure will recognize other variations on check unit **130** that may be activated to monitor results in execution cores **110** when processor **102** is in redundant mode.

The present invention provides a firmware-based recovery mechanism that may be implemented alone or in combination with hardware and software-based recovery mechanisms, to restore the processor to an error-free state when a discrepancy is detected in redundant mode. The soft errors targeted by the present invention are unlikely to occur in both execution cores **110** simultaneously. A difference between execution results detected by check unit **130** is likely due to a soft error in the circuitry of one of execution cores **110**. Storage structures such as register files, caches, latches and the like are particularly susceptible to these errors. These structures store voltage levels that represent operand values, instructions, or pops at various points of the instruction pipeline. The present invention provides a mechanism to restore integrity to the execution cores of a processor operating in redundant mode, when a discrepancy is detected between results generated by the execution cores.

According to one embodiment of the present invention, a firmware-based recovery routine is accessed when a discrepancy is detected between the results generated by the two execution cores of the processor in redundant mode. The error recovery routine switches the processor to split mode, in which mode each execution core examines data from its execution resources for errors and saves the uncorrupted data to a specified memory location. The error recovery routine initializes the execution cores, using the uncorrupted data, and returns control to the interrupted program thread. For one embodiment of the invention, the execution resources examined are parity-protected storage structures. Sufficient processor state data is recovered from the uncorrupted data to initialize both execution cores for continued processing of the interrupted program thread.

The use of a firmware-based recovery routine allows the processor to implement more sophisticated recovery mechanisms. These mechanisms can address a wider variety of errors than can be addressed through micro-code or hardware-based recovery mechanisms, both of which are limited by their impact on die area. Firmware-based recovery mechanisms can be updated more easily to handle different errors or implement improvements in error recovery algorithms. In addition, firmware-based recovery mechanisms can notify higher level systems, such as system level firmware or the operating system, to take appropriate action if necessary.

For one embodiment of the invention, the error recovery routine may be accessed through a machine check (MC). The MC may be triggered by check unit **130**, when a discrepancy is detected. Alternatively, the processor may first attempt to address the discrepancy through a hardware recovery mechanism and trigger the error recovery routine only if the hardware mechanism fails. Hardware recovery mechanisms include, for example, resteering the execution core(s) or portions of the execution core(s) to re-execute the instruction stream beginning with the instruction for which the discrepancy was detected.

FIG. **3** is a flow chart representing a method **300** in accordance with the present invention for correcting soft errors. Method **300** is initiated when an error is detected **310** while the processor is operating in redundant mode. The error may be indicated by a discrepancy between results generated by the first and second execution cores. When the

error is detected **310**, the processor jumps **320** to an error recovery routine (ERR). This may be done, for example, by triggering an MC, and directing the processor to the ERR through an associated vector table.

The ERR switches **330** the processor from redundant mode (RM) to split mode (SM). For one embodiment of method **300**, each execution core operates in split mode to save **340** uncorrupted data from its parity-protected storage structures to a specified memory location. Uncorrupted data may be identified by scanning the data in parity-protected resources for parity errors or by copying the data from these resources to a memory location and scanning the copied data for parity errors. When both execution cores complete **350** their data saves, the saved data is reconciled **360** to provide an uncorrupted set of processor state data. For another embodiment of method **300**, only the execution core that identifies no corrupted data in its storage structures saves **340** its data to the specified memory location. The processor is returned **370** to redundant mode, the execution cores are initialized **380** using the recovered processor state data, and execution of the interrupted process thread resumes **380**.

For one embodiment of the invention, processor **102** is switched between redundant and split modes under software control, e.g. through mode switch instruction. For another embodiment of the invention, the processor **102** may be switched through a hardware mechanism.

Prior to detecting a discrepancy, the processor is in redundant mode and the execution cores operate in lock step. When a discrepancy is detected, the error recovery routine may implement the mode switch by providing a switch-to-split-mode instruction (SW_SM) to each execution core. In redundant mode, these instructions are staged down the pipeline stages of the two execution cores in lock step, switching the processor to split mode when they retire. There is thus no need to synchronize the execution cores on the switch to split mode. Once the processor is in split mode, each execution core may independently execute the data recovery operations of ERR.

When the execution core(s) complete its data recovery operations, the processor transitions from split mode back to redundant mode to continue the interrupted program thread. For one embodiment of the invention, this transition may be implemented in stages by executing a switch-to-redundant-mode instruction (SW_RM) in each execution core. Staging the transition accommodates differences in the times at which the execution cores complete their data saving operations.

For the disclosed embodiment, SW_RM switches one execution core to a "ready state", e.g. CSB(b) **124**=1, in a first stage. In the second stage, an SW_RM instruction switches the other execution core to a "ready state", e.g. CSB(a)=1. The processor returns to redundant mode when both execution cores reach the ready state, e.g. PSB=CSB(a) AND CSB(b). The first and second stages may overlap in whole (simultaneous switch), in part, or not at all (sequential switch), depending on the algorithm used to schedule the mode switch instructions.

The ready state allows the execution cores to be synchronized before the processor switches from split mode to redundant mode. For other embodiments of the invention, synchronization may be implemented through a hardware mechanism. For example, the first execution core may transition to a ready state when an end-of-routine condition is detected, and the second execution core may transition to the ready state when the first execution core completes its transition. As with software controlled mode switches, the ready state allows the execution cores to rendezvous before the processor is switched back to redundant mode.

FIG. **4** illustrates one embodiment of a method **400** implemented by an execution core to save uncorrupted data. For example, method **400** may represent a portion of ERR implemented by an execution core following the switch to split mode. As noted above, ERR may be reached through a machine check, which transfers control of the processor to ERR through an associated vector table. ERR switches the processor from redundant to split mode, which allows each execution core to independently access and implement the data checking and recovery routine represented by FIG. **4**.

According to the disclosed method, non-critical storage structures are flushed **410**. Non-critical storage structures include, for example, caches, registers, and storage structures that do not store architectural state data. These storage structures typically are not parity-protected. Depending on the processor embodiment, these structures may include L0 instruction and data caches, branch target buffers, advanced load tracking tables and the like.

The contents of the data register files are checked **420** for parity errors. If no parity error is found **430**, the contents of the data register files are copied **440** to a specified memory location. As noted above, this may be a memory location shared with the other execution core or it may be a memory location reserved for the particular execution core. If a parity error is found **430**, no data is copied from the data register files. Alternatively, only uncorrupted data may be copied from the data register file(s).

The C/S register files are also checked **450** for parity errors. These include, for example, the core status register described above, various registers that store information about the current thread, and the translation registers in any translation look-aside buffers (TLB). If no parity error is found **460** in the C/S register files, the contents of the C/S register files are copied [**570**] **470** to a specified memory location. If a parity error is found, no data is copied from the C/S register file(s). Alternatively, only uncorrupted data may be copied from the C/S register file(s).

Embodiments of the processor may implement error correction codes (ECC) for selected caches in firmware. This eliminates the need for ECC correction hardware in the processor, although the caches still store the ECC bits. For these embodiments, 1 and 2 bit errors may be corrected **480** through ERR. Block **480** is bypassed if ECC is implemented in hardware. Once all detected errors are corrected, the execution core jumps **490** to a wait state.

Once each execution core has executed method **400**, the specified memory location contains only uncorrupted data. One consequence of the statistical nature of soft errors is that it is highly unlikely for the same data block to be corrupted in both execution cores at the same time. Consequently, the data saving process provides sufficient uncorrupted processor state data to initialize the execution cores and resume the interrupted program thread.

The present invention does not require the storage structures to be flushed or checked in any particular order. For example, data register files may be checked after C/S register files and non-critical files may be flushed at any convenient time in the process. The order indicated above is provided solely for purposes of illustration.

In addition to saving uncorrupted data to a designated memory location, ERR provides a mechanism to reconcile the saved data and provide each execution core with sufficient uncorrupted processor state data to resume the interrupted program thread.

FIG. **5**A is a block diagram representing one mechanism for reconciling the data saved by each execution core in a dual execution core processor. Execution core **110**(*b*) implements ERR to copy (1) its uncorrupted data to a designated location **510**(*b*) in memory system **500**. Execution core **110**(*a*) implements ERR to copy (2) its uncorrupted data to a designated memory location **510**(*a*). For the exemplary mechanism, a soft error in the storage structures of execution core **110**(*b*) leaves location **510**(*b*) with an incomplete set of processor state data. Execution core **110**(*a*), which encounters no parity errors, copies (2) a complete set of processor state data to location **510**(*a*). Here, "complete" refers to a subset of the processor state data that is sufficient to allow processor **102** to resume execution of the interrupted program thread. The corrupted data in the storage structures of execution core **110**(*b*) may be tracked as execution core **110**(*b*) copies (1) its uncorrupted data to memory location **510**(*b*), to maintain a record of the data that needs to be updated.

Following data saving operations (1), (2), execution core **110**(*b*) implements ERR to copy (3) uncorrupted data from location **510**(*a*) to location **510**(*b*). This provides a second complete set of processor state data for reinitializing the execution cores of processor **102**. For example, when execution core **110**(*b*) determines that execution core **110**(*a*) has finished its copy operations, it may copy data from location **510**(*a*) to location **510**(*b*) for only those data blocks that correspond to corrupted data in its associated storage structures. This data reconciliation mechanism provides two sets of processor state data in memory **500**, which may then be used to initialize (4) and (5) execution cores **110**(*a*) and **110**(*b*), respectively. Initializations (4) and (5) may be done separately by each execution core **110** in split mode or they may be done concurrently by the execution cores in redundant mode.

Except for the portion of ERR that copies (3) uncorrupted data from memory location **510**(*a*) to memory location **510**(*b*), the disclosed data reconciliation mechanism proceeds through redundant operations. Data is checked, saved to memory, and written back to storage structures by each execution core. This reduces the likelihood that a soft error generated in one set of processor state data during recovery will corrupt both execution cores.

FIG. **6** is a flowchart representing the data reconciliation mechanism of FIG. **5**A. This corresponds, for example, to block **490** of FIG. **4**. A processor implementing the data reconciliation portion of ERR determines **610** whether the other execution core has completed its data save operations. When these operations complete, the execution core determines **620** whether it needs to update its saved data.

For one embodiment of method **600**, the execution core that detected corrupted data in its storage structures updates **630** its saved data using corresponding data from the memory location of the other execution core. For example, the execution core may store an indication of the data blocks that appear corrupted during its save operations. This execution core may then use these indications to retrieve uncorrupted versions of the data from the appropriate memory address of the other execution core's designated memory location. The execution core that detected no corrupted data waits for the other execution core to complete **640** its update **630**. When both execution cores reach wait state **640**, the processor returns **650** to redundant mode and initializes **660** the execution cores with the reconciled data. An alternative embodiment of method **600** may initialize the execution cores in split mode and then return to redundant mode.

FIG. **5**B represents an alternative mechanism for data reconciliation. For this mechanism, both execution cores **110**(*a*) and **110**(*b*) write (1) and (2), respectively, their uncorrupted data to corresponding memory addresses in the same memory location **520**. For one embodiment of the mechanism, only the execution core having no corrupted data writes all its processor state data to memory location **520**. The execution core that detects corrupted data in its storage structures does not copy the corrupted data to memory location **520**.

Another embodiment of the mechanism in FIG. **5**B may write data from the execution core that detects no uncorrupted data in its associated memory structures and skip the save step for the other execution core, i.e. the one that includes corrupted data. In either case, the processor state data copied to location **520** is complete and can be written back (3) and (4) to execution cores **110**(*a*) and **110**(*b*), respectively, to reinitialize them for resumption of the interrupted program thread. Since only one set of processor state data is maintained in the data reconciliation mechanism of FIG. **5**B, a soft error in this data set during recovery is not detectable by a simple result comparison.

For the above described embodiments of the present invention, error recover is implemented with the processor in split mode to allow each execution core to identify independently any errors in its associated storage structures. It is also possible for soft error to arise during these error recovery operations, when redundant execution is not available to detect soft errors. For high reliability systems, even this limited vulnerability to soft errors may be too great.

For one embodiment of the invention, the execution resources of each execution core may be configured to operate in parallel as two or more logical clusters during split mode operation. While each execution core implements ERR independently, logically defined execution clusters within each execution core implement instructions from ERR redundantly. For this embodiment, instructions from ERR are replicated within each execution core and directed to different execution clusters. The results generated by the different execution clusters may be compared to determine whether a soft error arose as the ERR instructions were processed.

FIG. **7** is a block diagram of one embodiment of an execution core **710** that includes this internal redundancy feature. The disclosed embodiment of execution core **710** includes an instruction cache (I-cache) **714**, decode or dispersal unit **720**, a register file(s) **730**, first and second execution clusters **740**(*a*) and **740**(*b*), respectively, a check unit **760**, and a retirement unit **770**. In the following discussion, references to execution clusters **740**(*a*) and **740**(*b*) are not indexed unless necessary to avoid ambiguity. References to other resources in execution core **710** that may be duplicated, such as the execution units that make up execution clusters **740** are treated similarly.

For the disclosed embodiment, each execution cluster **740** includes a branch execution unit (BRU) **752**, an integer execution unit (IEU) **754**, a memory management (load/store) unit (MMU) **756**, and a floating point unit (FPU) **758** (generically, "execution units"). Different embodiments of execution core **710** may include different types and numbers of execution units without departing from the scope of the present invention. For example, the Merced processor of Intel® Corporation employs a branch execution unit that include three separate branch execution units. Execution clusters **740** are shown separated to illustrate their logical organization in split mode, not to reflect any actual separation between the different execution resources.

Other embodiments of execution core **710** may not duplicate all execution units **750** or may not duplicate instructions

to certain execution units. For example, branch execution units require significant processor resources for support, and considerable die area may be saved by not duplicating BRU **752**. This is indicated by the dashed box around BRU **752**(*b*) in FIG. **7**. In this case, branch instructions may either not be executed redundantly, or alternate mechanisms may be employed to check their execution. For example, branch instructions may be duplicated and executed serially. Similarly, duplicating load and store instructions may saturate bandwidth to the memory system in HR mode. In this case, multiple MMUs **756** may be implemented to accommodate the presence of multiple load and/or store operations in an instruction bundle, but the individual load/store operations are not duplicated these instructions in HR mode. Similar arrangements may be employed for other execution units **750** and instruction types without departing from the spirit of the invention.

I-cache **714** provides instructions to decode unit **720**, which directs them to appropriate execution units through register file(s) **730**. Register files **730** include data register files **732**, control/status (C/S) registers **734**, and a register rename unit **738**. Data register files **732** provide temporary storage for, e.g., the integer and floating point operands that are manipulated by execution units **750**. Rename unit **738** maps the virtual register identifiers specified in pops to the physical registers in register file **730**.

C/S registers **734** store information that controls the manner in which execution core **710** operates and the status of the various execution resources. For one embodiment of execution core **710**, issue unit **720** provides identical instructions to execution clusters **740**(*a*) and **740**(*b*) when the processor is in split mode. For example, issue unit **720** provides the μops from an instruction (or the μops from identical instructions) to appropriate execution units in execution clusters **740**(*a*) and **740**(*b*). Results generated by clusters **740**(*a*) and **740**(*b*) are compared by check unit **760**, and an error is indicated if the execution results are different. In redundant mode, execution clusters **740**(*a*) and **740**(*b*) may be controlled independently to process different instructions, and redundancy is provided by executing identical instructions on the other execution core. Execution core **710** provides execution cluster-level redundancy to protect processor from soft errors when it is operated in split mode. An embodiment of a processor that includes an execution core **710** is discussed in greater detail in U.S. patent application Ser. No. 09/470,098, entitled "Microprocessor With High Reliability Operating Mode and filed on even date herewith.

There has thus been disclosed a firmware-based mechanism for correcting soft errors in a dual core processor that can be switched between redundant execution mode and split execution mode. When an error is detected in a program thread executing in redundant mode, the processor switches to split mode. In split mode, an execution core identifies uncorrupted processor state data in its storage structure and copies this uncorrupted data to a specified memory location. The uncorrupted data is reconciled to provide sufficient processor state data to resume execution of the interrupted program thread. The processor is returned to redundant execution mode, the execution cores are initialized with the processor state data, and the interrupted program thread is resumed. Switches between redundant mode and split mode may be implemented through mode switch instructions.

The disclosed embodiments have been provided to illustrate various features of the present invention. Persons skilled in the art of processor design, having the benefit of this disclosure, will recognize variations and modifications of the disclosed embodiments, which none the less fall within the spirit and scope of the appended claims.

I claim:

1. A computer system, comprising:

a processor having a first execution core, a second execution core, and a check unit, the first and second execution cores to process instructions independently when the processor is in a split mode and to process identical instructions in lock step when the processor is in a redundant mode, and the check unit to compare results from the first and second execution cores when the processor is in redundant mode;

a non-volatile memory in which is stored a recovery routine to switch the processor to split mode when the check unit detects an error, to identify uncorrupted processor state data from at least one of the execution cores, and to initialize the first and second execution cores with the identified processor state data; and

a main memory, wherein the recovery routine saves the uncorrupted processor state data to an uncacheable portion of the main memory system.

2. The computer system of claim **1**, wherein the error recovery routine returns the processor to the redundant mode.

3. The computer system of claim **1**, wherein the error recovery routine identifies uncorrupted processor state data in selected storage structures associated with each execution core, copies the uncorrupted processor state data to a specified memory location, and reconciles the copied processor state data, when the processor enters split mode.

4. The computer system of claim **3**, wherein the error recovery routine synchronizes the first and second execution cores in a rendezvous state and restores the processor to redundant mode.

5. A method for handling soft errors in a processor capable of operating first and second execution cores in redundant and split modes, the method comprising:

detecting a soft error when the processor is operating in the redundant mode;

executing an error recovery routine on each execution core to save uncorrupted data from storage structures associated with the first and second execution cores, wherein the error recovery routine is stored in a non-volatile memory, the error recovery routine includes switching the processor from redundant mode to split mode and saving uncorrupted data from each execution core to an uncacheable portion of a main memory system; and

recovering processor state data from the saved uncorrupted data.

6. The method of claims **5**, further comprising:

reconciling the saved, uncorrupted data to recover the processor state data; and

initializing the first and second execution cores using the recovered processor state data.

7. The method of claim **6**, further comprising returning the processor to the redundant mode before initializing the first and second execution cores.

8. The method of claims **6**, wherein reconciling the saved, uncorrupted data comprises:

identifying corrupted processor state data in one of the first and second execution cores; and

updating the designated memory location associated with the one of the first and second execution cores using the saved, uncorrupted data of the other of the first and

second execution cores from the uncacheable portion of the main memory.

9. A machine readable medium on which are stored instructions that are executable by a dual execution core processor to implement a method for recovering from soft errors, the method comprising:

switching the processor to operate the dual execution cores independently when a soft error is detected;

executing an error recovery routine on each of the execution cores to recover a minimum set of processor state data from uncorrupted processor state data;

initializing each of the dual execution cores, using the minimum set of processor state data, wherein initializing each of the dual execution cores comprises switching the processor to operate the dual execution cores in lock step and copying the minimum set of processor state data to each of the dual execution cores; and

resuming execution of an interrupted program thread.

10. The machine readable medium of claim 9, wherein executing an error recovery routine comprises:

scanning parity-protected storage structures on each execution core to identify uncorrupted processor state data;

saving the uncorrupted processor state data to the designated memory locations associated with each execution core; and

recovering the minimum set of processor state data from the saved processor state data.

11. A computer system comprising:

a processor having first and second execution cores, the first and second execution cores to operate in lock step when the processor is in a redundant mode and to operate independently when the processor is in a split mode; and

a non-volatile memory in which is stored an error recovery routine to switch the processor to split mode when an error is detected in redundant mode, identify uncorrupted processor state data in each execution core, save the uncorrupted processor state data to an uncacheable portion of a main memory system, and restore the processor to redundant mode using a set of processor state data recovered from the uncorrupted processor state data.

12. The computer system of claim 11, wherein the error recovery routine switches the processor to split mode through a mode switch instruction.

13. The computer system of claim 12, wherein following the switch to split mode, each execution core independently executes a portion of the error recovery program to identify and save to a designated memory location uncorrupted processor data associated with its storage structures.

14. The computer system of claim 13, wherein one of the execution cores updates corrupted data from its storage structures using the saved, uncorrupted processor state data of the other execution core from the uncacheable portion of the main memory system.

15. The computer system of claim 14, wherein the error recovery routine restores the processor to redundant mode by through a mode switch instruction.

16. The computer system of claim 15, wherein the error recovery routine initializes each execution core using recovered processor state data from the uncacheable portion of the main memory system.

17. A computer system, comprising:

a processor having a first execution core and a second execution core, the first and second execution cores operate in lock step when the processor is in a redundant execution mode and operate independently when the processor is in a split execution mode; and

a non-volatile memory in which are stored instructions that may be implemented by the processor to implement a method for recovering from soft errors when the processor is in redundant execution mode, the method comprising

switching the processor to split execution mode;

saving uncorrupted processor state data from each execution core to a designated memory location;

recovering a minimal set of processor state data from the saved, uncorrupted processor state data, wherein recovering the minimal set of processor state data comprises replacing corrupted processor state data from one of the execution cores with uncorrupted processor state data saved from the other of the execution cores; and

initializing each execution core with the minimal set of processor state data.

18. The computer system of claim 17, wherein initializing each execution core comprises:

switching the processor to redundant execution mode; and

copying the processor state data from the designated memory locations to their associated execution cores.

19. A computer system, comprising:

a processor having a first execution core and a second execution core, the first and second execution cores operate in lock step when the processor is in a redundant execution mode and operate independently when the processor is in a split execution mode, wherein each execution core includes first and second execution clusters that operate in lock step when the processor is in split mode; and

a non-volatile memory in which are stored instructions that may be implemented by the processor to implement a method for recovering from soft errors when the processor is in redundant execution mode, the method comprising

switching the processor to split execution mode;

saving uncorrupted processor state data from each execution core to a designated memory location;

recovering a minimal set of processor state data from the saved, uncorrupted processor state data; and

initializing each execution core with the minimal set of processor state data.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 6,625,749 B1
DATED : September 23, 2003
INVENTOR(S) : Quach

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 1,
Line 67, after "mode or a", delete "[Functionalredundant redundant]".

Column 3,
Line 49, after "or", delete "[bootstrapping]".

Column 10,
Line 37, after "copied", delete "[570]".

Signed and Sealed this

Second Day of December, 2003



JAMES E. ROGAN
Director of the United States Patent and Trademark Office

intel

MultiProcessor Specification

Version 1.4

May 1997

THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.

A license is hereby granted to copy and reproduce this specification for internal use only.

No other license, express or implied, by estoppel or otherwise, to any other intellectual property rights is granted herein.

Intel disclaims all liability, including liability for infringement of any proprietary rights, relating to implementation of information in this specification. Intel does not warrant or represent that such implementation(s) will not infringe such rights.

Intel Corporation and Intel's FASTPATH are not affiliated with Kinetics, a division of Excelan, Inc. or its FASTPATH trademark or products.

* Third-party trademarks are the property of their respective owners.

Additional copies of this document or other Intel literature may be obtained from:

Intel Corporation
Literature Center
P.O. Box 7641
Mt. Prospect IL 60056-7641

or call 800-879-4683

printed on recycled paper

Copyright © 1993-1997. Intel Corporation, All Rights Reserved.

Revision History

Revision	Revision History	Date
	Pre-release Version 1.0. Formerly called "PC+MP Specification"	10/27/93
-001	Version 1.1. Resolves conflicts with MCA-based systems. The following changes have been made: 1. Two MP feature information bytes were moved from the BIOS System Configuration Table to the RESERVED area of the MP Floating Pointer Structure. 2. If the MP Floating Pointer Structure is present, it indicates that the system is MP-compliant, in accordance with this specification. (Previously, this was indicated by bit 0 of MP Feature Information Byte 1.) 3. One more hardware default system configuration was added for MCA+PCI with the integrated APIC.	4/11/94
-002	Minor technical corrections.	6/1/94
-003	Added Appendix D: Multiple I/O APIC Multiple PCI Bus Systems.	9/1/94
-004	Version 1.4. Added extended configuration table to improve support for multiple PCI bus configurations and improve future expandability. Made clarifications to Appendix B.	7/1/95
-005	Added Appendix E: Errata. Includes information for hierarchical PCI bus.	08/15/96
-006	Appendix E Update. Included Byte 2, Bit 6 information to MP Floating Point Structure section (4.1).	05/12/97

Table of Contents

Chapter 1 Introduction

1.1 Goals 1-1
1.2 Features of the Specification 1-2
1.3 Scope 1-2
1.4 Target Audience 1-3
1.5 Organization of This Document 1-3
1.6 Conventions Used in This Document 1-4
1.7 For More Information 1-4

Chapter 2 System Overview

2.1 Hardware Overview 2-2
2.1.1 System Processors 2-2
2.1.2 Advanced Programmable Interrupt Controller 2-3
2.1.3 System Memory 2-4
2.1.4 I/O Expansion Bus 2-4
2.2 BIOS Overview 2-5
2.3 Operating System Overview 2-5

Chapter 3 Hardware Specification

3.1 System Memory Configuration 3-1
3.2 System Memory Cacheability and Shareability 3-2
3.3 External Cache Subsystem 3-4
3.4 Locking 3-4
3.5 Posted Memory Write 3-5
3.6 Multiprocessor Interrupt Control 3-5
3.6.1 APIC Architecture 3-5
3.6.2 Interrupt Modes 3-6
3.6.2.1 PIC Mode 3-7
3.6.2.2 Virtual Wire Mode 3-9
3.6.2.3 Symmetric I/O Mode 3-11
3.6.3 Assignment of System Interrupts to the APIC Local Unit 3-12
3.6.4 Floating Point Exception Interrupt 3-12
3.6.5 APIC Memory Mapping 3-12

3.6.6 APIC Identification 3-13
3.6.7 APIC Interval Timers 3-13
3.6.8 Multiple I/O APIC Configurations 3-13
3.7 RESET Support 3-14
3.7.1 System-wide RESET 3-14
3.7.2 System-wide INIT 3-15
3.7.3 Processor-specific INIT 3-15
3.8 System Initial State 3-16
3.9 Support for Fault-resilient Booting 3-16

Chapter 4 MP Configuration Table

4.1 MP Floating Pointer Structure 4-3
4.2 MP Configuration Table Header 4-5
4.3 Base MP Configuration Table Entries 4-6
4.3.1 Processor Entries 4-7
4.3.2 Bus Entries 4-10
4.3.3 I/O APIC Entries 4-12
4.3.4 I/O Interrupt Assignment Entries 4-12
4.3.5 Local Interrupt Assignment Entries 4-15
4.4 Extended MP Configuration Table Entries 4-17
4.4.1 System Address Space Mapping Entries 4-18
4.4.2 Bus Hierarchy Descriptor Entries 4-21
4.4.3 Compatibility Bus Address Space Modifier Entries 4-22

Chapter 5 Default Configurations

5.1 Discrete APIC Configurations 5-2
5.2 Integrated APIC Configurations 5-4
5.3 Assignment Of I/O Interrupts To The APIC I/O Unit 5-6
5.3.1 EISA and IRQ13 5-7
5.3.2 Level-triggered Interrupt Support 5-7
5.4 Assignment Of System Interrupts To The APIC Local Unit 5-7

Appendix A System BIOS Programming Guidelines

A.1 BIOS Post Initialization A-1
A.2 Controlling the Application Processors A-2
A.3 Programming the APIC for Virtual Wire Mode A-2
A.4 Constructing the MP Configuration Table A-4

Appendix B Operating System Programming Guidelines

B.1 Operating System Boot-up B-1
B.2 Operating System Booting and Self-configuration B-2
B.3 Interrupt Mode Initialization and Handling B-2
B.4 Application Processor Startup B-3
B.4.1 USING INIT IPI B-4
B.4.2 USING STARTUP IPI B-5
B.5 AP Shutdown Handling B-5
B.6 Other IPI Applications B-6
B.6.1 Handling Cache Flush B-6
B.6.2 Handling TLB Invalidation B-6
B.6.3 Handling PTE Invalidation B-6
B.7 Spurious APIC Interrupts B-6
B.8 Supporting Unequal Processors B-7

Appendix C System Compliance Checklist

Appendix D Multiple I/O APIC Multiple PCI Bus Systems

D.1 Interrupt Routing with Multiple APICs D-1
D.1.1 Variable Interrupt Routing D-1
D.1.2 Fixed Interrupt Routing D-2
D.2 Bus Entries in Systems with More Than One PCI Bus D-3
D.3 I/O Interrupt Assignment Entries for PCI Devices D-3

Appendix E Errata

Glossary

Figures

1-1. Conceptual Overview 1-1
1-2. Memory Layout Conventions 1-4
2-1. Multiprocessor System Architecture 2-2
2-2. APIC Configuration 2-3
3-1. System Memory Address Map 3-2
3-2. PIC Mode 3-8
3-3. Virtual Wire Mode via Local APIC 3-9
3-4. Virtual Wire Mode via I/O APIC 3-10
3-5. Symmetric I/O Mode 3-11
3-6. Multiple I/O APIC Configurations 3-14
4-1. MP Configuration Data Structures 4-1
4-2. MP Floating Pointer Structure 4-3
4-3. MP Configuration Table Header 4-5
4-4. Processor Entry 4-7
4-5. Bus Entry 4-10
4-6. I/O APIC Entry 4-12
4-7. I/O Interrupt Entry 4-13
4-8. Local Interrupt Entry 4-15
4-9. System Address Space Entry 4-18
4-10. Example System with Multiple Bus Types and Bridge Types 4-19
4-11. Bus Hierarchy Descriptor Entry 4-21
4-12. Compatibility Bus Address Space Modifier Entry 4-23
5-1. Default Configuration for Discrete APIC 5-3
5-2. Default Configuration for Integrated APIC 5-5

Tables

1-1. Document Organization 1-3
3-1. Memory Cacheability Map 3-3
3-2. APIC Versions 3-6
4-1. MP Floating Pointer Structure Fields 4-3
4-2. MP Configuration Table Header Fields 4-6
4-3. Base MP Configuration Table Entry Types 4-7
4-4. Processor Entry Fields 4-8
4-5. Intel486™ and Pentium® Processor Signatures 4-9

4-6. Feature Flags from CPUID Instruction 4-9
4-7. Bus Entry Fields 4-10
4-8. Bus Type String Values 4-11
4-9. I/O APIC Entry Fields 4-12
4-10. I/O Interrupt Entry Fields 4-14
4-11. Interrupt Type Values 4-15
4-12. Local Interrupt Entry Fields 4-16
4-13. Extended MP Configuration Table Entry Types 4-17
4-14. System Address Space Mapping Entry Fields 4-19
4-15. Bus Hierarchy Descriptor Entry Fields 4-22
4-16. Compatibility Bus Address Space Modifier Entry Fields 4-24
4-17. Predefined Range Lists 4-24
5-1. Default Configurations 5-2
5-2. Default Configuration Interrupt Assignments 5-6
5-3. Assignment of System Interrupts to APIC Local Unit 5-7
D-1. I/O Interrupt Entry Source Bus IRQ Field for PCI Devices D-3

Examples

A-1. Programming Local APIC for Virtual Wire Mode A-3
B-1. Universal Start-up Algorithm B-3

1 Introduction

The MultiProcessor Specification, hereafter known as the "MP specification," defines an enhancement to the standard to which PC manufacturers design DOS-compatible systems. MP-capable operating systems will be able to run without special customization on multiprocessor systems that comply with this specification. End users who purchase a compliant multiprocessor system will be able to run their choice of operating systems.

The MP specification covers PC/AT-compatible MP platform designs based on Intel processor architectures and Advanced Programmable Interrupt Controller (APIC) architectures. The term "PC/AT-compatible" here refers to the software-visible components of the PC/AT, not to hardware features, such as the bus implementation, that are not visible to software. An implementation of this specification may incorporate one or more industry standard buses, such as ISA, EISA, MCA, PCI, or other OEM-specific buses.

1.1 Goals

The intent of this specification is to establish an MP Platform interface standard that extends the performance of the existing PC/AT platform beyond the traditional single processor limit, while maintaining 100% PC/AT binary compatibility.

The ultimate goal is to enable scalable, high-end workstations and enterprise server systems that provide computer users with superior price/performance and have the ability to execute all existing AT binaries, as well as MP-ready software packages on shrink-wrapped MP operating systems. Figure 1-1 shows that at the heart of the specification are the data structures that define the configuration of the MP system. The BIOS constructs the MP configuration data structures, presenting the hardware in a known format to the standard device drivers or to the hardware abstraction layer of the operating system. The specification details default hardware configurations, and, for added flexibility, outlines extensions to the standard BIOS.

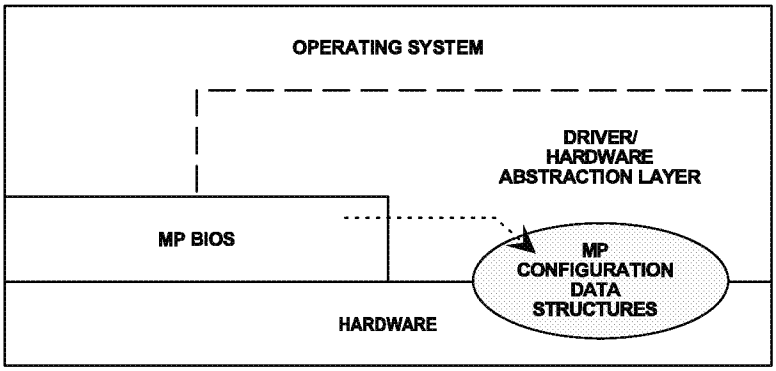


Figure 1-1. Conceptual Overview

1.2 Features of the Specification

The MP specification includes the following features:

- A multiprocessor extension to the PC/AT platform that runs all existing uniprocessor shrink-wrapped binaries, as well as MP binaries.
- Support for symmetric multiprocessing with one or more processors that are Intel architecture instruction set compatible, such as the CPUs in the Intel486™ and the Pentium® processor family.
- Support for symmetric I/O interrupt handling with the APIC, a multiprocessor interrupt controller.
- Flexibility to use a BIOS with minimal MP-specific support.
- An optional MP configuration table to communicate configuration information to an MP operating system.
- Incorporation of ISA and other industry standard buses, such as EISA, MCA, VL and PCI buses in MP-compliant systems.
- Requirements that make secondary cache and memory bus implementation transparent to software.

1.3 Scope

There are many vendors building innovative MP systems based on Intel architectures today. Intel supports and encourages vendors to develop advanced approaches to the technological requirements of today's computing environments. In no way does the MP specification prevent system manufacturers from adding their own unique value to MP systems. This specification does not define the only way that multiprocessor systems can be implemented. Vendors may, for example, create noncompliant, high-performance, scalable multiprocessor systems that do not have the PC/AT compatibility required by this specification. Hardware vendors will always have the option of offering one or more customized operating systems that support the unique, value-added capabilities that they have designed into their systems. End users will be able to benefit from the additional capabilities that these vendors offer.

The MP specification benefits PC vendors who wish to offer MP-enabled systems without having to invest in the customization of one or more operating systems. This specification focuses on providing a standard mechanism to add MP support to personal computers built around the PC/AT hardware standard. With that goal, the specification covers only the minimum set of capabilities required to extend the PC/AT platform to be MP-capable. The hardware required to implement the MP specification is kept to a minimum, as follows:

- One or more processors that are Intel architecture instruction set compatible, such as the CPUs in the Intel486 or Pentium processor family.
- One or more APICs, such as the Intel 82489DX Advanced Programmable Interrupt Controller or the integrated APIC, such as that on the Intel Pentium 735\90 and 815\100 processors, together with a discrete I/O APIC unit.
- The necessary supporting electronics to duplicate the initialization and signaling protocol described in this specification.
- PC/AT-compliant hardware.

In addition to the hardware requirements, this document also specifies MP features that are visible to the BIOS and operating system. However, it is important to understand that as hardware technology progresses, the functions performed by the BIOS may change in accordance with the hardware technology. **ONLY THE INTERFACE TO THE OPERATING SYSTEM LEVEL IS EXPECTED TO REMAIN CONSTANT.**

This specification does not address issues relating to the processor's System Management Mode (SMM).

1.4 Target Audience

This document is intended for the following users:

- OEMs who will be creating PC/AT-compatible, MP-ready hardware based on this specification.
- BIOS developers, either those who create general-purpose BIOS products or those who modify these products to suit specific MP hardware.
- Operating-system developers who will be adapting MP operating systems to run on the class of MP-ready platform specified here.

1.5 Organization of This Document

Table 1-1 shows the organization of this document.

Table 1-1. Document Organization

Chapter	Description
2	Overview of the MultiProcessor Specification
3	Specification of the MP hardware
4	Specification of MP configuration information available to OS
5	Specification of default hardware configurations
Appendix A	Guidelines for MP BIOS programming
Appendix B	Guidelines for MP operating system programming
Appendix C	Checklist for hardware compliance to the specification
Appendix D	Clarifications for multiple I/O APIC, multiple PCI bus systems
Glossary	Definitions of specialized terms used in this document

1.6 Conventions Used in This Document

Signal names that are followed by the character # represent active low signals. For example, FERR# is active when at its low-voltage state.

Throughout this document, the Intel 82489DX APIC is referred to as the "discrete APIC." The term "integrated APIC" is used to refer to an APIC integrated with other system components, such as the Pentium 735\90 and 815\100 processors. This specification uses the term APIC to refer to both discrete and integrated versions.

The processors of the Intel486 and Pentium processor family are "little endian" machines. This means that the low-order byte of a multibyte data item in memory is at the lowest address, while the high-order byte is at the highest address. Illustrations of data structures in memory show the lowest addresses at the bottom and the highest addresses at the top of the illustration, as shown in Figure 1-2. Bit positions are numbered from right to left.

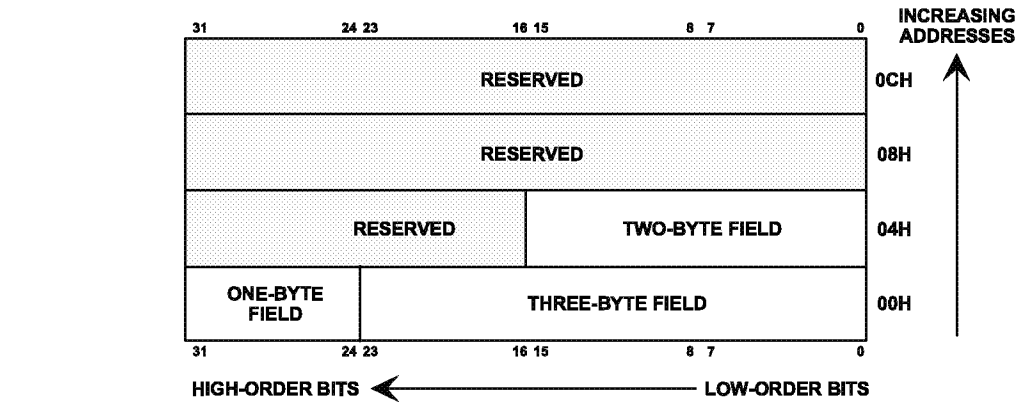


Figure 1-2. Memory Layout Conventions

In some memory layout descriptions, certain fields are marked **RESERVED**. Software should initialize these fields as binary zeros, but should otherwise treat them as having a future, though unknown effect. **SOFTWARE SHOULD AVOID ANY DEPENDENCE ON THE VALUES IN THE RESERVED FIELDS.**

1.7 For More Information

For more information, refer to any of the following documents:

- *82489DX Advanced Programmable Interrupt Controller* (data book), Intel order number 290446
- *Intel486 Microprocessor Programmer's Reference Manual,* Intel order number 240486
- *Intel Processor Identification with the CPUID Instruction* AP-485, Intel order number 241618
- *Pentium Processor User's Manual*, Intel order number 241428

2
System Overview

In the realm of multiprocessor architectures, there are several conceptual models for tying together computing elements, and there are a variety of interconnection schemes and details of implementation. Figure 2-1 shows the general structure of a design based on the MP specification. The MP specification's model of multiprocessor systems incorporates a tightly-coupled, shared-memory architecture with a distributed interprocessor and I/O interrupt capability. It is fully symmetric; that is, all processors are functionally identical and of equal status, and each processor can communicate with every other processor. There is no hierarchy, no master-slave relationship, no geometry that limits communication only to "neighboring" processors. The model is symmetric in two important respects:

- *Memory symmetry.* Memory is symmetric when all processors share the same memory space and access that space by the same addresses. Memory symmetry offers a very important feature—the ability for all processors to execute a single copy of the operating system. Any existing system and application software will execute the same, regardless of the number of processors installed in a system.
- *I/O symmetry.* I/O is symmetric when all processors share access to the same I/O subsystem (including I/O ports and interrupt controllers) and any processor can receive interrupts from any source. Some multiprocessor systems that have symmetric access to memory are actually asymmetric with regard to I/O interrupts, because they dedicate one processor to interrupt functions. I/O symmetry helps eliminate the potential of an I/O bottleneck, thereby increasing system scalability.

With both memory and I/O symmetry, a system that complies with the MP specification can achieve hardware scalability as well as support software standardization. Based on this kind of scalable architecture, systems developers can produce systems that span a wide range of price and performance, and that all execute the same binaries.

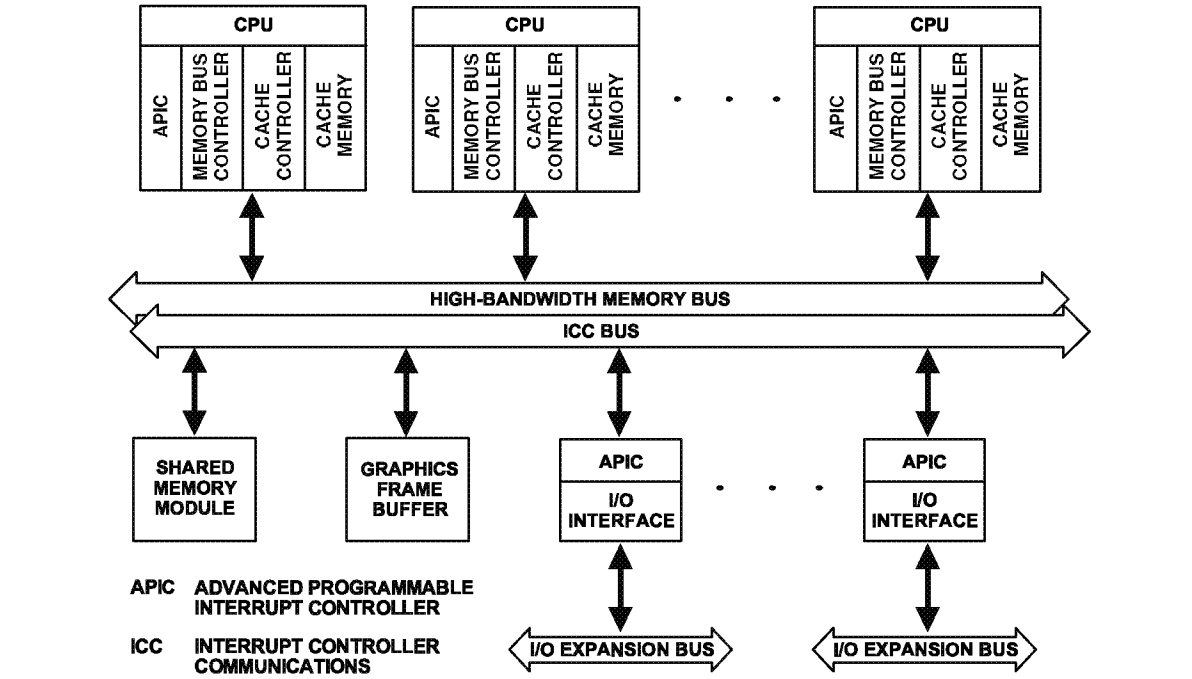


Figure 2-1. Multiprocessor System Architecture

2.1 Hardware Overview

The MP specification defines a system architecture based on the following hardware components:

- One or more processors that are Intel architecture instruction set compatible, such as the CPUs in the Intel486 and the Pentium processor family.
- One or more APICs, such as the Intel 82489DX Advanced Programmable Interrupt Controller or the integrated APIC on the Pentium 735\90 and 815\100 processors.
- Software-transparent cache and shared memory subsystem.
- Software-visible components of the PC/AT platform.

2.1.1 System Processors

To maintain compatibility with existing PC/AT software products, this specification is based on the Intel486 and the Pentium processor family. To achieve a minimum level of MP system performance, two or more processors that are Intel architecture instruction set compatible are required.

Figure 2-2 gives a different point of view of a compliant system, showing the configuration of the APICs with respect to the CPUs. While all processors in a compliant system are functionally identical, this specification classifies them into two types: the bootstrap processor (BSP) and the application processors (AP). Which processor is the BSP is determined by the hardware or by the hardware in conjunction with the BIOS. This differentiation is for convenience and is in effect

only during the initialization and shutdown processes. The BSP is responsible for initializing the system and for booting the operating system; APs are activated only after the operating system is up and running. CPU1 is designated as the BSP. CPU2, CPU3, and so on, are designated as the APs.

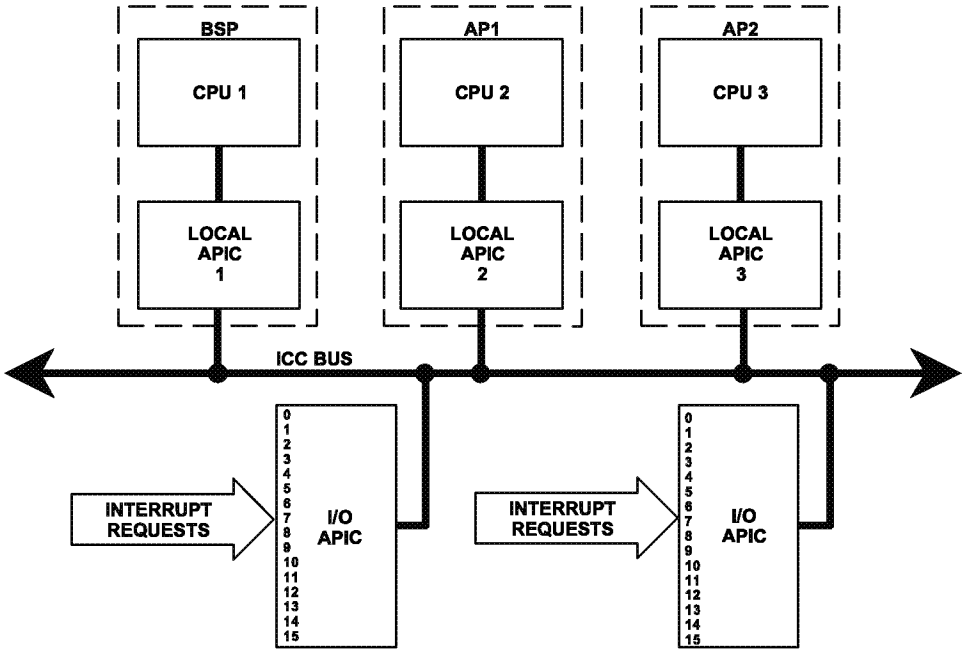


Figure 2-2. APIC Configuration

2.1.2 Advanced Programmable Interrupt Controller

The Advanced Programmable Interrupt Controller (APIC) is based on a distributed architecture in which interrupt control functions are distributed between two basic functional units, the local unit and the I/O unit. The local and I/O units communicate through a bus called the Interrupt Controller Communications (ICC) bus, as shown in Figure 2-2.

In a multiprocessor system, multiple local and I/O APIC units operate together as a single entity, communicating with one another over the ICC bus. The APIC units are collectively responsible for delivering interrupts from interrupt sources to interrupt destinations throughout the multiprocessor system.

The APICs help achieve the goal of scalability by doing the following:

- Off-loading interrupt-related traffic from the memory bus, making the memory bus more available for processor use.
- Helping processors share the interrupt processing load with other processors.

The APICs help achieve the goal of AT-compatibility by cooperating with 8259A-equivalent PICs in the system.

The local APIC units also provide interprocessor interrupts (IPIs), which allow any processor to interrupt any other processor or set of processors. There are several types of IPIs. Among them, the INIT IPI and the STARTUP IPI are specifically designed for system startup and shutdown.

Each local APIC has a Local Unit ID Register and each I/O APIC has an I/O Unit ID Register. The ID serves as a physical name for each APIC unit. It is used by software to specify destination information for I/O interrupts and interprocessor interrupts, and is also used internally for accessing the ICC bus.

Due to the distributed architecture, the APIC local and I/O units can be implemented in either a single chip, such as Intel's 82489DX interrupt controller, or they can be integrated with other parts of the system's components. For example, the local APIC may be integrated with the CPU chip, such as Intel's Pentium processors (735\90, 815\100), and the I/O APIC may be integrated with the I/O chipset, such as Intel's 82430 PCI-EISA bridge chipset.

2.1.3 System Memory

A system that complies with the MP specification uses the standard AT memory architecture. All memory is allocated for system memory with the exception of addresses 0A_0000h through 0F_FFFFh and 0FFFE_0000h through 0FFFF_FFFFh, which are reserved for I/O devices and the BIOS.

Compared to a uniprocessor system, a symmetric multiprocessor system imposes a high demand for memory bus bandwidth. The demand is proportional to the number of processors on the memory bus. To reduce memory bus bandwidth limitations, an implementation of this specification should use a secondary cache that has high-performance features, such as a write-back update policy and a snooping cache-consistency protocol. A secondary cache can push the scalability limit upward by reducing bus traffic and increasing bus bandwidth.

While both secondary caches and memory bus controllers are critical components for high performance in a symmetric multiprocessor system, this specification does not detail their implementation, requiring only that they be totally software transparent.

2.1.4 I/O Expansion Bus

The MP specification provides a multiprocessor extension to the industry-standard PC/AT platform. The term "industry-standard PC/AT platform" here refers to the software-visible components of the PC/AT. The standard does not designate a specific bus architecture. All industry-standard buses, such as ISA, EISA, MCA, VL, and PCI, can be incorporated in the design. Systems developers can implement one or more bus types in their designs, depending on the systems' I/O performance or capacity requirements.

2.2 BIOS Overview

A BIOS functions as an insulator between the hardware on one hand, and the operating system and applications software on the other. A standard uniprocessor BIOS performs the following functions:

- Tests system components.
- Builds configuration tables to be used by the operating system.
- Initializes the processor and the rest of the system to a known state.
- Provides run-time device-oriented services.

For a multiprocessor system, the BIOS may perform the following additional functions:

- Pass configuration information to the operating system that identifies all processors and other multiprocessing components of the system.
- Initialize all processors and the rest of the multiprocessing components to a known state.

This specification allows a wide range of capability in the BIOS. At the minimal end of the capability scale, the system developer can simply insert an MP floating pointer structure in the standard BIOS. The cost of this level of simplicity in the BIOS, however, is that the system developer has less flexibility in the design of the hardware. At the maximal end of the BIOS capability scale might be a BIOS that dynamically configures the system to provide resilience in the face of component malfunctions.

BIOS developers should read Chapters 3, 4, 5, and Appendix A to understand the tradeoffs between hardware and BIOS capabilities.

2.3 Operating System Overview

Enabling the creation of shrink-wrapped MP operating systems is one of the principal goals of this specification. This goal is achieved by allowing a flexible balance between the capabilities of the hardware and those of the BIOS. The potentially vast variety of hardware configurations is reduced by the BIOS to a few simple scenarios that can be readily handled by the low-level boot-up phase of the operating system.

Operating-system developers should read Chapters 3, 4, and 5, and Appendixes A and B to fully understand the interface between the BIOS and the operating system.

3
Hardware Specification

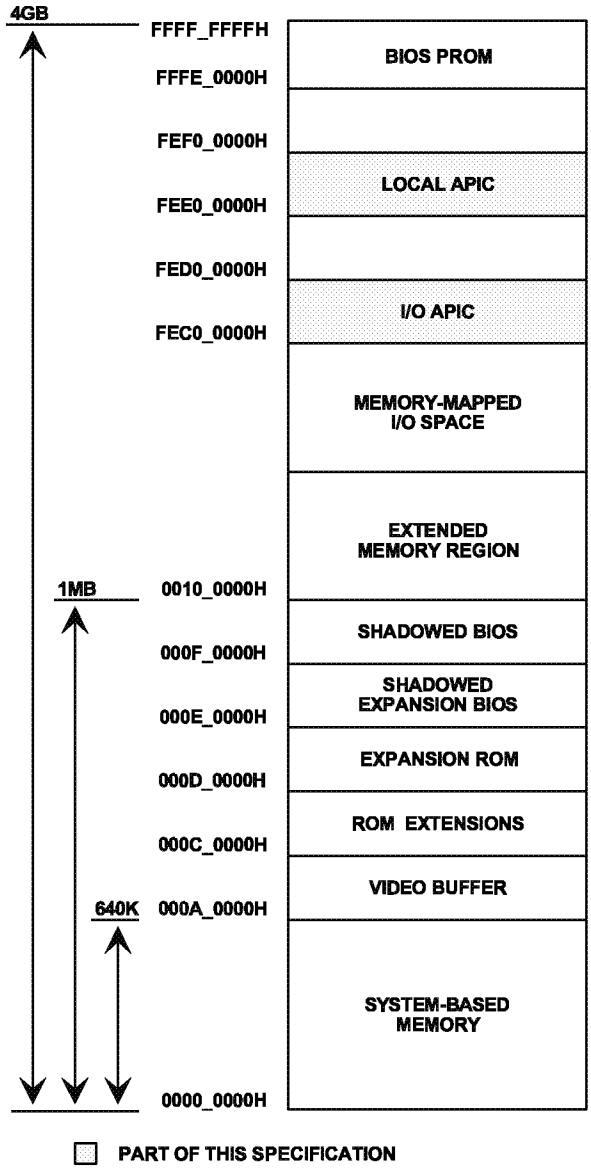
This section outlines the minimal set of common hardware features necessary for the operating system to operate on multiple hardware platforms. The MP hardware specification defines how the components mentioned in Chapter 2 are implemented. Compliance to the specification involves the following aspects of hardware implementation:

- System memory configuration
- System memory cacheability and shareability
- External cache implementation requirements
- Control of memory contention (locking)
- Ordering of posted memory writes
- Multiprocessor interrupt control
- Reset support
- Interval timer usage
- Support for fault-resilient booting

While the bulk of the MP hardware specification pertains to multiprocessor interrupt control, other areas also require some attention. The following sections take up each of these topics in turn.

3.1 System Memory Configuration

The MP memory specifications are based on the standard PC/AT memory map, which currently has a physical memory space of four gigabytes, as shown in see Figure 3-1. Physical memory should begin at 0 and be contiguous. Memory-mapped I/O devices should be mapped at the top of physical memory. The APIC default base memory addresses defined by this specification are 0FEC0_0000h and 0FEE0_0000h.



UNSHADED ADDRESS REGIONS ARE FOR REFERENCE ONLY AND SHOULD NOT BE CONSTRUED AS THE SOLE DEFINITION OF A PC/AT-COMPATIBLE ADDRESS SPACE.

Figure 3-1. System Memory Address Map

3.2 System Memory Cacheability and Shareability

The cacheability and shareability of the physical memory space are defined in Table 3-1. The address space reserved for the local APIC is used by each processor to access its own local APIC. The address space reserved for the I/O APIC must be shareable by all processors to permit dynamic reconfiguration.

Table 3-1. Memory Cacheability Map

Addresses (in hex)	Size	Description	Shared by All Processors?	Cacheable?	Comment
0000_0000h – 0009_FFFFh	640KB	Main memory	Yes	Yes	
000A_0000h – 000B_FFFFh	128KB	Display buffer for video adapters	Yes	No	
000C_0000h – 000D_FFFFh	128KB	ROM BIOS for add-on cards	Yes	Yes	
000E_0000h – 000F_FFFFh	128KB	System ROM BIOS	Yes	Yes	
0010_0000h – 0FEBF_FFFFh		Main memory	Yes	Yes	Maximum address depends on total memory installed in the system.
Not specified.		Memory-mapped I/O devices	Yes[2]	Not specified	Top unused memory
0FEC0_0000h – 0FECF_FFFFh[1]		APIC I/O unit	Yes	No	Refer to the register description in the APIC data book.
0FED0_0000h – 0FEDF_FFFFh		Reserved for memory-mapped I/O devices	Yes[2]	Not specified	
0FEE0_0000h- 0FEEF_FFFFh[1]		APIC Local Unit	No	No	Refer to the register description in the APIC data book.
0FEF0_0000h – 0FFFD_FFFFh		Reserved for memory-mapped I/O devices	Yes[2]	Not specified	
0FFFE_0000h – 0FFFF_FFFFh	128KB	Initialization ROM	Yes	Not specified	

NOTES:

1. These addresses are part of this specification. The other address regions in this table are shown for reference only, and should not be construed as the sole definition of a PC/AT-compatible address space format or cache.
2. Any memory-mapped device should be shareable unless the nature of the device is that there is one device per processor.

3.3 External Cache Subsystem

Intel-compatible processors support multiprocessing both on the processor bus and on a memory bus, both with and without secondary cache units. Due to the high bandwidth demands of multiprocessor systems, external caches are often employed to improve performance. The existence and implementation details of external caches are not a part of this specification. However, when external caches are used, they must conform to certain requirements with regard to the following design issues:

- *Maintaining cache coherency*—When one processor accesses data cached in another processor's cache, it must not receive incorrect data. If it modifies data, all other processors that access that data also must not receive stale data. External caches must maintain coherency among themselves, and with the main memory, internal caches, and other bus master DMA devices.
- *Cache flushing*—The processor can generate special flush and write-back bus cycles that must be used by external caches in a manner that maintains cache coherency. The actual responses are implementation-specific and may vary from design to design. A program can initiate hardware cache flushing by executing a WBINVD instruction. This instruction is only guaranteed to flush the caches of the local processor. See Appendix B for system-wide flushing mechanisms. Given that cache coherency is maintained by hardware, there is no need for software to issue cache flush instructions under normal circumstances.
- *Reliable communication*—All processors must be able to communicate with each other in a way that eliminates interference when more than one processor accesses the same area in memory simultaneously. The processor uses the LOCK# signal for this purpose. External caches must ensure that all locked operations are visible to other processors.
- *Write ordering*—In some circumstances, it is important that memory writes be observed externally in precisely the same order as programmed. External write buffers must maintain the write ordering of the processor.

3.4 Locking

To protect the integrity of certain critical memory operations, Intel-compatible processors provide an output signal called LOCK#. For any given memory access, LOCK# is asserted once, but may remain asserted for as many memory bus cycles as required to complete the memory operation. It is the responsibility of the system hardware designers to use this signal to control memory accesses among processors.

A compliant system in multiprocessor mode must guarantee atomicity of locked-aligned memory operations; however, the implementation is not specified in this specification. A compliant system must lock at least the area of memory defined by the destination operand. A specific implementation may lock a broader area—it may even lock the entire bus. Therefore, software must consider this behavior.

To guarantee AT compatibility, locking of misaligned memory operations over other AT-compatible buses in the compliant system must be strictly implemented in accordance with the bus specifications. A compliant system may not be required to support the misaligned memory

operations over its internal shared memory bus, if it is AT compatible. Operating system and software developers must ensure that data is aligned if locked access is required, because lock operations on misaligned data are not guaranteed to work on all platforms.

3.5 Posted Memory Write

When controlling I/O devices, it is important that memory and I/O operations be carried out in the order programmed. Intel-compatible processors do not buffer I/O writes; thus, strict ordering among I/O operations is enforced by the processors.

To optimize memory performance, processors and chipsets often implement write buffers and writeback caches. Intel-compatible processors guarantee processor ordering on all internal cache and write buffer accesses. However, chipsets must also guarantee processor ordering on all external memory accesses.

For systems based on the integrated APIC, posting of memory writes may result in spurious interrupts for memory-mapped I/O devices using level-triggered interrupts. I/O device drivers must serialize instructions to ensure that the device interrupt clear command reaches the device before the EOI command reaches the APIC and handles the spurious interrupt in case one occurs.

3.6 Multiprocessor Interrupt Control

In an MP-compliant system, interrupts are controlled through the APIC. The following sections describe the APIC architecture and the three interrupt modes allowed in an MP-compliant system.

3.6.1 APIC Architecture

The Intel Advanced Programmable Interrupt Controller (APIC) is based on a distributed architecture. Interrupt control functions are distributed between two basic functional units: the local unit and the I/O unit. The local and I/O units communicate through a bus called the ICC bus. The I/O unit senses an interrupt input, addresses it to a local unit, and sends it over the ICC bus. The local unit that is addressed accepts the message sent by the I/O unit.

In an MP-compliant system, one local APIC per CPU is required. Depending on the total number of interrupt lines in an MP system, one or more I/O APICs may be used. The bus interrupt line assignments can be implementation-specific and can be defined by the MP configuration table described in Chapter 4.

The Intel 82489DX APIC is a "discrete APIC" implementation. The programming interface of the 82489DX APIC units serves as the base of the MP specification. Each APIC has a version register that contains the version number of a specific APIC implementation. The version register of the 82489DX family has a version number of "0*x*," where *x* is a four-bit hexadecimal number. Version number "1*x*" refers to Pentium processors with integrated APICs, such as the Pentium 735\90 and 815\100 processors, and *x* is a four-bit hexadecimal number.

The integrated APIC maintains the same programming interface as the 82489DX APIC. Table 3-2 describes the features specific to the integrated APIC.

Table 3-2. APIC Versions

APIC Type	Local APIC Version Register (hexadecimal)	Integrated APIC Features
82489DX APIC	0x	
Integrated APIC, i.e., Pentium processors (735\90, 815\100)	1x	STARTUP IPI. See Appendix B.4.2 for details. Programmable interrupt input polarity

NOTE:
x is a 4-bit hexadecimal number.

To encourage future extendibility and innovation, the Intel APIC architecture definition is limited to the programming interface of the APIC units. The ICC bus protocol and electrical specifications are considered implementation-specific. That is, while different versions of APIC implementations may execute the same binary software, different versions of APIC components may be implemented with different bus protocols or electrical specifications. Care must be taken when using different versions of the APIC in a system.

The APIC architecture is designed to be scaleable. The 82489DX APIC has an 8-bit ID register that can address from one to 255 APIC devices. Furthermore, the Logical Destination register for the 82489DX APIC supports 32 bits, which can address up to 32 devices. For small system implementations, the APIC ID register can be reduced to the least significant 4 bits and the Logical Destination register can be reduced to the most significant 8 bits.

To ensure software compatibility with all versions of APIC implementations, software developers must follow the following programming guidelines:

1. Assign an 8-bit APIC ID starting from zero.
2. Assign logical destinations starting from the most significant byte of the 32-bit register.
3. Program the APIC spurious vector to hexadecimal "*x*F," where *x* is a 4-bit hexadecimal number.

The following features are only available in the integrated APIC:

1. The I/O APIC interrupt input signal polarity can be programmable.
2. A new interprocessor interrupt, STARTUP IPI is defined.

In general, the operating system must use the STARTUP IPI to wake up application processors in systems with integrated APICs, but must use INIT IPI in systems with the 82489DX APIC. Refer to Appendix B, Section B.4, for application processor startup.

3.6.2 Interrupt Modes

The MP specification defines three different interrupt modes as follows:

1. *PIC Mode*—effectively bypasses all APIC components and forces the system to operate in single-processor mode.
2. *Virtual Wire Mode*—uses an APIC as a virtual wire, but otherwise operates the same as PIC Mode.
3. *Symmetric I/O Mode*—enables the system to operate with more than one processor.

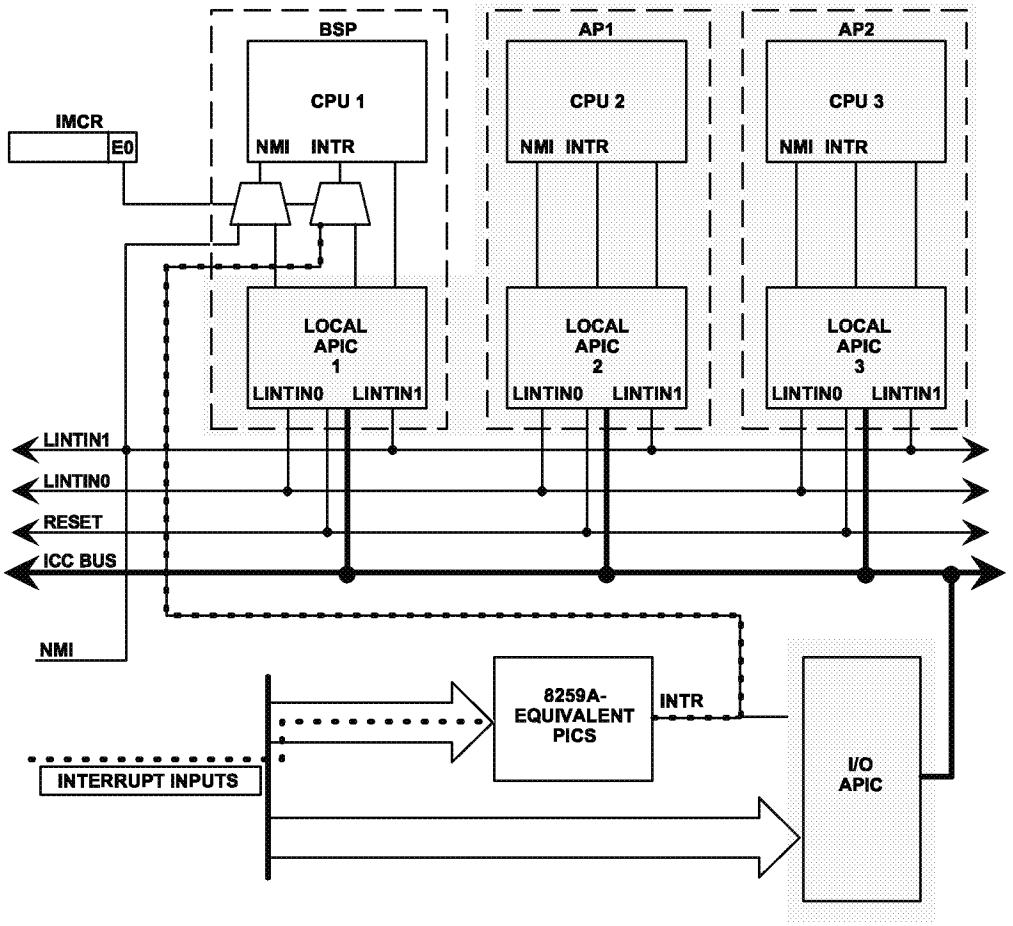
The first two interrupt modes, PIC Mode and Virtual Wire Mode, provide PC/AT-compatibility. At least one of these modes must be implemented in systems that comply with the MP specification. In these modes, full DOS compatibility with the uniprocessor PC/AT is provided by using the APICs in conjunction with standard 8259A-equivalent programmable interrupt controllers (PICs).

The third mode, Symmetric I/O Mode, must be implemented in addition to either PIC Mode or Virtual Wire Mode. An MP operating system is booted under either one of the two PC/AT-compatible modes. Later the operating system switches to Symmetric I/O Mode as it enters multiprocessor mode.

The interrupt modes are implemented by a combination of hardware and software. The hardware and programming specifications for each of these modes are further defined in the following subsections. BIOS programmers should refer to Appendix A, which includes information about programming the APIC for Virtual Wire Mode. Operating system programmers should refer to Appendix B, which provides more information about initializing the APIC for Symmetric I/O Mode.

3.6.2.1 PIC Mode

PIC Mode is software compatible with the PC/AT because it actually employs the same hardware interrupt configuration. As Figure 3-2 illustrates, the hardware for PIC Mode bypasses the APIC components by using an interrupt mode configuration register (IMCR). This register controls whether the interrupt signals that reach the BSP come from the master PIC or from the local APIC. Before entering Symmetric I/O Mode, either the BIOS or the operating system must switch out of PIC Mode by changing the IMCR.



SHADED AREAS INDICATE UNUSED CIRCUITS. DOTTED LINE SHOWS INTERRUPT PATH.

Figure 3-2. PIC Mode

The IMCR is supported by two read/writable or write-only I/O ports, 22h and 23h, which receive address and data respectively. To access the IMCR, write a value of 70h to I/O port 22h, which selects the IMCR. Then write the data to I/O port 23h. The power-on default value is zero, which connects the NMI and 8259 INTR lines directly to the BSP. Writing a value of 01h forces the NMI and 8259 INTR signals to pass through the APIC.

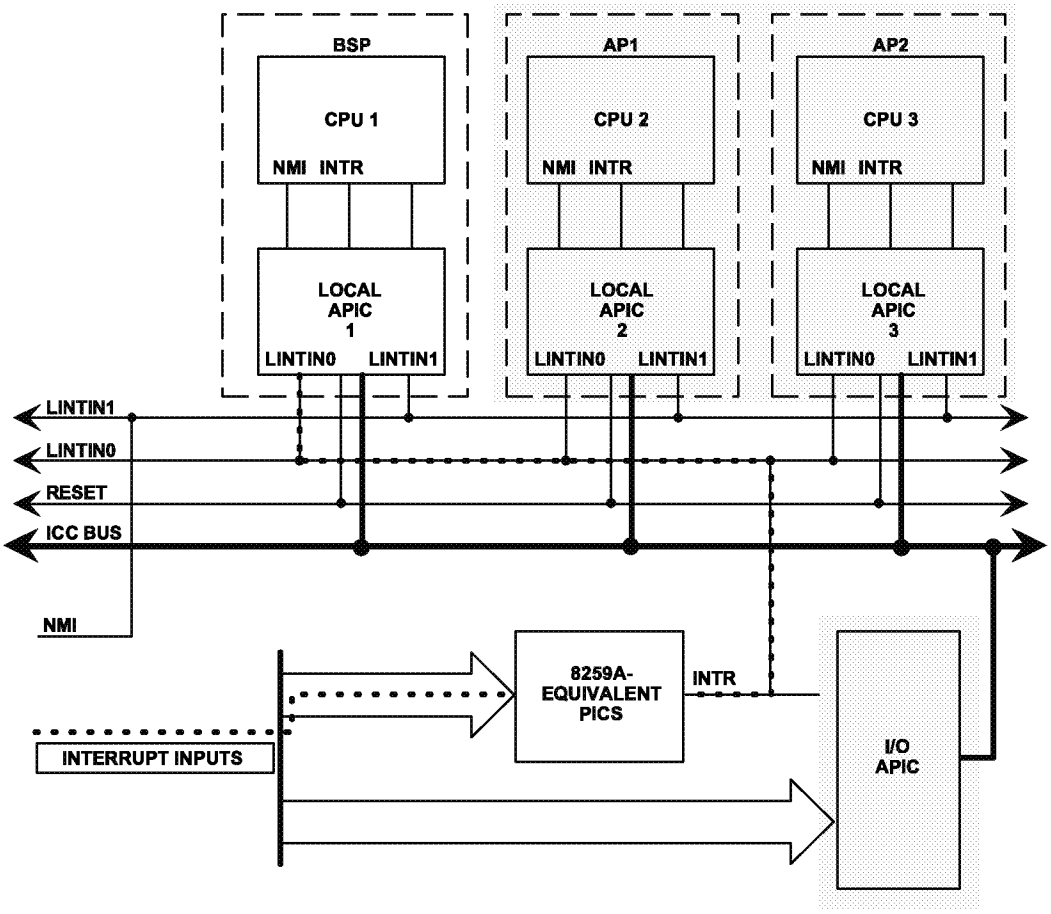
The IMCR must be cleared after a system-wide INIT or RESET to enable the PIC Mode as default. (Refer to Section 3.7 for information on the INIT and RESET signals.)

The IMCR is optional if PIC Mode is not implemented. The IMCRP bit of the MP feature information bytes (refer to Chapter 4) enables the operating system to detect whether the IMCR is implemented.

3.6.2.2 Virtual Wire Mode

Virtual Wire Mode provides a uniprocessor hardware environment capable of booting and running all DOS shrink-wrapped software.

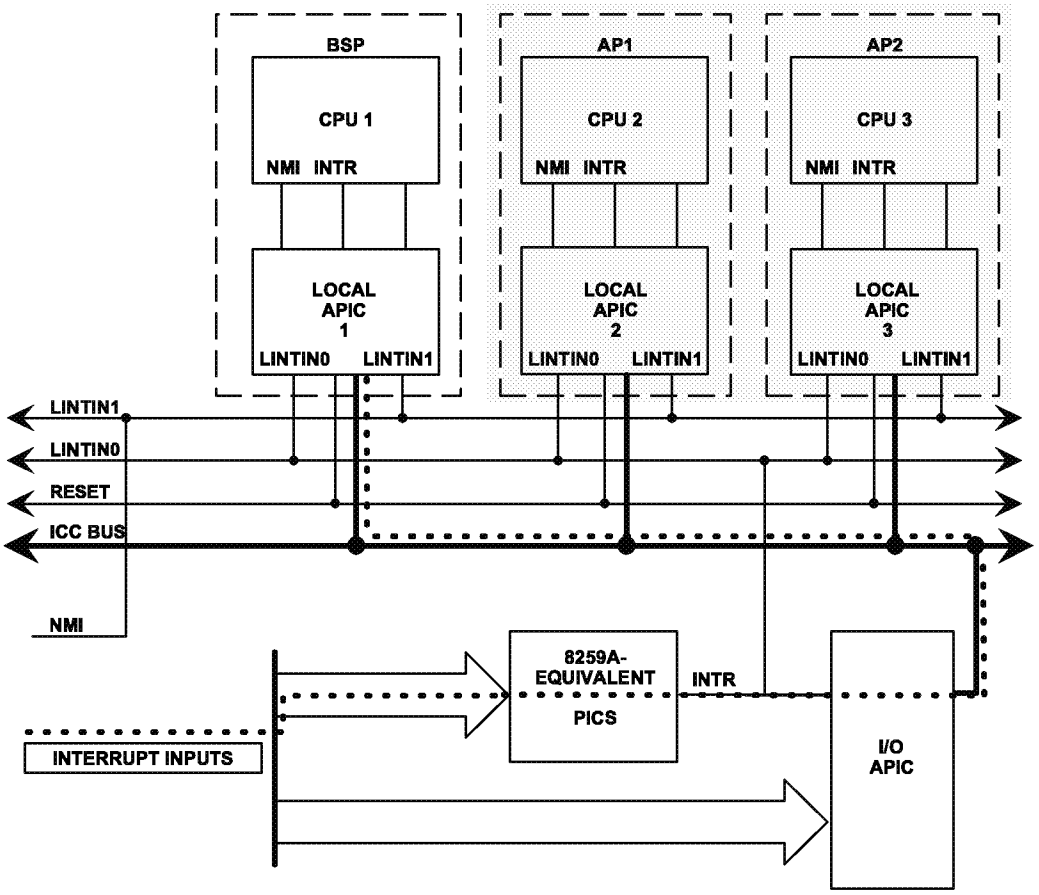
In Virtual Wire Mode, as shown in Figure 3-3, the 8259A-equivalent PIC fields all interrupts, and the local APIC of the BSP becomes a virtual wire, which delivers interrupts from the PIC to the BSP via the local APIC's local interrupt 0 (LINTIN0). The LINTIN0 pin of the local APIC is programmed as ExtINT, specifying to the APIC that the PIC is to serve as an external interrupt controller. Whenever the local APIC finds that a particular interrupt is of type ExtINT, it asserts the ExtINTA transaction along with the PINT interrupt to the processor. In this case, the I/O APIC is not used.



SHADED AREAS INDICATE UNUSED CIRCUITS. DOTTED LINE SHOWS INTERRUPT PATH.

Figure 3-3. Virtual Wire Mode via Local APIC

Figure 3-3 shows how Virtual Wire Mode can be implemented through the BSP's local APIC. It is also permissible to program the I/O APIC for Virtual Wire Mode, as shown in Figure 3-4. In this case the interrupt signal passes through both the I/O APIC and the BSP's local APIC.

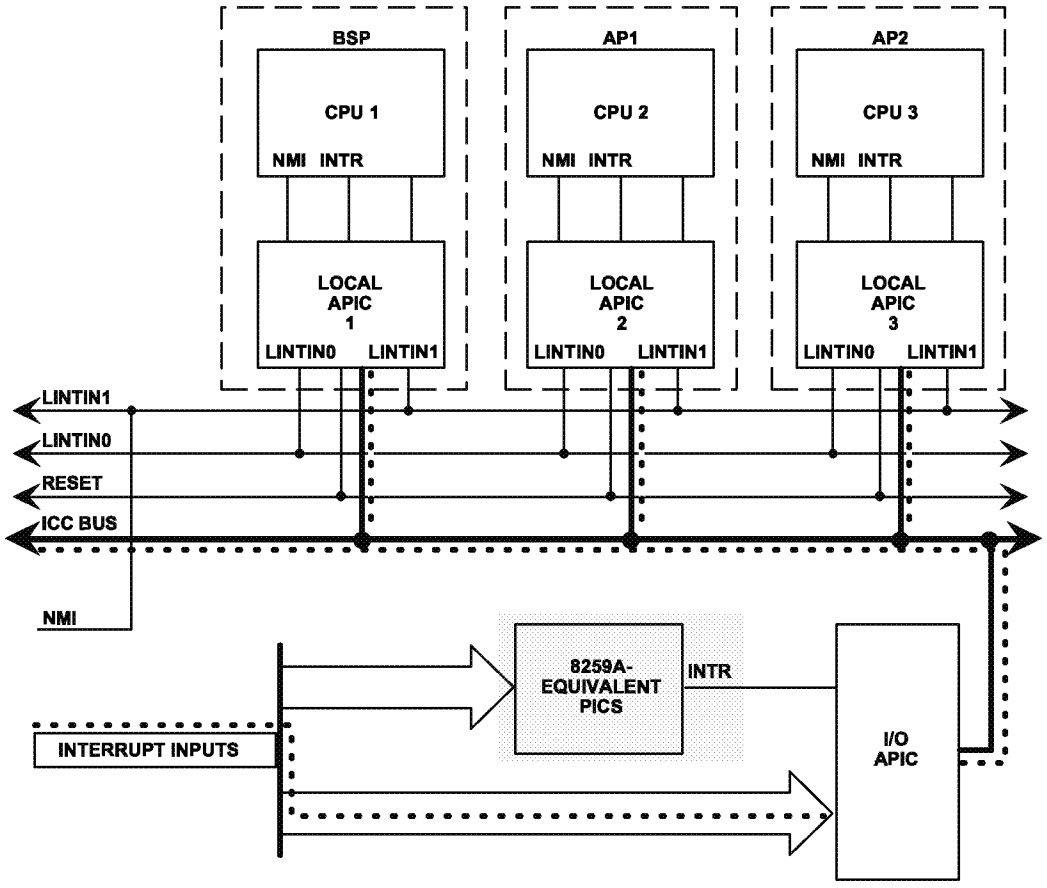


SHADED AREAS INDICATE UNUSED CIRCUITS. DOTTED LINE SHOWS INTERRUPT PATH.

Figure 3-4. Virtual Wire Mode via I/O APIC

3.6.2.3 Symmetric I/O Mode

Some MP operating systems operate in Symmetric I/O Mode. This mode requires at least one I/O APIC to operate. In this mode, I/O interrupts are generated by the I/O APIC. All 8259 interrupt lines are either masked or work together with the I/O APIC in a mixed mode. See Figure 3-5 for an overview of Symmetric I/O Mode.



SHADED AREAS INDICATE UNUSED CIRCUITS. DOTTED LINE SHOWS INTERRUPT PATH.

Figure 3-5. Symmetric I/O Mode

The APIC I/O unit has general-purpose interrupt inputs that can be individually programmed to different operating modes. The I/O APIC interrupt line assignments are system implementation specific. Refer to Chapter 4 for custom implementations and to Chapter 5 for default configurations.

The hardware must support a mode of operation in which the system can switch easily to Symmetric I/O mode from PIC or Virtual Wire mode. When the operating system is ready to switch to MP operation, it writes a 01H to the IMCR register, if that register is implemented, and enables I/O APIC Redirection Table entries. The hardware must not require any other action on the part of software to make the transition to Symmetric I/O mode.

3.6.3 Assignment of System Interrupts to the APIC Local Unit

The APIC local unit has two general-purpose interrupt inputs, which are reserved for system interrupts. These interrupt inputs can be individually programmed to different operating modes. Like the I/O APIC interrupt lines, the local APIC interrupt line assignments of a non-PC/AT-compatible system are system implementation specific. Refer to Chapter 4 for custom implementations and to Chapter 5 for default configurations.

3.6.4 Floating Point Exception Interrupt

For PC/AT compatibility, the bootstrap processor must support DOS-compatible FPU execution and exception handling while running in either of the PC/AT-compatible modes. This means that floating point error signals from the BSP must be routed to the interrupt request 13 signal, IRQ13, when the system is in PIC or virtual wire mode. While floating point error signals from an application processor need not be routed to IRQ13, platform designers may choose to connect the two. For example, connecting the floating point error signal from application processors to IRQ13 can be useful in the case of a platform that supports dynamic choice of BSP during boot.

In symmetric mode, a compliant system supports only on-chip floating point units, with error signaling via interrupt vector 16. Operating systems must use interrupt vector 16 to manage floating point exceptions when the system is in symmetric mode.

It is recommended that hardware platforms be designed to block delivery of floating point exception signals from the processors once the system has switched into symmetric mode to avoid delivery of superfluous interrupts. If done, such blocking must be implemented in a manner that is transparent to the operating system. However, the operating system must still be prepared to handle interrupts generated through assertion of floating point error signals, because on some platforms these signals may still be routed to IRQ13 even after the switch to symmetric mode.

3.6.5 APIC Memory Mapping

In a compliant system, all APICs must be implemented as memory-mapped I/O devices. APIC base addresses are at the top of the memory address space. All APIC local units are mapped to the same addresses, which are not shared. Each processor accesses its local APIC via these memory addresses. The default base address for the local APICs is 0FEE0_0000h.

Unlike the local APICs, the I/O APICs are mapped to give shared access from all processors, providing full symmetric I/O access. The default base address for the first I/O APIC is 0FEC0_0000h. Subsequent I/O APIC addresses are assigned in 4K increments. For example, the second I/O APIC is at 0FEC0_1000h.

Non-default APIC base addresses can be used if the MP configuration table is provided. (Refer to Chapter 4.) However, the local APIC base address must be aligned on a 4K boundary, and the I/O APIC base address must be aligned on a 1K boundary.

3.6.6 APIC Identification

Systems developers must assign APIC local unit IDs and ensure that all are unique. There are two acceptable ways to assign local APIC IDs, as follows:

- By hardware. The ID of each APIC local unit is sampled from the appropriate pins at RESET.
- By the BIOS. Software can override the APIC IDs assigned by hardware by writing to the Local Unit ID Register. This is possible only with help from the hardware; for example, using board ID registers that the software can read to determine which board has the BSP.

Local APIC IDs must be unique, and need not be consecutive.

The MP operating system can use the local APIC ID to determine on which processor it is executing.

The ID of each I/O APIC unit is set to zero during RESET. It is the responsibility of the operating system to verify the uniqueness of the I/O APIC ID and to assign a unique ID if a conflict is found. The assignment of APIC IDs for I/O units must always begin from the lowest number that is possible after the assignment of local APIC IDs. The operating system must not attempt to change the ID of an APIC I/O unit if the preset ID number is acceptable.

3.6.7 APIC Interval Timers

The 82489DX APIC local unit contains a 32-bit wide programmable timer with the following two independent clock input sources:

1. The CLK pin provides the clock signal that drives the 82489DX APIC's internal operation.
2. The TMBASE pin allows an independent clock signal to be connected to the 82489DX APIC for use by the timer functions.

The interval timers of the integrated APIC have only one clock input source, CLK. To maintain consistency, developers of compliant systems based on the 82489DX must choose CLK as the source of the 82489DX APIC timer clock. TMBASE must be left disabled. An MP operating system may use the IRQ8 real-time clock as a reference to determine the actual APIC timer clock speed.

Special consideration must be made for systems with stop clock (STPCLK#) capability. Timer interrupts are ignored while STPCLK# is asserted. The system time-of-day clock may need to be reset when STPCLK# is deasserted.

3.6.8 Multiple I/O APIC Configurations

Systems may provide more than one I/O APIC for increased interrupt scalability in Symmetric I/O Mode. To preserve PC/AT compatibility in PIC or Virtual Wire mode, interrupts connected to additional I/O APICs must also be connected to the 8259A programmable interrupt controller. Figure 3-6 represents one possible interrupt connection scheme for a system with two I/O APICs.

The non-ISA interrupts are connected to both the 8259A IRQ inputs and the inputs of the associated I/O APIC. To prevent non-ISA interrupts from appearing at inputs on both I/O APICs, the hardware must provide a means to disable the interrupt routing network when the system switches to symmetric I/O mode with the second I/O APIC enabled.

More information about the implementation of multiple I/O APIC configurations is presented in Appendix D.

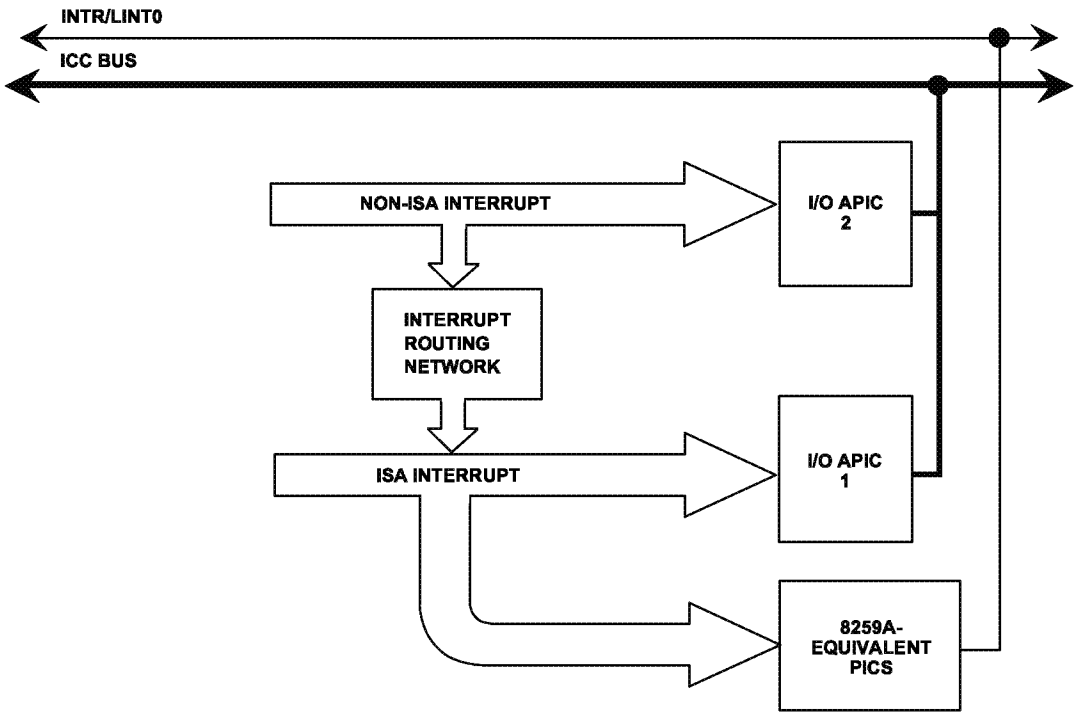


Figure 3-6. Multiple I/O APIC Configurations

3.7 RESET Support

To bring all circuitry in a computer system to an initial state, computer systems require a system-wide reset capability. To support multiple processors, a compliant system requires a processor-specific reset or initialization capability in addition to the typical system-wide reset and initialization capabilities.

- The term "RESET" refers to the system-wide hard reset. It refers to the RESET signal on both Pentium and Intel486 processors or the RESET signal of the 82489DX APIC. (See Section 3.7.1.)
- The term "INIT," refers to either a system-wide soft reset/initialization or a processor-specific initialization. For example, the term "INIT" may refer to the INIT signal on the Pentium processor or to the RESET signal on the Intel486 processor. (See Sections 3.7.2. and 3.7.3.)

Because the INIT signal is available on the Pentium processor but not on the Intel486 processor, the remainder of this document uses the above-mentioned special definitions for the terms "INIT" and "RESET":

3.7.1 System-wide RESET

The system-wide RESET, as defined by this specification, refers to a *hard* or *cold reset*, which sets all circuitry, including processor, coprocessor, add-in cards, and control logic, to an initial state. A hard reset is the reset signal that is sent to all components of the system during a power-on

or by the front panel reset button (if the system is so equipped). This type of reset operates without regard to cycle boundaries, and, for example, is connected to the RESET pin of Pentium processors.

3.7.2 System-wide INIT

The system-wide INIT, as defined by this specification, refers to a *soft* or *warm reset* that initializes only portions of the processor. This type of reset initializes the microprocessor in such a way that the reset does not corrupt any pending cycles, but waits instead for a cycle boundary, and does not invalidate the contents of caches and floating point registers. This type of reset request is connected to the INIT signal of newer processors, such as the Pentium processors. On Intel486 processors, the RESET pin is used for this function, as well as for hard resets, but the RESET pin does not provide the advantages of the INIT pin. There are many possible ways to assert a soft reset, including:

- A write either to a port of the 8042 Keyboard Controller or to some other port provided for the same purpose by a chipset.
- A shutdown special bus cycle. Usually a chipset senses a shutdown cycle and asserts a soft reset to the processor.

In a compliant system, the standard PC/AT-platform resets mentioned above, both hard and soft, must be directed to all processors in the system, except in the case of fault-tolerant MP systems, in which a soft reset may be handled on a per-processor basis.

3.7.3 Processor-specific INIT

A processor-specific INIT is one of the basic multiprocessor support functions of a compliant multiprocessor system, along with processor startup and shutdown. With it, the BSP can selectively initialize an AP for subsequent startup or recover an AP from a fatal system error. This type of INIT function is exclusively used by the MP operating system or BIOS self-test routine. The system must be designed so that the processor-specific INIT can be initiated by software programming; it is not necessary that it be initiated by hardware.

A compliant system supports the processor-specific INIT via a special interprocessor interrupt (IPI) mechanism called INIT IPI. For the 82489DX APIC, INIT IPI is an IPI that has the delivery mode RESET, which delivers the signal to all processors listed in the destination by asserting/deasserting the addressed APIC local unit's PRST output pin. When the PRST signal is connected to the INIT pin of the Pentium processor or to the RESET pin of the Intel486 processor, the INIT IPI forces the processor to begin executing at the reset vector.

For systems based on the Intel486 processor, the 82489DX APIC's PRST line must be the only line connected to the processor's RESET input, so that the INIT IPI resets the targeted processor only. The system reset signal is connected to the local 82489DX APIC's RESET input. Assertion of the system reset signal then causes all of the local 82489DX APICs to assert their PRST outputs, thereby resetting all the processors.

For integrated APIC versions of the Pentium processor, INIT IPI asserts and deasserts the internal INIT signal of the Pentium processor.

3.8 System Initial State

The system initial state is the state before the BIOS gives control to the operating system. It is identical to the system initial state of a typical PC/AT system, with the additional MP components in the following state:

1. All local APICs are disabled, except for the local APIC of the BSP if the system starts in Virtual Wire Mode.
2. All pending I/O APIC interrupts are cleared and disabled.
3. If IMCR is present, it should be set to 0 or 1 depending on the interrupt mode chosen for startup.
4. All APs are in Real Mode.
5. All APs are in HALT state or off the system bus.

The BIOS must disable interrupts to all processors and set the APICs to the system initial state before giving control to the operating system. The operating system is responsible for initializing all of the APICs.

3.9 Support for Fault-resilient Booting

OEMs may choose not to implement a fault-resilient booting capability in their systems. However, if such a capability is provided, systems developers must observe the following guidelines:

- BSP determination may be performed by special hardware or by the BIOS, but it must be totally transparent to the operating system.
- NMI and INTR must be connected to the BSP.
- FERR# and IGNNE# signals from the designated BSP must be used to support IRQ13.
- The A20M# signal from the designated BSP must be used to support the masking of physical address bit 20 on the BSP (to support DOS compatibility).

4
MP Configuration Table

The operating system must have access to some information about the multiprocessor configuration. The MP specification provides two methods for passing this information to the operating system: a minimal method for configurations that conform to one of a set of common hardware defaults, and a maximal method that provides the utmost flexibility in hardware design. Figure 4-1 shows the general layout of the MP configuration data structures.

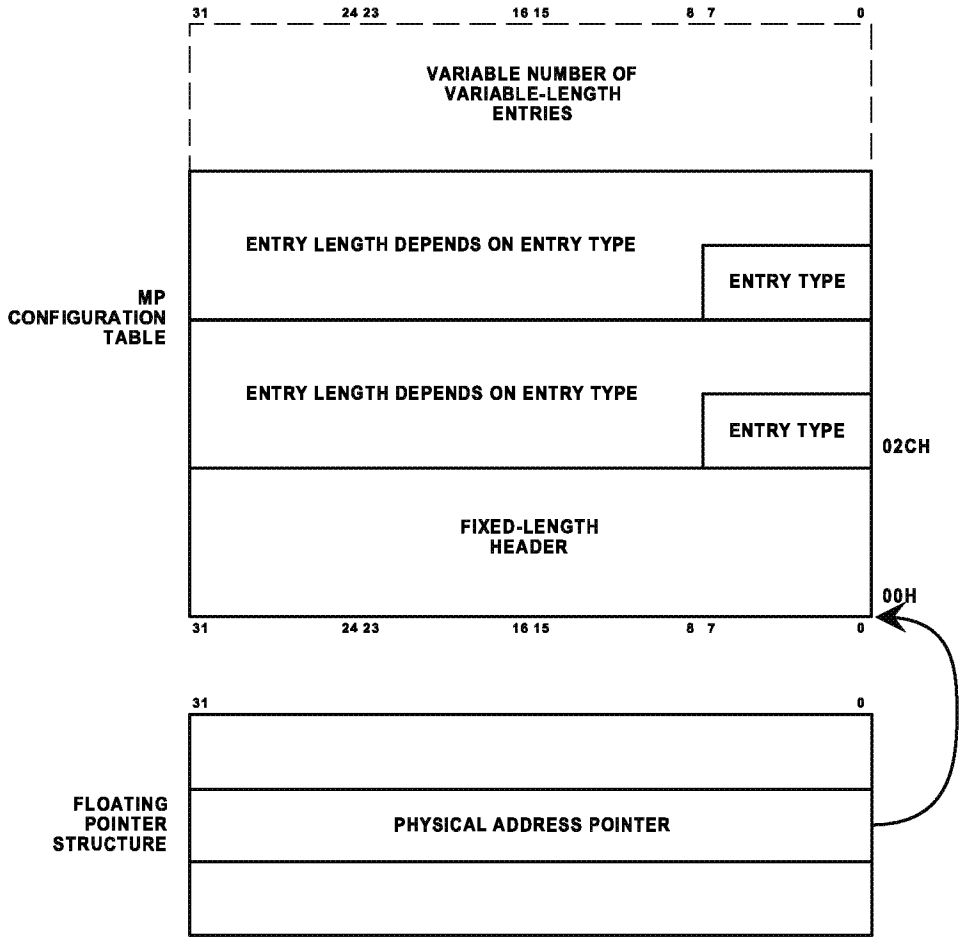


Figure 4-1. MP Configuration Data Structures

The following two data structures are used:

1. The *MP Floating Pointer Structure*. This structure contains a physical address pointer to the MP configuration table and other MP feature information bytes. When present, this structure indicates that the system conforms to the MP specification. This structure must be stored in at least one of the following memory locations, because the operating system searches for the MP floating pointer structure in the order described below:
   a. In the first kilobyte of Extended BIOS Data Area (EBDA), or
   b. Within the last kilobyte of system base memory (e.g., 639K-640K for systems with 640 KB of base memory or 511K-512K for systems with 512 KB of base memory) if the EBDA segment is undefined, or
   c. In the BIOS ROM address space between 0F0000h and 0FFFFFh.
2. The *MP Configuration Table*. This table is optional. The table is composed of a base section and an extended section. The base section contains entries that are completely backwards compatible with previous versions of this specification. The extended section contains additional entry types. The MP configuration table contains explicit configuration information about APICs, processors, buses, and interrupts. The table consists of a header, followed by a number of entries of various types. The format and length of each entry depends on its type. When present, this configuration table must be stored either in a non-reported system RAM or within the BIOS read-only memory space.

An MP configuration table is not required if the system design corresponds to one of the default configurations listed in Chapter 5. Note that these defaults are only for designs that are always equipped with two processors. Systems that support a variable number of installed processors must supply a complete MP configuration table that indicates the correct number of installed processors. For example, systems that ship with only one processor but provide for a later upgrade with a second processor may not use a default MP configuration.

The following is a list of the suggested memory spaces for the MP configuration table:

   a. In the first kilobyte of Extended BIOS Data Area (EBDA), or
   b. Within the last kilobyte of system base memory if the EBDA segment is undefined, or
   c. At the top of system physical memory, or
   d. In the BIOS read-only memory space between 0E0000h and 0FFFFFh.

The BIOS reports the base memory size in a two-byte location (40:13h) of the BIOS data area. The base memory size is reported in kilobytes minus 1K, which is used by the EBDA segment or for other purposes.

The exact starting address of the EBDA segment for EISA or MCA systems can be found in a two-byte location (40:0Eh) of the BIOS data area. If system memory is used, the BIOS must not report this memory as part of the available memory space.

These two MP configuration data structures can be located in the ROM space only if the system is not dynamically reconfigurable. The MP configuration information is intended to be read-only to the operating system.

Strings in the configuration tables are coded in ASCII. The first character of the string is stored at the lowest address of the string field. If the string is shorter than its field, the extra character locations are filled with space characters. Strings are not null terminated.

4.1 MP Floating Pointer Structure

An MP-compliant system must implement the MP floating pointer structure, which is a variable length data structure in multiples of 16 bytes. Currently, only one 16-byte data structure is defined. It must span a minimum of 16 contiguous bytes, beginning on a 16-byte boundary, and it must be located within the physical address as specified in the previous section. To determine whether the system conforms to the MP specification, the operating system must search for the MP floating pointer structure in the order specified in the previous section. Figure 4-2 shows the format of this structure, and Table 4-1 explains each of the fields. See also Appendix E, for more information.

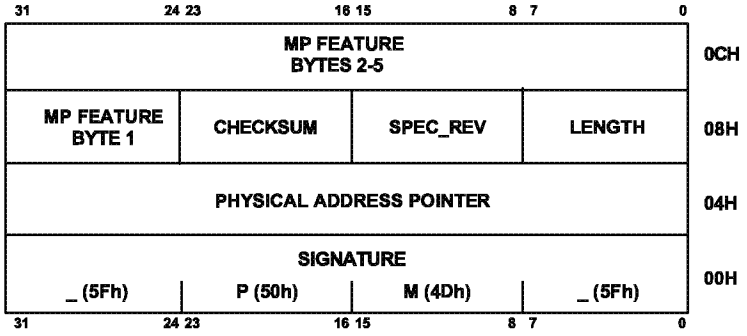


Figure 4-2. MP Floating Pointer Structure

Table 4-1. MP Floating Pointer Structure Fields

Field	Offset (in bytes:bits)	Length (in bits)	Description
SIGNATURE	0	32	The ASCII string represented by "_MP_" which serves as a search key for locating the pointer structure.
PHYSICAL ADDRESS POINTER	4	32	The address of the beginning of the MP configuration table. All zeros if the MP configuration table does not exist.
LENGTH	8	8	The length of the floating pointer structure table in paragraph (16-byte) units. The structure is 16 bytes or 1 paragraph long; so this field contains 01h.
SPEC_REV	9	8	The version number of the MP specification supported. A value of 01h indicates Version 1.1. A value of 04h indicates Version 1.4.
CHECKSUM	10	8	A checksum of the complete pointer structure. All bytes specified by the length field, including CHECKSUM and reserved bytes, must add up to zero.

Table 4-1. MP Floating Pointer Structure Fields (continued)

Field	Offset (in bytes:bits)	Length (in bits)	Description
MP FEATURE INFORMATION BYTE 1	11	8	**Bits 0-7:** MP System Configuration Type. When these bits are all zeros, the MP configuration table is present. When nonzero, the value indicates which default configuration (as defined in Chapter 5) is implemented by the system.
MP FEATURE INFORMATION BYTE 2	12:0 12:7	7 1	**Bits 0-6:** Reserved for future MP definitions. **Bit 7:** IMCRP. When the IMCR presence bit is set, the IMCR is present and PIC Mode is implemented; otherwise, Virtual Wire Mode is implemented.
MP FEATURE INFORMATION BYTES 3-5	13	24	Reserved for future MP definitions. Must be zero.

The MP feature information byte 1 specifies the MP system default configuration type. If nonzero, the system configuration conforms to one of the default configurations. The default configurations specified in Chapter 5, may only be used to describe systems that always have two processors installed.

Bit 7 of MP feature information byte 2, the IMCR present bit, is used by the operating system to determine whether PIC Mode or Virtual Wire Mode is implemented by the system.

The physical address pointer field contains the address of the beginning of the MP configuration table. If it is nonzero, the MP configuration table can be accessed at the physical address provided in the pointer structure. This field must be all zeros if the MP configuration table does not exist.

4.2 MP Configuration Table Header

Figure 4-3 shows the format of the header of the MP configuration table, and Table 4-2 explains each of the fields.

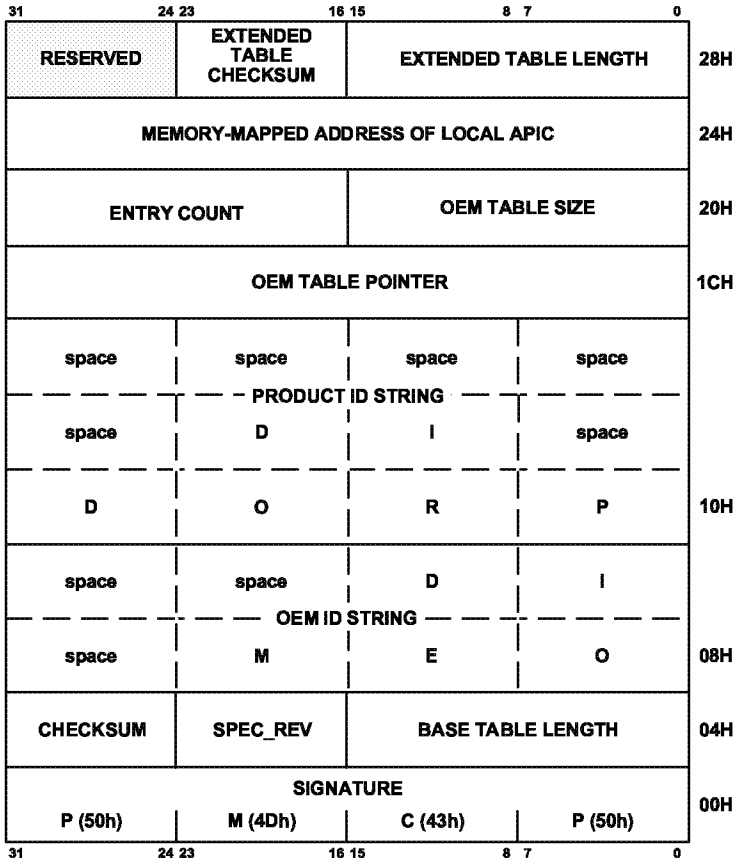


Figure 4-3. MP Configuration Table Header

Table 4-2. MP Configuration Table Header Fields

Field	Offset (in bytes)	Length (in bits)	Description
SIGNATURE	0	32	The ASCII string representation of "PCMP," which confirms the presence of the table.
BASE TABLE LENGTH	4	16	The length of the base configuration table in bytes, including the header, starting from offset 0. This field aids in calculation of the checksum.
SPEC_REV	6	8	The version number of the MP specification. A value of 01h indicates Version 1.1. A value of 04h indicates Version 1.4.
CHECKSUM	7	8	A checksum of the entire base configuration table. All bytes, including CHECKSUM and reserved bytes, must add up to zero.
OEM ID	8	64	A string that identifies the manufacturer of the system hardware.
PRODUCT ID	16	96	A string that identifies the product family.
OEM TABLE POINTER	28	32	A physical-address pointer to an OEM-defined configuration table. This table is optional. If not present, this field is zero.
OEM TABLE SIZE	32	16	The size of the base OEM table in bytes. If the table is not present, this field is zero.
ENTRY COUNT	34	16	The number of entries in the variable portion of the base table. This field helps the software identify the end of the table when stepping through the entries.
ADDRESS OF LOCAL APIC	36	32	The base address by which each processor accesses its local APIC.
EXTENDED TABLE LENGTH	40	16	Length in bytes of the extended entries that are located in memory at the end of the base configuration table. A zero value in this field indicates that no extended entries are present.
EXTENDED TABLE CHECKSUM	42	8	Checksum for the set of extended table entries, including only extended entries starting from the end of the base configuration table. When no extended table is present, this field is zero.

4.3 Base MP Configuration Table Entries

A variable number of variable length entries follow the header of the MP configuration table. The first byte of each entry identifies the entry type. Each entry type has a known, fixed length. The total length of the MP configuration table depends upon the configuration of the system. Software must step through each entry in the base table until it reaches ENTRY COUNT. The entries are sorted on ENTRY TYPE in ascending order. Table 4-3 gives the meaning of each value of ENTRY TYPE.

Table 4-3. Base MP Configuration Table Entry Types

Entry Description	Entry Type Code*	Length (in bytes)	Comments
Processor	0	20	One entry per processor.
Bus	1	8	One entry per bus.
I/O APIC	2	8	One entry per I/O APIC.
I/O Interrupt Assignment	3	8	One entry per bus interrupt source.
Local Interrupt Assignment	4	8	One entry per system interrupt source.

* All other type codes are reserved.

4.3.1 Processor Entries

Figure 4-4 shows the format of each processor entry, and Table 4-4 defines the fields.

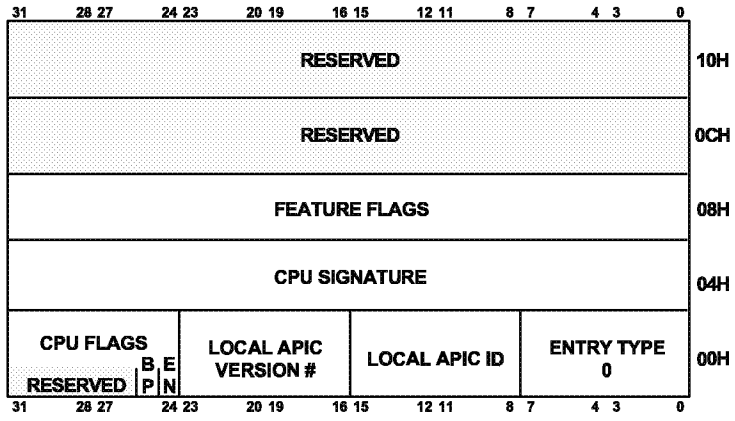


Figure 4-4. Processor Entry

In systems that use the MP configuration table, the only restriction placed on the assignment of APIC IDs is that they be unique. They do not need to be consecutive. For example, it is possible for only APIC IDs 0, 2, and 4 to be present.

Table 4-4. Processor Entry Fields

Field	Offset (in bytes:bits)	Length (in bits)	Description
ENTRY TYPE	0	8	A value of 0 identifies a processor entry.
LOCAL APIC ID	1	8	The local APIC ID number for the particular processor.
LOCAL APIC VERSION #	2	8	Bits 0–7 of the local APIC's version register.
CPU FLAGS: EN	3:0	1	If zero, this processor is unusable, and the operating system should not attempt to access this processor.
CPU FLAGS: BP	3:1	1	Set if specified processor is the bootstrap processor.
CPU SIGNATURE:			
STEPPING	4:0	4	Refer to Table 4-5 for values.
MODEL	4:4	4	Refer to Table 4-5 for values.
FAMILY	5:0	4	Refer to Table 4-5 for values.
FEATURE FLAGS	8	32	The feature definition flags for the processor as returned by the CPUID instruction. Refer to Table 4-6 for values. If the processor does not have a CPUID instruction, the BIOS must assign values to FEATURE FLAGS according to the features that it detects.

The configuration table is filled in by the BIOS after it executes a CPU identification procedure on each of the processors in the system. Whenever possible, the complete 32-bit CPU signature should be filled with the values returned by the CPUID instruction. The CPU signature includes but is not limited to, the stepping, model, and family fields. If the processor does not have a CPUID instruction, the BIOS must fill these and future reserved fields with information returned by the processor in the EDX register after a processor reset. See the *Pentium Processor User's Manual* and *Intel Processor Identification with the CPUID Instruction* (AP-485) for details on the CPUID instruction.

Table 4-5. Intel486 and Pentium Processor Signatures

Family	Model	Stepping[a]	Description
0000	0000	0000	Not a valid CPU signature.
0100	0000 and 0001	xxxx	Intel486 DX Processor
0100	0010	xxxx	Intel486 SX Processor
0100	0011	xxxx	Intel487 Processor
0100	0011	xxxx	IntelDX2™ Processor
0100	0100	xxxx	Intel486 SL Processor
0100	0101	xxxx	IntelSX2™ Processor
0100	1000	xxxx	IntelDX4™ Processor
0101	0001	xxxx	Pentium Processors (510\60, 567\66)
0101	0010	xxxx	Pentium Processors (735\90, 815\100)
	Values not shown are reserved for future processors. Refer to the documentation of each new processor for its family and model values.		
1111	1111	1111	Not a valid CPU signature. Indicates a processor that is not an Intel architecture-compatible processor (a graphics controller, for example)

[a] Intel releases information about stepping numbers as needed.

Table 4-6. Feature Flags from CPUID Instruction

Bit	Name	Description	Comments
0	FPU	On-chip Floating Point Unit	The processor contains an FPU that supports the Intel387 processor floating point instruction set.
1–6			Reserved.
7	MCE	Machine Check Exception	Exception 18 is defined for machine checks, including CR4.MCE for controlling the feature. This feature does not define the model-specific implementations of machine-check error logging reporting and processor shutdowns. Machine-check exception handlers may have to depend on processor version to do model-specific processing of the exception or test for the presence of the standard machine-check feature.
8	CX8	CMPXCHG8B Instruction	The 8 byte (64 bits) compare-and-exchange instruction is supported (implicitly locked and atomic). Introduced by the Pentium processor.
9	APIC	On-chip APIC	Indicates that an integrated APIC is present and hardware enabled. (Software disabling does not affect this bit.)
10–31			Reserved.

4.3.2 Bus Entries

Bus entries identify the kinds of buses in the system. Because there may be more than one bus in a system, each bus is assigned a unique bus ID number by the BIOS. The bus ID number is used by the operating system to associate interrupt lines with specific buses. Figure 4-5 shows the format of a bus entry, and Table 4-7 explains the fields of each entry.

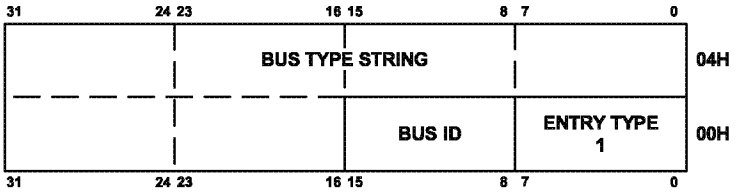


Figure 4-5. Bus Entry

Table 4-7. Bus Entry Fields

Field	Offset (in bytes)	Length (in bits)	Description
ENTRY TYPE	0	8	Entry type 1 identifies a bus entry.
BUS ID	1	8	An integer that identifies the bus entry. The BIOS assigns identifiers sequentially, starting at zero.
BUS TYPE STRING	2	48	A string that identifies the type of bus. Refer to Table 4-8 for values. These are 6-character ASCII (blank-filled) strings used by the MP specification.

Table 4-8. Bus Type String Values

Bus Type String	Description
CBUS	Corollary CBus
CBUSII	Corollary CBUS II
EISA	Extended ISA
FUTURE	IEEE FutureBus
INTERN	Internal bus
ISA	Industry Standard Architecture
MBI	Multibus I
MBII	Multibus II
MCA	Micro Channel Architecture
MPI	MPI
MPSA	MPSA
NUBUS	Apple Macintosh NuBus
PCI	Peripheral Component Interconnect
PCMCIA	PC Memory Card International Assoc.
TC	DEC TurboChannel
VL	VESA Local Bus
VME	VMEbus
XPRESS	Express System Bus

Each bus in a system must have a unique BUS ID if any one of the following criteria are true:

1. The bus does not share its memory address space with another bus.
2. The bus does not share its I/O address space with another bus.
3. The bus does not share interrupt lines with another bus.
4. Any aspect of the bus as an independent entity is software visible (such as PCI configuration space).

Special consideration must be given when assigning a BUS ID for local buses such as VL, which are designed to work in conjunction with another bus. If the bus looks like a part of another bus because it uses a subset of that bus's interrupts and address space, rendering it totally invisible to software, it does not need its own bus entry in the table. The two buses are then considered a single logical bus.

4.3.3 I/O APIC Entries

The configuration table contains one or more entries for I/O APICs. Figure 4-6 shows the format of each I/O APIC entry, and Table 4-9 explains each field.

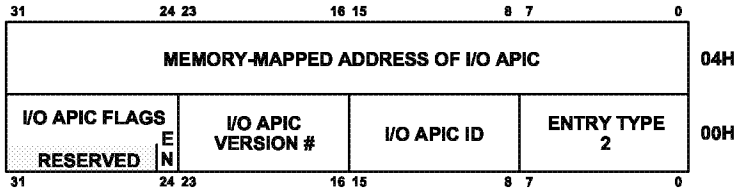


Figure 4-6. I/O APIC Entry

Table 4-9. I/O APIC Entry Fields

Field	Offset (in bytes:bits)	Length (in bits)	Description
ENTRY TYPE	0	8	A value of 2 identifies an I/O APIC entry.
I/O APIC ID	1	8	The ID of this I/O APIC.
I/O APIC VERSION #	2	8	Bits 0–7 of the I/O APIC's version register.
I/O APIC FLAGS: EN	3:0	1	If zero, this I/O APIC is unusable, and the operating system should not attempt to access this I/O APIC. At least one I/O APIC must be enabled.
I/O APIC ADDRESS	4	32	Base address for this I/O APIC.

4.3.4 I/O Interrupt Assignment Entries

These entries indicate which interrupt source is connected to each I/O APIC interrupt input. There is one entry for each I/O APIC interrupt input that is connected. Figure 4-7 shows the format of each entry, and Table 4-10 explains each field. Appendix D provides the semantics for encoding PCI interrupts.

The MP specification enables significantly more interrupt sources than the standard AT architecture by using I/O APICs. When using I/O APICs, it is preferable that the buses do not share interrupts with the other buses. Bus implementations that share interrupts, such as the PCI and VL local buses, support their bus interrupts by overloading them into another bus space. These buses can be supported in one of the following two ways:

1. Interrupt Assignment Entries for each of the bus interrupts are listed in the MP configuration table. Each interrupt destination matches the destination of another interrupt source interrupt that this interrupt shares. For example, if PCI-Device1/INTA# has the same vector as ISA-IRQ2, then both Interrupt Assignment Entries for these vectors would refer to the same destination I/O APIC and INTIN#.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re patent of Michaelis *et al.*	§	Attorney Docket No.: 185945.012600
	§	
U.S. Patent No. 7,624,302	§	Customer No.: 67395
	§	
Issue Date: November 24, 2009	§	
	§	
Filing Date: October 25, 2004	§	
	§	
For: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR	§	

DECLARATION OF ANDREW WOLFE, Ph.D, IN SUPPORT OF REQUEST FOR *EX PARTE* REEXAMINATION OF

U.S. PATENT NO. 7,624,302

Mail Stop "*Ex Parte* Reexam"
Attn: Central Reexamination Unit
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

TABLE OF CONTENTS

TABLE OF MATERIALS CONSIDERED **4**

TABLE OF MATERIALS SUBMITTED WITH DECLARATION **5**

I, Andrew Wolfe, Ph.D., hereby declare under penalty of perjury: **5**

I. INTRODUCTION **5**

A. Overview 5

B. Qualifications 5

C. Materials Considered 8

II. LEGAL PRINCIPLES **9**

A. Obviousness 9

1. General Framework 9

2. Analogous Art 10

3. Combining Known Elements in the Art, Common Sense 10

4. Teaching Away 12

5. Secondary Considerations 13

B. Claim Construction 13

III. OPINION **17**

A. Summary of My Opinions 17

B. Level of Ordinary Skill in the Art 17

C. Background Technology 17

D. Challenged Claims 18

E. Claim Construction 21

1. "boot processor" (Claims 1-27) 21

2. "system firmware means for detecting . . ." (Claim 26) 22

3. "means for determining . . ." (Claim 26) 22

4. "system firmware means for switching . . ." (Claim 26) 23

F. Ground 1 (Kalyanasundharam in view of Kennedy) Presents Substantial New Questions of Patentability 24

1. Motivation to Combine Kalyanasundharam and Kennedy 24

G. Ground 2 (Kennedy in view of Nguyen and Safford) Presents Substantial New Questions of Patentability 38

1. Motivation to Combine Kennedy, Nguyen, and Safford 38

H. Ground 4 (Safford in view of Cepulis) Presents Substantial New Questions of Patentability 62

1. Motivation to Combine Safford and Cepulis 63

I. Secondary Considerations 67

J. Declaration in Lieu of Oath 67

TABLE OF MATERIALS CONSIDERED

Exhibit	Description
Ex. 1001	U.S. Patent No. 7,624,302 to Michaelis et al. (the "'302 Patent")
Ex. 1002	U.S. File History of the '302 Patent ("U.S. File History")
Ex. 1004	U.S. Patent Application Pub. No. 2004/0123201 to Nguyen et al. ("Nguyen")
Ex. 1005	U.S. Patent No. 7,328,371 to Kalyanasundharam et al. ("Kalyanasundharam")
Ex. 1006	U.S. Patent No. 5,450,576 to Kennedy et al. ("Kennedy")
Ex. 1007	U.S. Patent No. 7,085,959 to Safford ("Safford")
Ex. 1008	U.S. Patent No. 6,625,749 to Quach ("Quach")
Ex. 1009	U.S. Patent Pub. No. 2005/0172164 to Fox ("Fox")
Ex. 1010	Wendy Bartlett, et. al., *Commercial Fault Tolerance: A Tale of Two Systems, IEEE Transactions on Dependable and Secure Computing*, Vol. 1, No. 1 (January-March 2004) ("Bartlett")
Ex. 1011	Intel, *Extensible Firmware Interface Specification*, Version 1.10 (Dec. 1, 2002) ("Intel EFI Specification")
Ex. 1012	U.S. Patent No. 7,191,328 to Hobson ("Hobson")
Ex. 1013	Bozidar Levi, *UNIX Administration, A Comprehensive Sourcebook for Effective Systems and Network Management* (2002) ("Levi")
Ex. 1014	U.S. Patent No. 5,491,788 to Cepulis et al. (Cepulis)
Ex. 1015	*Intel IA-64 Architecture Software Developer's Manual*, Revision 1.1 (July 2000) ("Intel Itanium Manual")
Ex. 1016	*Intel Multiprocessor Specification*, Version 1.4 (May 1997) ("Intel Multiprocessor Specification")
Ex. 1017	U.S. Patent No. 5,915,082 to Marshall ("Marshall")
Ex. AA	Appendix A: Claim chart comparing claims to Kalyanasundharam in view of Kennedy, as set forth in Ground 1.
Ex. BB	Appendix B: Claim chart comparing claims to Kennedy in view of Nguyen and Safford, as set forth in Ground 2.
Ex. CC	Appendix C: Claim chart comparing claims to Safford in light of Cepulis, as set forth in Ground 3.

TABLE OF MATERIALS SUBMITTED WITH DECLARATION

Exhibit	Description
Ex. A	Curriculum Vitae of Andrew Wolfe, Ph.D.

I, Andrew Wolfe, Ph.D., hereby declare under penalty of perjury:

I. INTRODUCTION

A. Overview

1. I have been retained by Unified Patents, LLC as an expert witness in the field of computer systems and computer technology including bus technology. I have been asked to analyze certain claims of U.S. Patent 7,624,302 ("the '302 patent") and provide my opinions on the patentability of certain claims of the '302 patent based on evaluating them for non-statutory obviousness-type double patenting. In addition to this Declaration, I am simultaneously providing declarations related to two other patents that are incorporated by reference in the '302 patent: U.S. Patent No. 7,502,958 (the "'958 Patent") and U.S. Patent 7,627,781 ("the '781 patent").

2. I am being compensated at my standard rate for my work performed on this matter and I am being reimbursed for my reasonable approved expenses. My compensation is not dependent on the content of my opinions. I have no financial interest in the outcome of this proceeding or pin Unified Patents, LLC. I am over the age of 18 and am competent to make this Declaration.

B. Qualifications

3. My professional qualifications, experience, publications and presentations, and a listing of previous cases in which I have provided expert testimony are set forth in my CV, attached as Exhibit A.

4. I am the founder and sole employee of Wolfe Consulting. Through Wolfe Consulting, I provide technical and business analytics to businesses on processor technology, computer

systems, consumer electronics, software, design tools, data security, cryptography and intellectual property issues. I have more than thirty-five years' experience developing products, researching, consulting, and teaching in those fields. In that time, I have worked as a computer architect, computer system designer, and as an executive in the PC and electronics business. I have also taught at some of the world's leading institutions in those fields, including Stanford University, Princeton University, Carnegie Mellon University, and Santa Clara University.

5. In 1985, I earned the B.S.E.E. degree in Electrical Engineering and Computer Science from the Johns Hopkins University. In 1987, I received the M.S. degree in Electrical and Computer Engineering from Carnegie Mellon University and in 1992, I received the Ph.D. degree in Computer Engineering from Carnegie Mellon University. My doctoral dissertation proposed a new approach for the architecture of a computer processor.

6. In 1983, I began designing touch sensors, microprocessor-based computer systems, and I/O (input/output) cards for personal computers as a senior design engineer for Touch Technology, Inc. During the course of my design projects with Technology, I designed I/O cards for PC-compatible computer systems, including the IPM PC-AT, to interface with interactive touch-based computer terminals that I designed for use in public information systems. I continued designing and developing related technology as a consultant to the Carroll Touch division of AMP, Inc., where in 1986 I designed one of the first custom touch-screen integrated circuits. I designed the touch/pen input system for the Linus WriteTop, which many believe to be the first commercial tablet computer.

7. From 1986 through 1987, I designed and built a high-performance computer system as a student at Carnegie Mellon University. From 1986 through early 1988, I also developed the curriculum, and supervised the teaching laboratory, for processor design courses.

8. In the latter part of 1989, I worked as a senior design engineer for ESL-TRW Advanced Technology Division. While at ESL-TRW, I designed and built a bus interface and memory controller for a workstation-based computer system, and also worked on the design of a multiprocessor system for real-time defensive weapons applications.

9. At the end of 1989, I (along with some partners) reacquired the rights to the technology I had developed at Touch Technology and at AMP and founded The Graphics Technology Company. Over the next seven years, as an officer and a consultant for The Graphics

Technology Company, I managed the company's engineering development activities and personally developed dozens of touch screen sensors, controllers, and interactive touch-based computer systems.

10. I have consulted, formally and informally, for a number of fabless semiconductor companies, particularly in the area of processor development. In particular, I have served on the technology advisory boards for two processor design companies: BOPS, Inc., where I chaired the board, and Siroyan Ltd., where I served in a similar role for three networking chip companies – Intellon, Inc., Comsilica, Inc., and Entridia, Inc. – and one 3D game accelerator company, Ageria, Inc.

11. I have also served as a technology advisor to Motorola and to several venture capital funds in the U.S. and Europe. Currently, I am a director at Turtle Beach Corporation (NASDAQ: HEAR), providing guidance in its development of premium audio peripheral devices for a variety of commercial electronic products.

12. From 1991 through 1997, I served on the Faculty of Princeton University as an Assistant Professor of Electrical Engineering. At Princeton, I taught undergraduate and graduate-level courses in Computer Architecture, Advanced Computer Architecture, Display Technology, and Microprocessor Systems, and conducted sponsored research in the area of computer systems and related topics. From 1999 through 2002, I taught a Computer Architecture course to both undergraduate and graduate students at Stanford University multiple times as a Consulting Professor. At Princeton, I received several teaching awards, both from students and from the School of Engineering. I have also taught advanced microprocessor architecture to industry professionals in IEEE and ACM sponsored seminars. I am currently a lecturer at Santa Clara University teaching courses on Microprocessor Systems, Advanced Logic Design, Real-Time Computing, and Mechatronics.

13. From 1997 through 2002, I held a variety of executive positions at a publicly held fabless semiconductor company originally called S3, Inc. and later called Sonicblue Inc. I held the positions of Chief Technology Officer, Vice President of System Integration Products, Senior Vice President of Business Development, and Director of Technology, among others.

14. In the roles listed above, I worked with and managed architecture and design teams responsible for the development of general-purpose and special-purpose processors.

15. My teams also developed the Rio MP3 players and a music delivery platform and webstore backend service for selling music. In 1999, this music delivery system was spun out as a separate company called RioPort.com. I served on the RioPort.com board of directors and became involved in their product and technology strategy. I also managed engineering and marketing for the Rio product line for a period of time as an interim general manager. During my time at SonicBlue we launched more than 30 new consumer electronics products.

16. I served as a board member and technical advisor as KBGear Inc. from 1999-2001. KBGear Inc. designed and produced digital cameras and music players.

17. I have published more than fifty peer-reviewed papers in computer architecture and computer systems and IC design. I have also chaired IEEE and ACM conferences in microarchitecture and integrated circuit design and served as an associate editor for IEEE and ACM journals. I served on the IEEE Computer Society Awards committee. I am a Fellow of the IEEE and a Distinguished Contributor to the IEEE Computer Society. I am a named inventor on at least fifty-seven U.S. patents and thirty-seven foreign patents.

18. I have been the invited keynote speaker at the ACM/IEEE International Symposium on Microarchitecture and at the International Conference on Multimedia. I have also been an invited speaker on various aspects of technology or the PC industry at numerous industry events including the Intel Developer's Forum, Microsoft Windows Hardware Engineering Conference, Microprocessor Forum, Embedded Systems Conference, Comdex, and Consumer Electronics Show as well as at the Harvard Business School and the University Illinois Law School. I have been interviewed on subjects related to technology and the electronics industry by publications such as the Wall Street Journal, New York Times, LA Times, Time, Newsweek, Forbes, and Fortune as well as CNN, NPR, and the BBC. I have also spoken at dozens of universities including MIT, Stanford, University of Texas, Carnegie Mellon, UCLA, University of Michigan, Rice University, and Duke University.

C. Materials Considered

19. In addition to relying on my extensive experience in the field of computer systems (summarized above), I have considered certain materials in formulating my opinions.

Those materials include the exhibits listed in the Table of Materials Considered and Table of Materials Submitted with Declaration above.

II. LEGAL PRINCIPLES

A. Obviousness

20. I am a technical expert and do not offer any legal opinions. However, I have been informed about certain legal principles regarding patentability and related matters under United States patent law, which I have applied in performing my analysis and arriving at my technical opinions in this matter.

21. I understand that for *ex parte* reexamination, invalidity must be shown under a preponderance of the evidence standard, and this is the standard I have used throughout my declaration. Further, I understand that each patent claim is considered separately for purposes of unpatentability.

1. General Framework

22. I am informed that a person cannot obtain a patent on an invention if the differences between the invention and the prior art are such that the subject matter as a whole would have been obvious to a person having ordinary skill in the art obvious at the time of the invention. I am informed that a conclusion of obviousness may be founded upon more than a single item of prior art. I am also informed that obviousness is determined by evaluating the following factors: (1) the scope and content of the prior art, (2) the differences between the prior art and the claim at issue, (3) the level of ordinary skill in the pertinent art, and (4) secondary considerations of non- obviousness. In addition, the obviousness inquiry should not be done in hindsight. Instead, the obviousness inquiry should be done through the eyes of a person having ordinary skill in the art at the time of the alleged invention (often referred to as a "PHOSITA," "POSITA," or "POSITA").

23. In considering whether certain prior art renders a particular patent claim obvious, I am informed that one must consider the scope and content of the prior art, examples of which include disclosures in patents, trade publications, journal articles, conference papers, industry standards, product literature and documentation, texts describing competitive technologies, requests for comment published by standard setting organizations, and materials from industry conferences.

24. I am informed that for a patent claim to be found obvious, the proper perspective to consider is that of a person having ordinary skill in the art at the time of the alleged invention. I am informed that factors to consider in determining the level of ordinary skill in the art include (1) the educational level and experience of people working in the field at the time the invention was made, (2) the types of problems faced in the art and the solutions found to those problems, and (3) the sophistication of the technology in the field.

2. Analogous Art

25. I am informed that for a prior art reference to be proper for use in an obviousness analysis, the reference must be "analogous art" to the claimed invention. I am informed that a reference is analogous art to the claimed invention if: (1) the reference is from the same field of endeavor as the claimed invention (even if it addresses a different problem); or (2) the reference is reasonably pertinent to the problem faced by the inventor (even if it is not in the same field of endeavor as the claimed invention). In order for a reference to be "reasonably pertinent" to the problem, it must logically have commended itself to an inventor's attention in considering his problem. In determining whether a reference is reasonably pertinent, one should consider the problem faced by the inventor, as reflected either explicitly or implicitly, in the specification. I believe that all of the references that I considered in forming my opinions in this reexamination request are well within the range of references a person having ordinary skill in the art would have consulted by the time of the invention to address the type of problems described in the Challenged Claims.

3. Combining Known Elements in the Art, Common Sense

26. I am informed that, in order to establish that a claimed invention was obvious based on a combination of prior art elements, a clear articulation of the reason(s) why a claimed invention would have been obvious must be provided. Specifically, I am informed that, under the U.S. Supreme Court's **KSR** decision, a combination of multiple items of prior art can render a patent claim obvious when there was an apparent reason for a person having ordinary skill at the time of the invention to combine or modify the prior art. I am informed that such reason may include, but is not limited to, one or more of the following rationales: (A) combining prior art methods according to known methods to

yield predictable results; (B) substituting one known element for another to obtain predictable results; (C) using a known technique to improve a similar device in the same way; (D) applying a known technique to a known device ready for improvement to yield predictable results; (E) trying a finite number of identified, predictable potential solutions, with a reasonable expectation of success; (F) identifying that known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces if the variations are predictable to one of ordinary skill in the art; or (G) identifying an explicit teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior art reference or to combine the prior art references to arrive at the claimed invention.

27. I am informed that the test for obviousness is not whether the features of a secondary reference may be bodily incorporated into the structure of the primary reference, but rather whether one of skill would have been motivated to combine the teachings of the prior art references to achieve the claimed invention. Nevertheless, I am informed that an invention that is a combination of prior art must do more than yield predictable results to be non-obvious and that, even where one or more motivations to combine exist, claims are not necessarily obvious unless a person having ordinary skill in the art would have had a reasonable expectation of success in implementing or applying the proposed combination at the time of the invention.

28. I also am informed that in conducting an obviousness analysis, a precise teaching directed to the specific subject matter of the challenged claim need not be sought out because it is appropriate to take account of the inferences and creative steps that a person having ordinary skill in the art at the time of the invention would employ. In other words, the prior art need not be directed towards solving the same specific problem as the problem addressed by the patent. Further, the individual prior art references themselves need not all be directed towards solving the same problem. The prior art considered can be directed to any need or problem known in the field of endeavor at the time of invention and can provide a reason for combining the elements of the prior art in the manner claimed. Common sense teaches that familiar items may have obvious uses beyond their primary purposes, and a person having ordinary skill

in the art will often have been able to fit the teachings of multiple prior art references like pieces of a puzzle. I am informed that, under the KSR standard, common sense is important and should be considered in assessing obviousness.

29. I also am informed that the fact that a particular combination of prior art elements was "obvious to try" may indicate that the combination was obvious even if no one attempted the combination. If the combination was obvious to try (regardless of whether it was actually tried) or leads to anticipated success, then it is likely the result of ordinary skill and common sense rather than innovation. I am further informed that in many fields it may be that there is little discussion of obvious techniques or combinations, and it often may be the case that market demand, rather than scientific literature or knowledge, will drive the design of an innovation. When there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person having ordinary skill in the art has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation, but of ordinary skill and common sense.

4. Teaching Away

30. I am informed that it is improper to combine references where the references teach away from their combination. I am informed that a reference may be said to teach away when a person of skill, upon reading the reference, would be discouraged from following the path set out in the reference, or would be led in a direction divergent from the path that was taken by the patent applicant. In general, a reference will teach away if it suggests that the line of development flowing from the reference's disclosure is unlikely to be productive of the result sought by the patentee. I am informed that a reference teaches away, for example, if (1) the combination would produce a seemingly inoperative device, or (2) the references leave the impression that the product would not have the property sought by the patentee. However, I am informed that a reference does not teach away if it merely expresses a general preference for an alternative invention but does not criticize, discredit, or otherwise discourage investigation into the invention claimed. Further, a person of skill will often appreciate that different courses of action

often have simultaneous advantages and disadvantages, and I am informed that the existence of such does not necessarily obviate a motivation to combine.

5. Secondary Considerations

31. I am informed that even if a prima facie case of obviousness is established, the final determination of obviousness must also consider "secondary considerations" if presented. I am informed that in most instances, the patentee raises these secondary considerations of non-obviousness. In that context, the patentee argues an invention would not have been obvious in View of these considerations, which include: (a) commercial success of a product due to the merits of the claimed invention; (b) a long-felt, but unsatisfied need for the invention; (c) failure of others to find the solution provided by the claimed invention; (d) deliberate copying of the invention by others; (e) unexpected results achieved by the invention; (f) praise of the invention by others skilled in the art; (g) lack of independent simultaneous invention within a comparatively short space of time; (h) teaching away from the invention in the prior art.

32. I am further informed that secondary-considerations evidence is only relevant if the offering party establishes a connection, or nexus, between the evidence and the claimed invention. The nexus cannot be based on prior art features. For example, if the commercial success of products allegedly embodying the claims is presented as a secondary indicium of non-obviousness, the patent owner must show that such success is due to features of the claimed invention and not features unrelated to the claimed invention or that already existed in the prior art. The establishment of a nexus is a question of fact. I have considered the secondary considerations that I am aware of at this time and determined that they do not impact my opinions.

B. Claim Construction

33. I am informed that in performing an obviousness analysis, it is necessary to understand the scope of the claims. I am also informed the first step in an unpatentability analysis, therefore, often involves construing the claims. Second, the construed claim language is then compared to the disclosures of the prior art.

34. I have also been informed that the claims must be given their broadest reasonable interpretation consistent with the patent's specification. It is my understanding that

what is to be considered includes the claims, the patent specifications and drawings, and the prosecution history, including any art listed by the examiner or the applicant. I understand that the broadest reasonable interpretation may differ from the Phillips standard, where claim terms are given their ordinary and custom meaning.

35. Counsel has informed me that the specification of the '302 Patent provides definitions for certain terms and phrases used in the claims. Therefore, I have applied these definitions here.

36. I have been asked to evaluate novelty and obviousness as these legal concepts apply against certain claims of the '302 patent. I am not a lawyer, and therefore, I have been given certain instructions regarding the law of anticipation and obviousness and have been asked to apply those principles in conducting my analysis. I have done so.

37. I understand that to determine whether a claimed invention is patentably distinct, I must consider whether there are reasons why a person of ordinary skill in the art would have concluded that the invention defined in the claim at issue would have been an obvious variation of the invention defined in a reference claim. I also understand that a claimed apparatus or device is not patentably distinct over a reference claim directed towards a method of using that apparatus or device.

38. I understand that even though the specification of the reference patent is not technically considered to be prior art, it may still be used to interpret the claims including the reference claim. I understand that the specification of the reference patent can be used as a dictionary to learn the meaning of a term in the reference claim. I also understand that to determine if the claim at issue merely defines an obvious variation of what was earlier disclosed and claimed in the reference patent, portions of the specification of the reference patent that provide support for the reference claim may also be used to determine whether what is later claimed in the claim at issue is an obvious variation of the reference claim.

39. I also understand that because the question of whether the claim at issue is patentably distinct implicates the question of obviousness, it is also proper to consider what has been disclosed by the prior art and what is within the knowledge and skill of the person of ordinary skill in the art.

40. I also understand that a claim is invalid under 35 U.S.C. § 103(a) (pre-AIA) if the subject matter described by the claim as a whole would have been obvious to a hypothetical person

of ordinary skill in the art in view of a prior art reference or in view of a combination of references at the time the claimed invention was made. Therefore, I understand that obviousness is determined from the perspective of a hypothetical person of ordinary skill in the art ("POSITA") and that the asserted claims of the patent should be read from the point of view of such a person at the time the claimed invention was made. I further understand that a hypothetical POSITA is assumed to know and to have all relevant prior art in the relevant field of endeavor.

41. I understand that an analysis of whether a claimed invention would have been obvious should be considered in light of the scope and content of the prior art, the differences (if any) between the prior art and the claimed invention, and the level of ordinary skill in the pertinent art involved. I understand as well that a prior art reference should be viewed as a whole. These are two of the factual inquiries in the obviousness analysis. I understand that relevant factual inquiries also include the level of ordinary skill in the pertinent art and objective indicia of non-obviousness.

42. I understand that in considering whether an invention for a claimed combination would have been obvious, I may assess whether there are apparent reasons to combine known elements in the prior art in the manner claimed in view of interrelated teachings of multiple prior art references, the effects of demands known to the design community or present in the marketplace, and/or the background knowledge possessed by a person having ordinary skill in the art. I understand that other principles may be relied on in evaluating whether a claimed invention would have been obvious, and that these principles include the following:

- A combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results;
- When a device or technology is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or in a different one, so that if a person of ordinary skill can implement a predictable variation, the variation is likely obvious;
- If a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill;

- An explicit or implicit teaching, suggestion, or motivation to combine two prior art references to form the claimed combination may demonstrate obviousness, but proof of obviousness does not depend on or require showing a teaching, suggestion, or motivation to combine;
- Market demand, rather than scientific literature, can drive design trends and may show obviousness;
- In determining whether the subject matter of a patent claim would have been obvious, neither the particular motivation nor the avowed purpose of the named inventor controls;
- One of the ways in which a patent's subject can be proved obvious is by noting that there existed at the time of invention a known problem for which there was an obvious solution encompassed by the patent's claims;
- Any need or problem known in the field of endeavor at the time of invention and addressed by the patent can provide a reason for combining the elements in the manner claimed;
- "Common sense" teaches that familiar items may have obvious uses beyond their primary purposes, and in many cases a person of ordinary skill will be able to fit the teachings of multiple patents together like pieces of a puzzle;
- A person of ordinary skill in the art is also a person of ordinary creativity, and is not an automaton;
- A patent claim can be proved obvious by showing that the claimed combination of elements was "obvious to try," particularly when there was a design need or market pressure to solve a problem and there were a finite number of identified, predictable solutions such that a person of ordinary skill in the art would have had good reason to pursue the known options within his or her technical grasp; and
- One should be cautious of using hindsight in evaluating whether a claimed invention would have been obvious.

43. I further understand that, in making a determination as to whether or not the claimed invention would have been obvious to a person of ordinary skill, certain objective factors if they are present should be considered, such as: commercial success of products practicing the claimed invention; long-felt but unsolved need; teaching away; failure of

others; unexpected results; copying; and praise by others in the field. These factors are generally referred to as "secondary considerations" or "objective indicia" of non-obviousness. I understand, however, that for such objective evidence to be relevant to the obviousness of a claim, there must be a causal relationship (called a "nexus") between the claim and the evidence, and that this nexus must be based on a novel element of the claim rather than something in the prior art. I also understand that even when they are present, secondary considerations may be unable to overcome primary evidence of obviousness (e.g., motivation to combine with predictable results) that is sufficiently strong.

III. OPINION

A. Summary of My Opinions

44. As shown below, it is my understanding that Requester submits that the prior art references raise substantial new questions of patentability because "the teaching of the (prior art) patents and printed publications is such that a reasonable examiner would consider the teaching to be important in deciding whether or not the claim is patentable." *See* Manual of Patent Examining Procedure (M.P.E.P.) § 2242. For at least the reasons discussed below, I agree with Requester.

B. Level of Ordinary Skill in the Art

45. I agree that a Person of Ordinary Skill in The Art ("POSITA") would have had, as of October 25, 2004: (1) at least a Bachelor's Degree in electrical engineering, electronics engineering, computer engineering, or a related scientific field; and (2) at least one to two years of experience in the computer science field, including chip design and software engineering, with additional education substituting for less experience and vice versa.

C. Background Technology

46. Claims 1-27 recite a "boot processor," which should be given the broadest reasonable interpretation that is both "consistent with the ordinary and customary meaning of the term" and "consistent with the use of the term in the specification and drawings." *See* MPEP § 2111. In my opinion, a boot processor is any processor designated for handling bootstrap functions. In my experience, a boot processor can also be referred as a bootstrap processor (BSP) or by a computer-readable designator (e.g., logical CPU 0). I agree that a boot processor is likely a part of any multiprocessor computer system.

D. Challenged Claims

Claims 1-27 are challenged:

1. A method comprising:

detecting loss of lockstep (LOL) for a processor in a multi-processor system;

determining that the processor for which said LOL is detected is assigned a role of boot processor;

changing a system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected, to a different value, which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor; and

changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to a spare processor without shutting down the system's operating system.

2. The method of claim 1 wherein the spare processor is a hot spare that is held in idle until switched the role of boot processor.

3. The method of claim 1 wherein the spare processor is an active spare.

4. The method of claim 3 wherein said switching further comprises:

causing the system's operating system to idle the active spare.

5. The method of claim 1 wherein the spare processor is a processor module that includes a master and slave processors that operate in lockstep.

6. The method of claim 1 wherein said determining that the processor for which said LOL is detected is assigned the role of boot processor comprises:

accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor.

7. The method of claim 1 further comprising: determining the processor that is assigned the role of spare processor.

8. The method of claim 7 wherein said determining the processor that is assigned the role of spare processor comprises:

accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of spare processor.

9. The method of claim 1 wherein said determining step is performed by system firmware.

10. The method of claim 1 wherein said switching step is performed by system firmware.

11. The method of claim 1 wherein the processor for which said LOL is detected includes a lockstep pair of processors, and wherein said switching comprises:
determining one of the lockstep pair of processors that is not the cause of said LOL; and
copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to said spare processor.

12. The method of claim 11 wherein said spare processor includes a lockstep pair of processors, and wherein said copying comprises:
copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to each of said lockstep pair of processors of said spare processor.

13. A method comprising:
system firmware determining that loss of lockstep (LOL) is detected for a lockstep pair of processors that are assigned the role of boot processor in a system;
determining one of said lockstep pair of processors that is not the cause of said LOL;
copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to a spare processor;
changing a system identifier, used by the system's operating system to recognize the boot processor, of said lockstep pair of processors that are assigned the role of boot processor, to a different value, which causes the s system's operating system to not recognize the lock step pair of processors for which said LOL is detected as the processor; and
changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to said spare processor.

14. The method of claim 13 wherein the spare processor is a hot spare that is held in idle until switched the role of boot processor.

15. The method of claim 13 wherein the spare processor is an active spare.

16. The method of claim 15 further comprising: causing the system's operating system to idle the active spare.

17. The method of claim 13 wherein said switching causes the system's operating system to recognize said spare processor as said boot processor without shutting down said operating system.

18. The method of claim 13 wherein said determining step is performed by system firmware.

19. The method of claim 13 wherein said copying step is performed by system firmware.

20. The method of claim 13 wherein said switching step is performed by system firmware.

21. A system comprising:

a plurality of processor modules that each include a master processor and a slave processor that operate in lockstep;

an operating system;

error detection logic operable to detect loss of lockstep (LOL) for at least one of said processor modules; and

system firmware operable, responsive to the detection of LOL for a first of said processor modules, to determine whether said first processor module is assigned a role of system boot processor;

wherein if determined that said first processor module is assigned the role of system boot processor, said system firmware is further operable to cause said operating system to recognize another processor module as system boot processor without shutting down said operating system.

22. The system of claim 21 wherein the another processor module is a processor module assigned a role of spare processor.

23. The system of claim 21 wherein the another processor module is held in idle until said system firmware causes said operating system to recognize said another processor module as said system boot processor.

24. The system of claim 21 wherein the system firmware causes the operating system to idle the another processor module before causing said operating system to recognize said another processor module as said system boot processor.

25. The system of claim 21 further comprising:

non-volatile data storage storing information that assigns the role of system boot processor to one of said plurality of processor modules.

26. A system comprising:

system firmware means for detecting loss of lockstep (LOL) for a processor in a multi-processor system;

means for determining whether the processor for which said LOL is detected is assigned the role of boot processor; and

system firmware means for switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which said LOL is detected is assigned the role of boot processor.

27. The system of claim 26 wherein said processor in said multi-processor system is an Itanium Processor Family (IPF) processor.

E. Claim Construction

47. The claims of the '302 Patent include limitations that invoke 35 U.S.C. § 112 par. 6 pre-AIA or 35 U.S.C. § 112(f) ("112(f)") post-AIA. 35 U.S.C. § 112(f) § (limitations "may be expressed as a means . . . for performing a specified function without the recital of structure, material, or acts in support thereof, and such [limitation] shall be construed to cover the corresponding structure, material, or acts described in the specification and equivalents thereof.").

48. I have applied the plain and ordinary meaning of all other claim terms in the Challenged Claims not discussed below.

1. "boot processor" (Claims 1-27)

49. I agree that a boot processor is a critical device for any computing system and is likely included in any multiprocessor computing system. I agree that, in multi-processor (MP) system architectures, a boot processor may either be selected by default or through a selection process during bootup. I agree that, when all processors in an MP-compliant system are functionally identical, one of the processors will ordinarily be designated as the bootstrap processor (BSP) at system initialization by the system hardware or by the system hardware in conjunction with the basic input/output system (BIOS).

50. I agree that, in some MP system architectures, a boot processor may be referred to by a logical, rather than a physical, position (e.g., logical CPU 0 or CPU L0 rather than physical CPU 0 or CPU P0) that is indicative of its designation for controlling other system hardware since the role of boot processor may often be assigned to a processor regardless of its physical position within a system. I agree that this interpretation of boot processor is consistent with the context provided by the '781 Patent.

51. In my opinion, the broadest reasonable interpretation for boot processor that is both "consistent with the ordinary and customary meaning of the term" and "consistent with the use of the term in the specification and drawings" is any processor designated for handling

bootstrap functions. To the extent that the PTO does not agree with this, my opinions may also be applicable under similar interpretations.

52. I have applied the plain and ordinary meaning of all other claim terms in the Challenged Claims.

53. To the extent that the PTO does not agree with this construction, my opinions may also be applicable under similar constructions.

2. "system firmware means for detecting . . ." (Claim 26)

54. Claim 26 of the '302 Patent recites "*system firmware means for detecting loss of lockstep (LOL) for a processor in a multi-processor system;…*" In my opinion, because the limitation recites a modifier ("system firmware") before "means," the limitation does not require interpretation under 112(f). A POSITA would understand the structure of this claim element to be system firmware as recited in the claim.

55. I agree that, if the "system firmware means" requires interpretation under 112(f) and would include equivalents, the '302 Patent describes structure for the system firmware means in its specification. The '302 Patent states:

> Processor architecture generally comprises corresponding **supporting firmware**, such as firmware 18 of system 10. For example, as described further below in conjunction with the specific example of FIG. 2, the IPF processor architecture comprises such Supporting firmware as Processor Abstraction Layer (PAL), System Abstraction Layer (SAL), and Extended Firmware Interface (EFI). **Such supporting firmware may enable**, for example, **the OS to access a particular function implemented for the processor**… Accordingly, the Supporting firmware. Such as the PAL, is implemented to provide an **interface to the processor(s)** for accessing the functionality provided by such processor(s). Each of those interfaces provide standard, published procedure calls that are Supported.
>
> *Ex. 1001* at 8:34-59.

56. Accordingly, if the PTO determines that this is a means-plus-function limitation, then in my opinion, the claimed function is "detecting loss of lockstep (LOL) for a processor in a multi-processor system" and the corresponding disclosed structure for performing the claimed function is supporting firmware, including a Processor Abstraction Layer, a System Abstraction Layer, and an Extended Firmware Interface, and equivalents thereof.

3. "means for determining . . ." (Claim 26)

57. Claim 26 of the '302 Patent recites a "*means for determining whether the processor for which said LOL is detected is assigned the role of boot processor.*" In my opinion, a POSITA would consider the recitation of "means" in this claim limitation to be a nonce word and would be unable to reasonably ascertain the claimed structure without referring to the structure disclosed in the specification.

58. The '302 Patent disclosed that "[a]s described further herein, in certain embodiments, **a device tree** is stored to a data storage device (e.g., SRAM) such that it is accessible to the firmware 18, wherein the **device tree includes information identifying processor 12 as the system's boot processor** and identifying processor 15 as the spare." Ex. 1001 at 9:4-8. The '302 Patent further indicates that "**device tree 203 includes information designating which processor is the boot processor**, as well as a spare processor for the boot processor." *Id.* at 13:21-23. Accordingly, it is my opinion that the claimed function is the claimed function of determining whether the processor for which said LOL is detected is assigned the role of boot processor. The corresponding disclosed structure for performing this function is a device tree, and firmware or hardware for accessing the processor designators in the device tree, and equivalents thereof.

4. "system firmware means for switching . . ." (Claim 26)

59. Claim 26 of the '302 Patent recites "system firmware means for switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which said LOL is detected is assigned the role of boot processor." In my opinion, because the limitation recites a modifier ("system firmware") before "means," the limitation does not require interpretation under 112(f). A POSITA would understand the structure of this claim element to be system firmware as recited in the claim.

60. I agree that, if the "system firmware means" requires interpretation under 112(f) and would include equivalents, the '302 Patent describes structure for the system firmware means in its specification. The '302 Patent states:

> Processor architecture generally comprises corresponding **supporting firmware**, such as firmware 18 of system 10. For example, as described further below in conjunction with the specific example of FIG. 2, the IPF processor architecture comprises such Supporting firmware as Processor Abstraction Layer (PAL), System Abstraction Layer (SAL), and Extended Firmware Interface (EFI). **Such**

> **supporting firmware may enable**, for example, **the OS to access a particular function implemented for the processor**... Accordingly, the Supporting firmware. Such as the PAL, is implemented to provide an **interface to the processor(s)** for accessing the functionality provided by such processor(s). Each of those interfaces provide standard, published procedure calls that are supported.

Id. at 8:34-59.

61. Accordingly, if the PTO determines that this is a means-plus-function limitation, then in my opinion the claimed function is "switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which said LOL is detected is assigned the role of boot processor." The corresponding disclosed structure for performing the claimed function is supporting firmware, including a Processor Abstraction Layer, a System Abstraction Layer, and an Extended Firmware Interface, or equivalents thereof.

F. Ground 1 (Kalyanasundharam in view of Kennedy) Presents Substantial New Questions of Patentability

62. I agree with the analysis in the Request that Kalyanasundharam in view of Kennedy renders obvious each and every element of claims 1, 5-8, 13, 17, 21, 22, and 25-26. For ease of reference, I identify evidence supporting my opinions regarding obviousness below for 1, 5-8, 13, 17, 21, 22, and 25-26 below. I have also reviewed and assisted in the preparation of Appendices A-C. Appendix A provides a detailed mapping of Kalyanasundharam in view of Kennedy to claims 1, 5-8, 13, 17, 21, 22, and 25-26 that accurately reflects my opinions regarding obviousness.

1. Motivation to Combine Kalyanasundharam and Kennedy

63. In my opinion, and as explained in detail in Appendix B, the subject matter of the claims for which reexamination is requested would have been obvious based on the combination of Kalyanasundharam in view of Kennedy. Kalyanasundharam generally describes a process of recovering from an error by disabling a failed primary processor core and selecting an auxiliary processor core to function as the new primary processor core. *Kalyanasundharam* (Ex. 1005) at 13:16-23. Kalyanasundharam further expressly indicates that the error can be a mismatch between processor outputs or loss of lockstep between processor cores. *Id.* at 10:26-42. While Kalyanasundharam describes the process of failing over from a primary processor core to an auxiliary processor core in response to such a

mismatch, it fails to explicitly mention a "boot processor" as recited in the '302 Patent. Notably, Kalyanasundharam does not limit its "primary" or "secondary" processor cores to any specifically defined function or role. That is, the disclosure of Kalyanasundharam, while not explicitly disclosing a "boot processor," does not teach or suggest that its processor cores cannot be "boot processors." Moreover, a POSITA would understand that Kalyanasundharam includes a boot processor as a role for one or more processors.

64. In my opinion, Kennedy, in an analogous reference, explicitly describes the process of failing over from a failed boot processor to an alternative processor that takes over the functions of the failed boot processor. *Kennedy* (Ex.1006) at 13:38-44. A POSITA would have been motivated to combine Kennedy with Kalyanasundharam to provide the added reliability of boot processor failover to Kalyanasundharam's described methods of processor failover. Specifically, with regard to Kalyanasundharam's system, which fails over from a primary processor core to a secondary processor core in response to a loss of lockstep, a POSITA would have been motivated to add the boot processor and spare processor described by Kennedy to extend its failover functionality to a broader number of processor types. This would provide loss of lockstep failover not just between primary and secondary processor cores, but also between boot and spare processors. Because the Kalyanasundharam and Kennedy systems both describe a similar computing architecture in which processors fail over to backup processors upon the occurrence of a failure, the POSITA would reasonably expect to succeed when combining these teachings. This combination of Kalyanasundharam with Kennedy would have rendered the claims of the '302 Patent obvious, as is explained in further detail below with regard to multiple rationales that each support a conclusion of obviousness.

"Combining prior art elements according to known methods to yield predictable results"

65. I agree that a POSITA would have been motivated to combine known methods of processor failover, as described in Kalyanasundharam, with the boot-specific processor failover methods described in Kennedy. Even when combining the processor failover elements of Kalyanasundharam with the boot-processor-specific failover elements of Kennedy, each element would perform the same function, and the resulting combination of elements would provide a predictable outcome.

66. I agree that, for example, when Kalyanasundharam's system detects an error or data mismatch between a primary processor core and an auxiliary processor core, it "disable[s] the primary processor core and select[s] one of the auxiliary processor cores to be the primary core processor." *Kalyanasundharam* (Ex. 1005) at 13:16-22. This allows Kalyanasundharam's system to continue processing inputs after a noted failure. *Id.* at 13:23-27. Similarly, Kennedy states that if a default boot processor fails, "control transfers automatically to an alternate processor." *Kennedy* at 2:18-20. Kennedy follows by stating that "This provides fault tolerance and high availability for the system." *Id.* 2:21-22.

67. I agree that a POSITA would have been motivated to combine the known prior art elements of processor failover from a primary processor core to an auxiliary processor core, as taught by Kalyanasundharam, with the teachings of boot processor failover, as described by Kennedy, to increase the reliability and availability of the underlying system. These processes, when combined, would create the predictable outcome of a highly reliable system that would continue to process input data or would continue to perform boot processes even after a failure in one processor has occurred.

68. I agree that, in both Kalyanasundharam and Kennedy, the known method of carrying out the changeover from a failed processor to an alternative processor was to alter the identifier of the processor. For instance, in Kalyanasundharam, an OE field of a processor's configuration register may be changed from "primary processor core" to an "auxiliary processor core" or, in Kennedy, a SLOT_ID field for a processor may be changed from "boot processor" to "alternative processor." *Kalyanasundharam* (Ex. 1005) at 14:48-64, and *Kennedy* (Ex. 1006) at 13:32-44. Providing processor failover by changing a processor's identifier was a known process where each element performs the same function as it did separately. As such, a POSITA would be motivated to add the functionality of boot processor failover by changing slot identifiers, as taught by Kennedy, to the known method of changing configuration register fields, as taught by Kalyanasundharam. *Id.* Moreover, a POSITA would have recognized that the results of this combination are predictable, providing an outcome where an alternative processor performs the functions of a failed processor.

"Some teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention."

69. I agree that Kalyanasundharam's multiprocessor system is designed for high availability. *Kalyanasundharam* (Ex. 1005) at 1:8-11. Kalyanasundharam indicates that "many servers are implemented on fault-tolerant systems so that the servers may be available as close to 24 hours a day as possible." *Id.* at 1:14-17. Kalyanasundharam further states that, "Any mission-critical computing functions may benefit from being assigned to fault-tolerant computer systems." *Id.* at 17-19. With reference to mission-critical computing functions, Kennedy states that, "In a multiprocessor environment, one critical determination is which processor will boot the system." *Kennedy* (Ex. 1006) at 10:2-3. Kennedy further states that, "… if one processor fails, provision for disabling the non-functional processor and allowing another alternative processor to boot the system enhances the availability of the system." As such, a POSITA would be aware that booting a computer system qualifies as a "mission-critical" computing function that, according to Kalyanasundharam, would benefit from being assigned to a fault-tolerant computer system. *Kalyanasundharam* (Ex. 1005) at 1:17-19.

70. I agree that a POSITA would further be motivated to enhance the availability of Kalyanasundharam's system by specifically noting that failover can occur between boot processors, as taught by Kennedy, and as claimed in the '302 Patent. Combining teachings of Kalyanasundharam and Kennedy would lead a POSITA to arrive at the claimed invention of the '302 Patent, namely a system that, in response to detecting an error (e.g., a loss of lockstep (LOL)), determines whether the processor that experienced the error (e.g., the LOL) is a boot processor and, if so, causes the system to recognize another processor as the boot processor (see, for example, claim 21 in the '302 Patent). Kennedy recognizes a failed processor as a boot processor and switches to an alternative processor, while Kalyanasundharam recognizes a failed primary processor and switches to an auxiliary processor in response to a loss of lockstep. *Kennedy* (Ex. 1006) at 7:7-13, and Kalyanasundharam 13:16-22. A POSITA would have reasonably expected that this combination would enhance the availability of Kalyanasundharam's system by incorporating the boot processor failover method of Kennedy. *Kalyanasundharam* (Ex. 1005) at 1:20-25. Indeed, both Kalyanasundharam's and Kennedy's systems describe a

similar computing architecture in which one set of processors fails over to another set of processors upon the occurrence of a failure. As such, the POSITA would reasonably expect to succeed when combining these teachings.

Claim 1: ***[1P] A method comprising:***

71. I agree that Kalyanasundharam teaches a method of processor failover by disclosing: "FIG. 6 is a flowchart illustrating operation of one embodiment of the node controller including the redundancy check circuit shown in FIG. 5 in response to receiving a communication from a processor core." *Kalyanasundharam* (Ex. 1005) at 2:7-10.

[1A] detecting loss of lockstep (LOL) for a processor in a multi-processor system

72. I agree with the analysis in the Request that Kalyanasundharam teaches detecting loss of lockstep (LOL) for a processor in a multi-processor system by disclosing an error detecting mechanism that checks for errors within processors 18A-18B that execute in lockstep. "In embodiments in which **the processor cores 18A-18B execute in lockstep**, the redundancy check circuit 26A may check that corresponding communications are concurrently received from the processor cores 18A-18B (e.g. received on the same clock cycle). For example, if a valid communication is received from one of the processor cores 18A-18B and a valid communication is not concurrently received from the other processor core 18A-18B, **an execution error may be detected**. *Id.* at 4:27-35).

73. I agree that Kalyanasundharam is a multi-processor system. *See e.g. Kalyanasundharam* (Ex. 1005) at Fig. 1, 1:7-11.

[1B] determining that the processor for which said LOL is detected is assigned a role of boot processor

74. I agree that Kalyanasundharam and Kennedy teach determining whether a processor module is assigned a role of system boot processor by disclosing "provision needs to be made to determine which processor will boot the system, and if this "default" boot processor fails to boot the system, an alternative processor, if one is installed, should take over boot operations to enhance the availability of the system." *Kennedy* (Ex. 1006) at 7:5-13. Kalyanasundharam describes use of multiple pairs of processors that operate in lockstep:

> *The processor cores may be configured as N/2 pairs of redundant processor cores. Each pair may redundantly execute different software code from other*

pairs. In some embodiments, the processor cores may be preassigned into pairs for redundant execution purposes.

Kalyanasundharam (Ex. 1005) at 13:60-64.

75. I agree that Kalyanasundharam further notes: "If the masked comparison results indicate that the communication from the primary processor core mismatches with all auxiliary processor cores (that are enabled) (decision block 100, "yes" leg), the control circuit 84 may **disable the primary processor core and select one of the auxiliary processor cores to be the primary processor core** (block 102)." *Kalyanasundharam* (Ex. 1005) at 13:16-22. While Kalyanasundharam determines, upon detecting a loss of lockstep, whether a processor is a primary processor core, Kalyanasundharam fails to specifically indicate that the processor is a "boot processor." In analogous art, Kennedy shows a multiprocessor system that identifies default boot processors and alternative boot processors should the default boot processor fail to boot. *Kennedy* (Ex. 1006) at 7:5-13). This is directly analogous to being assigned the role of boot processor in the '302 Patent's system. "In the multiprocessor system 100 which utilizes a system bus 102, a number of problems can arise if more than one processor type is installed on the 64-bit bus 104. For instance, provision needs to be made to **determine which processor will boot the system**, and if this **"default" boot processor** fails to boot the system, **an alternative processor**, if one is installed, **should take over boot operations** to enhance the availability of the system." *Id.* It would have been obvious to combine Kalyanasundharam with Kennedy to expand Kalyanasundharam's ability to fail over to many different processors including, specifically, boot processors, as noted by Kennedy. Because Kalyanasundharam and Kennedy are both high-reliability computing systems, a POSITA would reasonably expect to succeed when combining the boot processor of Kennedy to the primary and secondary processor cores of Kalyanasundharam.

[1Ci] changing a system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected, to a different value

76. I agree that Kalyanasundharam and Kennedy teach changing a system identifier, used by the system's operating system to recognize the boot processor to a different value by disclosing a control circuit that can "update the PC field of the configuration register 42 to reflect the newly selected primary processor core." *Kalyanasundharam* (Ex. 1005) at

13:22-29. Updating the PC field of a configuration register in Kalyanasundharam is equivalent to changing the system identifier of claim 1 in the '302 patent. Kalyanasundharam states: "As with the discussion above for disabling the auxiliary processor core, the control circuit 84 may simply **disable the primary processor core** from comparison **by clearing the corresponding enable bit in the configuration register 42**, or may disable the primary processor core from executing at all. The control circuit 84 may also **update the PC field of the configuration register 42** to reflect the newly selected primary processor core."

77. In my opinion, to the extent that changing a primary processor core to a secondary processor core (or vice versa) by changing an enable bit in a configuration register is not interpreted as changing a system identifier of a boot processor, Kennedy states, "If the CPU1 120 in slot 1 does not boot (proper boot is indicated by a change in the level of the SLOT SELECT signal line 193 from high to low) before the timer 184 in slot 2 times-out, then the CPU2 132 exits the reset state and attempts to boot the system as an alternate boot CPU." *Kennedy* (Ex. 1006) at 15:9-14. "If the CPU1 120 does not complete boot operations within the time allotted, then the CPU2 132 **takes over default boot operations as an alternative boot processor instead of CPU1** 120. If the CPU1 120 does boot properly, then the CPU2 132 checks in as an additional processor rather than taking over as a boot processor." *Kennedy* (Ex. 1006) at 13:38-44).

[1Cii] which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor

78. I agree that Kalyanasundharam and Kennedy teach causing the system's operating system to not recognize the processor for which said LOL is detected as the boot processor by disclosing that if one CPU fails to complete boot operations, another CPU will be used by the operating system as boot processor. Furthermore, to the extent that Kalyanasundharam is not interpreted to include a boot processor, Kalyanasundharam as modified by Kennedy, would cause the system's operating system to not recognize the processor for which said LOL is detected as the boot processor by indicating that slot select signals used by the operating system for communication would be changed in the event of a failure to boot: "If the CPU1 120 does not complete boot operations within the time allotted, then the CPU2 132 **takes over default boot operations as an alternative boot processor instead of**

CPU1 120. If the CPU1 120 does boot properly, then the CPU2 132 checks in as an additional processor rather than taking over as a boot processor." *Kennedy* (Ex. 1006) at 13:38-44.

[1Di] changing a system identifier of the spare processor to that of the boot processor,

79. I agree that Kalyanasundharam and Kennedy teach changing a system identifier of the spare processor to that of the boot processor by disclosing a control circuit that can "update the PC field of the configuration register 42 to reflect the newly selected primary processor core." *Kalyanasundharam* (Ex. 1005) at 13:22-29. Kalyanasundharam states: "As with the discussion above for disabling the auxiliary processor core, the control circuit 84 may simply **disable the primary processor core** from comparison **by clearing the corresponding enable bit in the configuration register 42**, or may disable the primary processor core from executing at all. The control circuit 84 may also **update the PC field of the configuration register 42** to reflect the newly selected primary processor core." *Id*). Moreover, to the extent that changing a primary processor core to a secondary processor core (or vice versa) by changing an enable bit in a configuration register is not interpreted as changing a system identifier of a boot processor, Kennedy states, "If the CPU1 120 in slot 1 does not boot (proper boot is indicated by a change in the level of the SLOT SELECT signal line 193 from high to low) before the timer 184 in slot 2 times-out, then the CPU2 132 exits the reset state and attempts to boot the system as an alternate boot CPU." *Kennedy* (Ex. 1006) at 15:9-14.

[1Dii] thereby switching the role of boot processor to a spare processor without shutting down the system's operating system.

80. I agree that Kalyanasundharam and Kennedy teach switching the role of boot processor to a spare processor without shutting down the system's operating system by disclosing a control circuit that can "update the PC field of the configuration register 42 to reflect the newly selected primary processor core." *Kalyanasundharam* (Ex. 1005) at 13:22-29. Since the configuration register is part of a redundancy checking circuit, changes to the configuration register would not require an operating system shutdown to implement To the extent that Kalyanasundharam is not interpreted to include a boot processor, Kalyanasundharam as modified by Kennedy, would cause the system's operating system to switch the role of boot processor to a spare processor: "If the CPU1 120 does not

complete boot operations within the time allotted, then the CPU2 132 **takes over default boot operations as an alternative boot processor instead of CPU1** 120. If the CPU1 120 does boot properly, then the CPU2 132 checks in as an additional processor rather than taking over as a boot processor." *Kennedy* (Ex. 1006) at 13:38-44).

Claim 5: ***[5] wherein the spare processor is a processor module that includes a master and slave processors that operate in lockstep***

81. I agree that Kalyanasundharam and Kennedy teach that the spare processor is a processor module that includes a master and slave processors that operate in lockstep by disclosing primary and auxiliary processor cores that are equivalent to master and slave processors.

> *The processor cores may be configured as N/2 pairs of redundant processor cores. Each pair may redundantly execute different software code from other pairs. In some embodiments, the processor cores may be preassigned into pairs for redundant execution purposes.*
>
> *Kalyanasundharam (Ex. 1005) at 13:60-64.*
>
> *Generally, one of the processor cores 18A-18B may be designated as a primary processor core and the other processor core 18A-18B is referred to as the auxiliary processor core. The primary processor core may be the processor core whose execution results will be used as the result of executing the redundantly-executed software code. The auxiliary processor core is used to verify correct execution by the primary processor core.*
>
> *Id. at 4:58-65.*

Claim 6: ***[6] wherein said determining that the processor for which said LOL is detected is assigned the role of boot processor comprises accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor.***

82. In my opinion, to the extent that Kalyanasundharam is not interpreted to disclose accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor, Kalyanasundharam as modified by Kennedy teaches accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor. Kennedy states: "The CPU1 120 has a corresponding **non-volatile**

memory 123 which contains **configuration information and CPU specific initialize and self test code** (ISTC) for the CPU1 120…" *Kennedy* (Ex. 1006) at 5:15-18). It would have been obvious to a POSITA to combine Kalyanasundharam with Kennedy to increase the reliability of the system by storing and accessing CPU-specific information from non-volatile memory, as taught by Kennedy. *Id.*

Claim 7: ***[7] further comprising: determining the processor that is assigned the role of spare processor.***

83. I agree that Kalyanasundharam as modified by Kennedy teaches determining the processor that is assigned the role of spare processor by disclosing: "…provision needs to be made to **determine which processor will boot the system**, and if this **"default" boot processor fails** to boot the system, **an alternative processor**, if one is installed, **should take over boot operations** to enhance the availability of the system." *Kennedy* (Ex. 1006) at 7:5-13). It would have been obvious to a POSITA to combine Kalyanasundharam with Kennedy to increase the reliability of the system by identifying which processors are assigned certain roles, as taught by Kennedy. *Id.*

Claim 8: ***[8] wherein said determining the processor that is assigned the role of spare processor comprises: accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of spare processor.***

84. I agree that Kalyanasundharam as modified by Kennedy teaches accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of spare processor by disclosing an alternative processor that takes over if the default boot processor fails. Kennedy states: "…provision needs to be made to **determine which processor will boot the system**, and if this **"default" boot processor fails** to boot the system, **an alternative processor**, if one is installed, **should take over boot operations** to enhance the availability of the system." *Id.* at 7:5-13). It would have been obvious to a POSITA to combine Kalyanasundharam with Kennedy to avoid system errors by identifying which processors are assigned certain roles, as taught by Kennedy. *Id.*

Claim 13: ***[13P] A method comprising:***

85. *See* Limitation [1P].

[13A] system firmware determining that loss of lockstep (LOL) is detected for a lockstep pair of processors that are assigned the role of boot processor in a system; (*See* Limitation [1A]).

[13B] determining one of said lockstep pair of processors that is not the cause of said LOL

86. *See* Limitation [1B].

[13C] copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to a spare processor

87. I agree that Kalyanasundharam and Kennedy teach copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to a spare processor by disclosing that state information for processor communications is stored. Kalyanasundharam states that "[a]dditionally, the SRO control unit 32 may be configured to update state in the SRO 30 for a given communication in response to completion of various internal activities. For example, if a probe is in the SRO 30 and the corresponding responses from the processors 18A-18B are received for the probe, the SRO control unit 32 may update the state in the entry storing the probe to record the receipt of the response and to record the coherency State reported in the response." *Kalyanasundharam* (Ex. 1005) at 7:34-43.

[13Di] changing a system identifier, used by the system's operating system to recognize the boot processor, of said lockstep pair of processors that are assigned the role of boot processor, to a different value

88. *See* Limitation [1Ci].

[13Dii] which causes the s system's operating system to not recognize the lock step pair of processors for which said LOL is detected as the processor

89. *See* Limitation [1Cii].

[13E] changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to said spare processor.

90. *See* Limitations [1Di] and [1Dii].

Claim 17: (I agree that Claim 17 recites substantially the same subject matter as Claim 1 ([1Dii]) and is therefore obvious for the same reasons as Claim 1 above.

Claim 21: **[21P]** ***A system comprising:***

91. I agree that Kalyanasundharam and Kennedy disclose a system 10 of FIG. 1 that includes nodes 12A-12B, memories 14A-14B, and peripheral devices 16A-16B" *Kalyanasundharam* (Ex. 1005) at 2:26-29.

[21A] a plurality of processor modules that each include a master processor and a slave processor that operate in lockstep

92. I agree that Kalyanasundharam and Kennedy teach a plurality of processor modules that each include a master processor and a slave processor that operate in lockstep by disclosing primary and auxiliary processors that operate in lockstep. "In one embodiment, **the processor cores 18A-18B may execute in lockstep** (that is, may execute the same instructions on the same clock cycle)." *Kalyanasundharam* (Ex. 1005) at 4:15-18. "Generally, one of the processor cores 18A-18B may be designated as a primary processor core and the other processor core 18A-18B is referred to as the auxiliary processor core. The primary processor core may be the processor core whose execution results will be used as the result of executing the redundantly-executed software code. **The auxiliary processor core is used to verify correct execution by the primary processor core**." *Id.* at 4:58-65). ***[21B] an operating system;*** ("Generally, the processor cores 18A-18B may use the 15 interface(s) to the node controller 20A to communicate with other components of the computer system 10 (e.g., peripheral devices 16A-16B, processor cores 18C-18D, the memory controllers 22A-22B, etc.)." *Id.*, 3:15-18.

[21C] error detection logic operable to detect loss of lockstep (LOL) for at least one of said processor modules

93. I agree that Kalyanasundharam and Kennedy teach error detection logic operable to detect loss of lockstep (LOL) by disclosing an error detecting mechanism that checks for errors within processors 18A-18B that execute in lockstep. "In embodiments in which **the processor cores 18A-18B execute in lockstep**, the redundancy check circuit 26A may check that corresponding communications are concurrently received from the processor cores 18A-18B (e.g., received on the same clock cycle)." For example, if a valid communication is received from one of the processor cores 18A-18B and a valid communication is not concurrently received from the other processor core 18A-18B, **an execution error may be detected**. *Kalyanasundharam* (Ex. 1005) at 4:27-35.

[21Di] system firmware operable, responsive to the detection of LOL for a first of said processor modules, to determine whether said first processor module is assigned a role of system boot processor

94. I agree that Kalyanasundharam and Kennedy teach system firmware operable, responsive to the detection of LOL, to determine whether a processor module is assigned a role of system boot processor by disclosing "provision needs to be made to determine which processor will boot the system, and if this "default" boot processor fails to boot the system, an alternative processor, if one is installed, should take over boot operations to enhance the availability of the system." *Kennedy* (Ex. 1006) at 7:5-13. Kalyanasundharam notes: "If the masked comparison results indicate that the communication from the primary processor core mismatches with all auxiliary processor cores (that are enabled) (decision block 100, "yes" leg), the control circuit 84 may **disable the primary processor core and select one of the auxiliary processor cores to be the primary processor core** (block 102)." *Kalyanasundharam* (Ex. 1005) at 13:16-22. While Kalyanasundharam determines, upon detecting a loss of lockstep, whether a processor is a primary processor core, Kalyanasundharam fails to specifically indicate that the processor is a "boot processor." In analogous art, Kennedy shows a multiprocessor system that identifies default boot processors and alternative boot processors should the default boot processor fail to boot. This is directly analogous to being assigned the role of boot processor in the '302 Patent's system. "In the multiprocessor system 100 which utilizes a system bus 102, a number of problems can arise if more than one processor type is installed on the 64-bit bus 104. For instance, provision needs to be made to **determine which processor will boot the system**, and if this **"default" boot processor** fails to boot the system, **an alternative processor**, if one is installed, **should take over boot operations** to enhance the availability of the system." *Kennedy* (Ex. 1006) at 7:5-13. It would have been to combine Kalyanasundharam with Kennedy to expand Kalyanasundharam's ability to fail over to many different processors including, specifically, boot processors, as noted by Kennedy. Because Kalyanasundharam and Kennedy are both high-reliability computing systems, a POSITA would reasonably expect to succeed when combining the boot processor of Kennedy to the primary and secondary processor cores of Kalyanasundharam.

[21Dii] wherein if determined that said first processor module is assigned the role of system boot processor, said system firmware is further operable to cause said operating system to recognize another processor module as system boot processor without shutting down said operating system.

95. I agree that Kalyanasundharam and Kennedy teach system firmware that causes the operating system to recognize another processor module as system boot processor without shutting down the operating system by disclosing a control circuit that can "update the PC field of the configuration register 42 to reflect the newly selected primary processor core." *Kalyanasundharam* (Ex. 1005) at 13:22-29. Kalyanasundharam states: "As with the discussion above for disabling the auxiliary processor core, the control circuit 84 may simply **disable the primary processor core** from comparison **by clearing the corresponding enable bit in the configuration register 42**, or may disable the primary processor core from executing at all. The control circuit 84 may also **update the PC field of the configuration register 42** to reflect the newly selected primary processor core." *Id.*

Claim 22: ***[22] wherein the another processor module is a processor module assigned a role of spare processor.***

96. I agree that Kalyanasundharam and Kennedy teach that the other processor module is a processor module assigned a role of spare processor by disclosing an alternative processor that takes over if the default boot processor fails. Kennedy states: "…provision needs to be made to **determine which processor will boot the system**, and if this **"default" boot processor fails** to boot the system, **an alternative processor**, if one is installed, **should take over boot operations** to enhance the availability of the system." *Id.* at 7:5-13). It would have been obvious to a POSITA to combine Kalyanasundharam with Kennedy to increase reduce system errors by assigning processor roles, as taught by Kennedy. *Id.*

Claim 25: (I agree that Claim 25 recites substantially the same subject matter as Claim 8 ([8]) and is therefore obvious for the same reasons as Claim 8 above.

Claim 26: (I agree that Claim 26 recites substantially the same subject matter as Claim 21 ([21]) and is therefore obvious for the same reasons as Claim 21 above.

G. Ground 2 (Kennedy in view of Nguyen and Safford) Presents Substantial New Questions of Patentability

97. I agree with the analysis in the Request that Kennedy in view of Nguyen and Safford renders obvious each and every element of claims 1-27. For ease of reference, I identify evidence supporting my opinions regarding obviousness below for 1-27 below. I have also reviewed and assisted in the preparation of Appendices A-C. Appendix B provides a detailed mapping of Kennedy in view of Nguyen and Safford to claims 1-27 that accurately reflects my opinions regarding obviousness.

1. Motivation to Combine Kennedy, Nguyen, and Safford

98. An overview of Kennedy is provided in the Request. I have reviewed and agree with that overview of Kennedy.

99. In my opinion, Kennedy is analogous art to the '302 Patent because Kennedy is from the same field of endeavor (e.g., providing recovery from failure in a high-availability computing system) and is reasonably pertinent to a problem faced by the inventors of the '302 Patent (e.g., detecting and/or recovering from processor failures in multiprocessor systems). See Ex. 1006 at 2:15-21, 10:4-7; see also Ex. 1001 at 4:34-53, 3:16-18.

100. An overview of Nguyen is provided in the Request. I have reviewed and agree with that overview of Nguyen.

101. In my opinion, Nguyen is analogous art to the '302 Patent because Nguyen is both from the same field of endeavor (e.g., providing recovery from failure in a high-availability computing system) as the '302 Patent and is reasonably pertinent to a problem faced by the inventors of the '302 Patent (e.g., system crashes caused by loss of lockstep). *See Nguyen* (Ex. 1004) at 3:21-22, 1:45-47; *see also '302 Patent* (Ex. 1001) at 3:16-18.

102. An overview of Safford is provided in the Request. I have reviewed and agree with that overview of Safford.

103. In my opinion, Safford is analogous art to the '302 Patent because Safford is both from the same field of endeavor (e.g., providing recovery from failure in a high-availability computing system) as the '302 Patent and is reasonably pertinent to at least one problem faced by the inventors of the '302 Patent (e.g., system crashes caused by loss of lockstep). *Madisetti Decl.* (Ex. 1003) ¶ XX. For example, the '302 Patent, in focusing on "the increased desire for many systems to maintain high availability" notes the problem that

"crashing the system each time LOL is detected is not an attractive proposition." Ex. 1001 at 3:16-18. Similarly, Safford in focusing on "improv[ing] availability of the processor assets of the computer system," notes that "loss of lock step, if not promptly corrected, may cause the computer system 18 to crash." *Id.* at 3:21-22 and 1:45-47.

104. In my opinion, as mentioned above, Kennedy describes failing over a boot processor to an alternate boot processor in response to a failure of the boot processor. See, e.g., *Kennedy* (Ex. 1006) at 7:5-13. For example, Kennedy states that "one default boot CPU is designated to boot the system, and if this CPU fails, control transfers automatically to an alternate processor." *Id.* at 2:17-21. To the extent that Kennedy does not explicitly describe failing over the boot processor **in response to a loss of lockstep** for the boot processor, Kennedy as modified by Nguyen would have rendered failover of a boot processor obvious. *See, e.g., Nguyen* (Ex. 1008), ¶¶ [0008] and [0054]-[0055].

105. For example, Nguyen describes detecting a loss of lockstep within a processor. *Id.* I agree that Nguyen describes a processor with multiple "[e]xecution cores 120 [that] operate in lock step during normal FRC mode." *Id.* ¶ [0024]. "In FRC mode, one of execution cores 120 may be designated as the master." *Id.* ¶ [0027]. "The master refers to the execution core that is responsible for updating the resources shared by execution cores 120." *Id.* "The other execution core 120 may be designated as the slave." *Id.* "The slave is responsible for generating results from the same code sequence to be checked against those of the master." *Id.* If an FRC error occurs at the processor, the master and slave cores have lost lock step and a recovery process in instigated. *Id.* ¶ [0024].

106. In my opinion, a POSITA would have been further motivated to combine the teachings of Kennedy, Nguyen and Safford to improve the reliability of Kennedy's system (e.g., by failing over the boot processor to an alternative processor in response to a loss of lockstep). For example, a POSITA would have been motivated to modify Kennedy's system, which implements boot processor failover in response to an error, by Nguyen's teachings about detection of an error, i.e., a loss of lockstep. *See Id.* A POSITA would have been motivated to do so because a POSITA would have recognized the obvious benefits and advantages of enabling Kennedy's system to dynamically fail over boot processors in response to a loss of lockstep, especially in view of Kennedy's express objective of "provid[ing] fault tolerance and high availability." *Kennedy* (Ex. 1006) at 2:21-22.

107. In my opinion, to the extent that Kennedy and Nguyen are not interpreted as changing a system identifier for a boot processor upon such a loss of lockstep, Kennedy and Nguyen, as modified by Safford, would have rendered such changing of a system identifier obvious. For instance, Safford's system implements multiple processors operating in a lock step mode. *Safford* (Ex. 1007) at 2:55-60. One pair of these processors "operating in a lock step mode" initially "appear[s] to the computer system…to be…a **logical CPU 0.**" *Id.* at 3:4-6. A POSITA would recognize that the "logical CPU 0" designation is equivalent to a system identifier used by the system to determine which processor is boot processor. *Id.* Later, that **logical CPU 0** designation is reassigned to another processor during failover. *See Safford* (Ex. 1007) at 3:4-5, 4:52-53. A new processor pair 125/127 then becomes the **logical CPU 0** in the computer system 100." *Id.* As such, the system identifier "logical CPU 0" is reassigned to a working processor during failover. *Id.*

108. In my opinion, a POSITA would have recognized Safford's reassignment of the logical CPU 0 designation from one processor to another as being a change in system identifier used by the operating system to identify the boot processor. The POSITA would have been motivated to combine Safford's change of system identifier with Kennedy's change of processor during failover (and specifically during a loss of lockstep, as taught by Nguyen) to further increase the likelihood of a successful failover between processors. Moreover, a POSITA would have been motivated to do so because Safford's changed system identifier would enable Kennedy's system to seamlessly transfer executing processes from failing boot processor to an alternative processor without affecting the operating system (OS), the applications running on Nguyen's system, or the high availability of Kennedy's system. Indeed, a POSITA would have recognized the obvious benefits and advantages of providing Kennedy's system with a changeable system identifier, especially in view of Safford's suggestion that, upon processor failure, the failed processors are taken offline, and a new processor pair "becomes the logical CPU 0 in the computer system 100." *Safford* (Ex. 1007) at 4:38-53.

109. In my opinion, a POSITA would have had a reasonable expectation of success in combining the teachings of Kennedy, Nguyen, and Safford. For example, Kennedy describes failing over a boot processor to an alternate boot processor. *See, e.g., Kennedy* (Ex. 1006) at 7:5-13. Even if Kennedy does not explicitly describe detecting a loss of

lockstep, Kennedy as modified by Nguyen would have rendered such a technique obvious. *See, e.g., Nguyen* (Ex. 1004), ¶¶ [0008] and [0054]-[0055]. For example, Nguyen describes detecting a functional redundancy checking (FRC) error (*e.g.,* loss of lockstep) in the master execution core. *See Id.* As mentioned above, the '302 Patent describes such cores as each being an example of a processor. *See, e.g., '302 Patent* (Ex. 1001) at 6:45-47. Because Kennedy's teachings are directed to failing over the boot processor, because Nguyen describes a loss of lockstep as triggering such processor failover, and because Safford describes changing a system identifier during failover, the proposed combination of Kennedy, Nguyen, and Safford would have been expected to succeed. In my opinion, it was well within the skill of a POSITA to modify Kennedy's system by Nguyen and Safford in the manner proposed, and Kennedy did not teach away from or discourage applying such a modification. *See Id.* More specifically, Kennedy does not limit or otherwise discourage application of its techniques from being triggered by a loss of lockstep.

110. Moreover, I agree that, even if neither Kennedy nor Nguyen explicitly changed a system identifier upon failover, Kennedy as modified by Nguyen and Safford would have rendered obvious such an identifier change. *See, e.g., Safford* (Ex. 1007) at 4:38-53. In my opinion, it was well within the skill of a POSITA to further modify Kennedy's system by Safford in the manner proposed, and Kennedy did not teach away from or discourage applying such a further modification. That is, Kennedy does not limit or otherwise discourage application of its techniques to implementations that may implement such a change in system identifier during failover. Accordingly, the proposed combination of Kennedy, Nguyen, and Safford—in which Kennedy's system is modified by Nguyen and Safford to achieve dynamically replacing a failed boot processor with an alternative processor in response to a loss of lockstep while, in the process, changing a system identifier—would have been reasonably expected to succeed. *Id.* Consistent with the Supreme Court's holding in *KSR*, the rationales provided below further support this conclusion of obviousness.

"Combining Prior-Art Elements According to Known Methods to Yield Predictable Results" MPEP § 2143.I(A)

111. In my opinion, a POSITA would have combined the teachings of Kennedy, Nguyen, and Safford according to known methods to yield predictable results. Combining Kennedy, Nguyen, and Safford would yield predictable results because, pursuant to MPEP § 2143,

(1) together, Kennedy, Nguyen, and Safford disclose each element of claims 1-27, (2) a POSITA could have combined these references by known methods, (3) these elements perform the same function in isolation and in the combination, and (4) the POSITA would have recognized the results of the proposed combination as being predictable. As shown below, the proposed combination of Kennedy, Nguyen, and Safford satisfies all the prongs of this rationale.

112. First, I agree that the combination of Kennedy, Nguyen, and Safford discloses each element recited in the applicable claims. (See Claim Chart in Appendix B).

113. Second, I agree that a POSITA could have combined the prior-art elements by known methods. For example, Kennedy described failing over boot processors to spare processors when an error is detected. *See, e.g., Kennedy* (Ex. 1006) at 1:16-19, 2:15-22, 7:5-13. Nguyen described a system that determines whether an error has occurred such as a loss of lockstep between processor cores (i.e., that a processor is out of lockstep). *See, e.g., Nguyen* (Ex. 1004) ¶¶ [0008], [0054]-[0055]. The POSITA would have combined Kennedy's well known boot processor failover method with Nguyen's system of detecting a loss of lockstep to fail over from a processor having the designation "boot processor" to "alternative processor", as taught by Kennedy." *Kennedy* (Ex. 1006) at 2:15-22, 7:5-13; Still further, Safford described changing a system identifier so that a spare processor receives the designation of logical CPU 0 upon failure of the previous CPU 0. *See, e.g., Safford* (Ex. 1007) at 4:38-53. A POSITA would have been motivated to combined Kennedy's modified system with Safford's change of system identifier by the methods disclosed in Safford to perform the boot processor failover without affecting Kennedy's OS or corresponding applications. *Id.*

114. Third, I agree that each prior-art element performs the same function in isolation and in the combination. For example, in isolation and in the combination, Kennedy's system automatically transfers the role of boot processor to a spare processor in response to a detected failure, Nguyen's system detects a loss of lockstep, and Safford's system changes a system identifier. *Id.* Accordingly, the functions performed by the prior-art elements do not change from the individual Kennedy, Nguyen, and Safford references to the proposed combination.

115. Fourth, in my opinion, the POSITA would have recognized the results of the proposed combination as being predictable. For example, the POSITA would have recognized that, by failing over Kennedy's boot processor to an alternate processor in response to Nguyen's detection of a loss of lockstep while implementing Safford's change in system identifier, Kennedy's modified system would have been able to predictably fail over the boot processor to an alternate processor in response to a loss of lockstep.

"Applying a Known Technique to a Known Device Ready for Improvement" MPEP § 2143.I(D)

116. In my opinion, a known technique could have been applied to a known device ready for improvement to yield predictable results. Combining Kennedy, Nguyen, and Safford would apply a known technique to a known device because, pursuant to MPEP § 2143, (1) Kennedy disclosed a base device upon which the claimed invention can be seen as an improvement, (2) Nguyen disclosed a known technique that is applicable to the base device, (3) Safford disclosed a known technique that is applicable to the base device, (4) a POSITA would have recognized that applying the known technique would have yielded predictable results, and (5) the predictable results would have improved the base device. As shown below, the proposed combination of Kennedy, Nguyen, and Safford satisfies all the prongs of this rationale.

117. First, I agree that Kennedy disclosed a base device upon which the claimed invention can be seen as an improvement. For example, Kennedy disclosed a device that replaces the boot processor to an alternate processor in response to a failure of the boot processor. *See, e.g., Kennedy* (Ex. 1006) at 7:5-13. Similarly, the claimed invention transfers the role of boot processor to a spare processor in response to a failure such as a loss of lockstep. *'302 Patent* (Ex. 1001), claim 1. Accordingly, an improvement appears to be that the claimed invention fails over from the boot processor to the spare processor in response to a loss of lockstep.

118. Second, I agree that the prior art also disclosed known techniques that are applicable to the base device. For example, Nguyen described a known technique for detecting a functional redundancy checking (FRC) error (*e.g.,* loss of lockstep) in the master execution core. *See, e.g., Nguyen* (Ex. 1004), ¶¶ [0008] and [0054]-[0055]; *see also* Safford described a known technique of changing a system identifier in response to a processor failure. *Safford*, (Ex.

1007) at 4:38-53. Because Nguyen's known technique involves detecting a loss of lockstep, because Kennedy involves replacing a failed processor with an alternative processor, and because Safford's known technique involves changing a system identifier during failover, Nguyen's and Safford's known techniques would have been applicable to Kennedy's device, which is very similar to and includes the same components and functionality as Kennedy's device.

119. Third, in my opinion, a POSITA would have recognized that applying the known techniques to the base device would have yielded predictable results. For example, the POSITA would have recognized that applying Nguyen's and Safford's known techniques to Kennedy's device would have yielded predictable results. *Id.* More specifically, the POSITA would have recognized that, by applying Nguyen's and Safford's known techniques, Kennedy's device would have been able to predictably fail over the boot processor to the alternative processor by changing a system identifier of a failed processor in response to a loss of lockstep. *Id.*

120. Fourth, I agree that the predictable results would have improved the base device. *Id.* For example, the predictable results would have improved the reliability of Kennedy's device (*e.g.,* by failing over the boot processor to an alternative processor by changing a system identifier in response to loss of lockstep). *Id.* More specifically, Kennedy's device would become more reliable by failing over the boot processor to an alternative processor such that the alternative processor immediately assumes responsibility for booting the underlying system in case a reboot is needed. *Id.* By doing so, Kennedy's device as modified by Nguyen and Safford would avoid a situation in which the underlying system attempts to reboot without a designated, functional boot processor (i.e., a logical CPU 0) after a loss of lockstep.

"Some Teaching, Suggestion, or Motivation in the Prior Art Would Have Led a POSITA to Combine Prior-Art Teachings to Arrive at the Claimed Invention" MPEP § 2143.I(G)

121. In my opinion, the teachings of Kennedy, Nguyen, and Safford would have led a POSITA to arrive at the claimed invention because, pursuant to MPEP § 2143, (1) there was some teaching, suggestion, or motivation either in the references themselves or in the knowledge generally available to one of ordinary skill in the art to combine the references' teachings and (2) there was a reasonable expectation of success for the combination. As shown

below, the proposed combination of Kennedy, Nguyen, and Safford satisfies all the prongs of this rationale.

122. First, I agree that Nguyen emphasized and/or reiterated certain teachings, suggestions, and/or motivations that would have led a POSITA to combine the teachings of Kennedy, Nguyen, and Safford. *See, e.g., Nguyen* (Ex. 1007) ¶ [0006], [0008], [0054]; *see also* For example, Nguyen taught that "[e]rrors that are not detected at the FRC boundary can lead to [Silent Data Corruption] SDC." *Id.* ¶ [0006]. Further, Nguyen taught that, "for FRC-enabled processors, such errors are likely to be manifested as FRC errors, since they take the execution cores out of lock-step." *Id.* ¶ [0008]. Nguyen also taught that soft errors, which pose a major challenge to both high reliability and high availability, "result from collisions between high-energy particles, e.g. alpha particles, and charge storing nodes." *Id.* ¶ [0004]. Any or all of these teachings, suggestions, and/or motivations from Nguyen would have led a POSITA to combine the teachings of Kennedy, Nguyen, and Safford.

123. Still further, I agree that Safford emphasized and/or reiterated certain teachings, suggestions, and/or motivations that would have led a POSITA to combine the teachings of Kennedy, Nguyen, and Safford. *See, e.g., Safford* (Ex. 1006) at 2:55-63. For example, Safford taught that its technique of changing system identifiers would ensure that at least one "good" processor would continue processing, while the "bad" processor would be turned off and could be recovered at a later time. *Id.* at 3:43-50. To ensure that the bad processor is no longer implemented by the system, Safford changes the "logical CPU 0" designation from the failed processor to another processor (e.g., from failed processors 111/113 to spare processors 125/127). *Id.* at 4:47-57. Changing the system identifier of a processor would lend further stability to the system, thereby increasing its reliability and high availability. Any or all of these teachings, suggestions, and/or motivations from Safford would have led a POSITA to combine the teachings of Kennedy and Safford.

124. In my opinion, even disregarding the express teachings, suggestions, and/or motivations cited from Nguyen, the POSITA would have been motivated to combine Kennedy, Nguyen, and Safford based on their own knowledge. *See, e.g.,* MPEP § 2143.I(G); *see also DyStar Textilfarben GmbH & Co. Deutschland KG v. C.H. Patrick Co.,* 464 F.3d 1356, 1360 (Fed. Cir. 2006). "Because the desire to enhance commercial opportunities by improving a product or process is universal—and even common-sensical—we have held

that there exists in these situations a motivation to combine prior art references even absent any hint of suggestion in the references themselves." *Id.* "In such situations, the proper question is whether the ordinary artisan possesses knowledge and skills rendering him *capable* of combining the prior art references." *Id.* The POSITA would have possessed the knowledge and skills to combine the teachings of Kennedy, Nguyen, and Safford in this way. *See, e.g.,* Still further, the POSITA would have been motivated to combine Kennedy with Nguyen and Safford to increase the reliability of its high-reliability system by failing over from a boot processor to an alternative processor while changing the system identifier of the failed processor in response to a loss of lockstep.

125. Second, I agree that there was a reasonable expectation of success for the proposed combination of Kennedy, Nguyen, and Safford. For example, Kennedy described failing over a boot processor to an alternate boot processor in response to a failure of the boot processor. *See, e.g., Kennedy* (Ex. 1005) at 7:5-13. Even if Kennedy did not explicitly describe failing over the boot processor in response to a loss of lockstep for the boot processor, Kennedy as modified by Nguyen did. *See, e.g., Nguyen* (Ex. 1004), ¶¶ [0008] and [0054]-[0055]. Even if neither Kennedy nor Nguyen is interpreted as explicitly changing a system identifier in response to a failure, Kennedy as modified by Nguyen and Safford would have rendered obvious such a change in system identifier. Kennedy does not discourage or otherwise limit its disclosure to replacing the boot processor with an active spare processor in response to a non-lockstep error, and it was well within the skill of a POSITA to modify Kennedy's system by Nguyen and Safford in the manner proposed, nor does Kennedy discourage the change of a system identifier. *See, e.g.,* Accordingly, the proposed combination of Kennedy, Nguyen, and Safford—in which Kennedy's system is modified by Nguyen and Safford to achieve dynamically replacing a failed boot processor with an alternative processor while changing the system identifier of the failed processor in response to a loss of lockstep—would have been reasonably expected to succeed. *Id.*

126. Still further, to the extent that Kennedy and Nguyen do not explicitly describe the use of "hot spare" processors when failing over after a loss of lockstep, I agree that Safford recovers from the loss of lockstep by transferring the role of the failed processor pair to a hot standby processor pair. *Id.* at 4:37-40. As noted above, the hot standby processor pairs of Safford are ready to assume the functionality of a failed processor at any moment. *Id.*

As such, a POSITA would be motivated to further improve the reliability of high-availability computer processing systems by transferring the role of a failed processor pair to a hot standby processor pair. Implementing hot standby processing pairs ensures that the failover occurs quickly and seamlessly and, as such, the POSITA would be motivated to combine Safford with the teachings of Kennedy and Nguyen. *Id.* at 4:40-54. Moreover, because each of the computing systems described in Nguyen, Kennedy, and Safford describe failover from one processor to another processor, the POSITA would reasonably expect to succeed in adding the hot spare processor pair to the computing architectures of Kennedy and Nguyen.

127. In my opinion, like Nguyen and Kennedy, Safford also describes a multi-processor system capable of recovering from an error such as a loss of lockstep detected in connection with a certain processor pair. Ex. 1007, Abstract. Just as the prior art boot processor failover elements of Kennedy would have been added to the general-purpose processor failover elements of Nguyen to yield the predicable result of a more highly available computing system, the prior art elements of processor failover described in Safford would have been similarly combined with Nguyen and Kennedy "to improve reliability of processing assets…" *Id.* at 2:55-57.

128. Specifically, in my opinion, to the extent that Nguyen and Kennedy do not explicitly describe the use of "hot spare" processors when failing over after a loss of lockstep, Safford recovers from the loss of lockstep by transferring the role of the failed processor pair to a hot standby processor pair. *Id.* at 4:37-40. As noted above, the hot standby processor pairs of Safford are ready to assume the functionality of a failed processor at any moment. *Id.* As such, a POSITA would be motivated to further improve the reliability of high-availability computer processing systems by transferring the role of a failed processor pair to a hot standby processor pair. Implementing hot standby processing pairs ensures that the failover occurs quickly and seamlessly and, as such, the POSITA would be motivated to combine Safford with the teachings of Nguyen and Kennedy. *Id.* at 4:40-54. Moreover, because each of the computing systems described in Nguyen, Kennedy, and Safford describe failover from one processor to another processor, the POSITA would reasonably expect to succeed in adding the hot spare processor pair to the computing architectures of Nguyen and Kennedy.

Claim 1: ***[1P] A method comprising:***

129. I agree that Kennedy describes a method outlined in as follows: "The architecture provides a method and apparatus to interface multiple processors to the backplane system bus such that upon start-up of the computer system, one default boot CPU is designated to boot the system, and if this CPU fails, control transfers automatically to an alternate processor."." *Kennedy* (Ex. 1006) at 2:15-21.

[1A] detecting loss of lockstep (LOL) for a processor in a multi-processor system

130. I agree that Kennedy, as modified by Nguyen and Safford, teaches detecting loss of lockstep for a processor in a multi-processor system by disclosing "execution cores 120 [that] operate in lock step" and an FRC checker that compares "signals from execution cores 120 . . . to detect soft errors in either execution core." *Nguyen* (Ex. 1004) ¶ [0029]. "Execution cores 120 operate in lock step during normal FRC mode…" *Id.* ¶ [0024]. "In FRC mode, signals from execution cores 120 may be processed by FRC checker 130, which compares them **to detect soft errors in either execution core**." *Id.* ¶ [0029]. These soft errors or mismatches that are detected between processor pair outputs are equivalent to detecting a loss of lockstep. Indeed, Kennedy as modified by Nguyen further notes, "For this reason, FRC checker 130 is temporarily disabled **if error detector 140 detects an error in either error pipeline 128**, since execution cores 120 are **no longer in lock step**." *Id.* ¶ [0023]. As such, Kennedy as modified by Nguyen detects errors that indicate that two execution cores are no longer operating in lock step. This is equivalent to "detecting a loss of lockstep" as claimed in claim 1 of the '302 Patent. To the extent that Nguyen is not held to teach a multi-processor system, Kennedy states, "The present invention provides a **multiprocessor interconnection architecture** with a backplane system bus for use with multiple processors." *Kennedy* (Ex. 1006) at 2:13-15. In my opinion, a POSITA would have been motivated to improve the reliability and robustness of Kennedy's multiprocessor system by detecting loss of lockstep, as taught by Nguyen, in order to fail over to an alternate boot processor in scenarios where a processor's failure is due to loss of lockstep.

[1B] determining that the processor for which said LOL is detected is assigned a role of boot processor

131. I agree that Kennedy, Nguyen, and Safford teach determining that the processor for which said LOL is detected is assigned a role of boot processor by disclosing detecting a

recoverable error and allowing a processor designated as "alternative processor" to take over for a processor designated as "boot processor." *Kennedy* (Ex. 1006) at 7:9-12. Kennedy discloses that: "…provision needs to be made to determine which processor will boot the system, and if this "default" boot processor fails to boot the system, an alternative processor, if one is installed, should take over boot operations to enhance the availability of the system." *Id.* at 7:5-13. Kennedy further states: "When multiple CPUs are installed in the system, if the **default boot processor exceeds the time allotted for it to boot (it fails)**, an alternative boot processor can take over boot operations as explained below. *Id.* at 9:65-68. Thus, Kennedy determines both that a recoverable failure has occurred, and that the failure occurred at a default boot processor.

132. In my opinion, to the extent that Kennedy does not specifically indicate that the error detected was a loss of lockstep, Nguyen determines that an LOL occurs on its processor cores. *Nguyen*, (Ex. 1004) ¶ [0027]. "Execution cores 120 operate in lock step during normal FRC mode…" *Id.* ¶ [0024]. "In FRC mode, signals from execution cores 120 may be processed by FRC checker 130, which compares them **to detect soft errors in either execution core**." *Id.* ¶ [0029]. These soft errors or mismatches that are detected between processor pair outputs are equivalent to detecting a loss of lockstep. Indeed, Nguyen further notes, "For this reason, FRC checker 130 is temporarily disabled **if error detector 140 detects an error in either error pipeline 128**, since execution cores 120 are **no longer in lock step**." *Id.* ¶ [0023]. As such, Nguyen detects errors that indicate that its two execution cores are no longer operating in lock step. This is equivalent to "detecting a loss of lockstep" as claimed in claim 1 of the '302 Patent.

133. I agree that, in this manner, the combination of Kennedy, Nguyen, and Safford determines whether a recoverable failure has occurred and whether the failure has occurred at a designated default boot processor. *Kennedy* (Ex. 1006) at 7:5-13. Nguyen indicates that such recoverable failures may include a loss of lockstep. *Nguyen*, (Ex. 1004) ¶ [0027]. Because Kennedy, Nguyen, and Safford describe high-availability computing architectures, a POSITA would, in my opinion, be motivated to combine these references and would have a reasonable expectation that combining the two references would lead to a more reliable multiprocessor system.

[1Ci] changing a system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected, to a different value

134. I agree that Kennedy as modified by Nguyen and Safford teaches this limitation by disclosing that, "To improve reliability of processing assets, a computer system employs lock stepped processor cores that operate in a **master/check pair**." *Safford* (Ex. 1007) at 2:55-57. Safford further describes one pair of these processors "operating in a lock step mode" as initially "appear[ing] to [a] computer system…to be…a **logical CPU 0,**" but later that **logical CPU 0** designation is reassigned to another processor upon failover. *See Safford* (Ex. 1007) at 3:4-5, 4:52-53. For example, "The processor pair 125/127 then becomes the **logical CPU 0** in the computer system 100". *Id.* A POSITA would have recognized Safford's logical CPU 0 as being the designation used by computing system 100 to identify its boot processor at least because, in multi-processor system architectures like Safford's, "**logical CPU 0**" was commonly used to indicate that a processor had been selected as a boot processor. *See, e.g., Cepulis* (Ex. 1014) at 2:52-64, *see also* For example, during boot-processor selection,

> If CPU P0 is not functioning properly and cannot perform even the most basic functions, it is designated as inoperative. The hardware system then awakens CPU P1 and repeats the process of testing the CPU. If CPU P1 is operational, then it is designated as CPU L0, and it boots the remainder of the system. On the other hand, if CPU P1 also fails, it is also given an inoperative designation. The computer system then turns on CPU P2, and repeats the process. The process repeats until an operational microprocessor is found to perform the CPU L0 functions.[1]
>
> *Cepulis* (Ex. 1014) at 2:52-64.

135. In my opinion, a POSITA would have also recognized "CPU 0" (and other designators taught by Safford, such as "CPU 1") as being system designations used by computing system 100 to identify processors 111 and 113 in Safford's system (e.g., processors 111/113 are initially recognized as "logical CPU 0" and, after failure of one of these processors, processors 125/127 are designated as "logical CPU 0." *Safford* (Ex. 1007) at 3:4-5, 4:52-53. The use of system designators by a computing system is generally necessary

[1] Cepulis is provided herein to explain the meaning of the term "boot processor" used in Safford. *See* M.P.E.P. § 2131.01.

when the role of boot processor can be assigned and reassigned to a processor regardless of its physical position within a system. *Safford* (Ex. 1007) at 4:52-53 ("The processor pair 125/127 then becomes the **logical CPU 0** in the computer system 100"). Thus, in my opinion, a POSITA would have understood "**logical CPU 0**" to be the boot-processor designation used by an operating system to determine which of its processors is the boot processor and/or to distinguish its boot processor from its non-boot processors. *See* Moreover, Safford teaches the storing of state information for failed processors. *Safford*, (Ex. 1007) at 4:40-51. Because the state information of Safford's processors is copied from memory, a POSITA would have understood the memory to be storing the boot-processor designation (*e.g.,* logical CPU 0) for the processors so assigned by Safford's system. *See Id.* In view of these citations and the knowledge that a boot processor was a common component identified as "logical CPU 0" in computer systems (*e.g.,* Safford's system), a POSITA would have understood Safford as disclosing data storage that stores information assigning the role of boot processor to one of the processors and changing that assignment by changing the identifier "logical CPU 0" during a failover. *See supra* § I.C.2; *see also* Still further, A POSITA would have combined Safford with Kennedy and Nguyen because each is concerned with providing processor failover mechanisms in high-availability environments, and because Kennedy and Nguyen would perform the failover more reliably when changing a system identifier as taught by Safford. *Id., see also*

[1Cii] which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor

136. I agree that Kennedy, Nguyen, and Safford teach causing the system's operating system to not recognize the processor for which said LOL is detected as the boot processor by disclosing that a failed processor pair 111/113 is taken off-line, that its state information is copied to another processor pair (e.g., a "hot standby" processor 125/127), and that [t]he processor pair 125/127 then becomes the logical CPU 0 in the computer system 100, and the computer system 100 operates without a hot standby processor pair. *Safford* (Ex. 1007) at 4:38-53. Thus, in Kennedy's system as modified by Nguyen and Safford, in the event of a processor failure (such as a loss of lockstep, as taught by Nguyen), a new processor is designated as the logical CPU 0, and the old (failed) processor no longer retains that designation. *Id.*, see also In my opinion, a POSITA would have been motivated to improve

the reliability of Kennedy's system by incorporating a change in system identifiers, as taught by Safford, to ensure that its operating system would no longer refer to failed processors as "logical CPU 0" when booting.

[1Di] changing a system identifier of the spare processor to that of the boot processor

137. I agree that Kennedy as modified by Safford teaches that "To improve reliability of processing assets, a computer system employs lock stepped processor cores that operate in a **master/check pair**." *Safford* (Ex. 1007) at 2:55-57. Safford further describes one pair of these processors "operating in a lock step mode" as initially "appear[ing] to the computer system…to be…a **logical CPU 0,**" but later that **logical CPU 0** designation is reassigned to another processor. *See Safford* (Ex. 1007) at 3:4-5, 4:52-53 ("The processor pair 125/127 then becomes the **logical CPU 0** in the computer system 100"). *Id.*, *see also* By changing the former standby processor 125/127 to the "logical CPU 0" processor, the system identifier of the spare processor (e.g., CPU 1) is changed to that of the boot processor (e.g., CPU 0). *Id.*, *see also* In light of the above citations, a POSITA would, in my opinion, have understood "**logical CPU 0**" to be the boot-processor designation used by an operating system to determine which of its processors is the boot processor and/or to distinguish its boot processor from its non-boot processors. *See* Moreover, in my opinion, a POSITA would have been motivated to improve the reliability of Kennedy's system by changing these system identifiers during failover, as taught by Safford, to ensure that its operating system would properly refer to the designated "logical CPU 0" when operating the processor to which failover occurred.

[1Dii] thereby switching the role of boot processor to a spare processor without shutting down the system's operating system.

138. I agree that Kennedy as modified by Nguyen and Safford teaches this limitation. Kennedy notes that an error can lead to switching the role of boot processor from one processor to another: "The architecture provides a method and apparatus to interface multiple processors to the backplane system bus such that upon start-up of the computer system, one default boot CPU is designated to boot the system, and if this CPU fails, **control transfers automatically to an alternate processor**." *Kennedy* (Ex. 1006) at 2:15-21. Moreover, Kennedy states that: "…provision needs to be made to determine which processor will boot the system, and if this "default" boot processor fails to boot the system,

an alternative processor, if one is installed, should take over boot operations to enhance the availability of the system." *Id.* at 7:5-13. As such, Kennedy as modified by Nguyen and Safford teaches an automatic switching of roles from default boot processor to alternative processor or vice versa.

139. In my opinion, to the extent that Kennedy is not interpreted to teach role switching of a boot processor to a spare processor without shutting down the system's operating system, Nguyen teaches switching from a processor to a spare processor without shutting down the system's operating system. In combination, Kennedy would be modified by Nguyen and Safford to include PAL-based implementations that do not require a system shutdown. Specifically, Kennedy, as modified by Nguyen and Safford, would switch from a boot processor to an alternative processor using a PAL-based, firmware implementation that would allow the switch between processors to be performed quickly enough to avoid trigging a timeout period that would otherwise be enforced by the operating system. Nguyen notes that "PAL-based implementations of the recovery mechanism should be able to complete quickly enough to avoid triggering a time-out period enforced by the operating system. *Nguyen* (Ex. 1004) ¶ [0045]." Nguyen states:

> As noted above, the recovery mechanism for errors detected within the FRC boundary may be handled by **various combinations of hardware, software and firmware modules**. One embodiment of the recovery mechanism employs code that is closely associated with the processor. For example, the Itanium® Processor Family of Intel® Corporation, employs a **layer of firmware** called the processor abstraction layer (PAL), which **provides an abstraction of the processor to the rest of the computing system**. Implementing recovery in the PAL **hides the recovery process from system level code**, such as the system abstraction layer (SAL), e.g. BIOS, **and the operating system**. PAL-based implementations of the recovery mechanism should be able to complete quickly enough to **avoid triggering a time-out period enforced by the operating system**.
>
> *Id.*

140. I agree that, in this example, implementing PAL-based firmware to implement the recovery mechanism quickly enough to avoid triggering a time-out period enforced by the operating system is equivalent to performing the recovery mechanism without shutting down the

system's operating system. Using a firmware-based PAL layer hides the recovery process from system level code such as the operating system. Changes in system level code would require an operating system shutdown to implement. *Id.* ¶ XX. PAL-based implementations, however, provide recovery without triggering a time-out period or other operations enforced by the operating system. This is equivalent to avoiding an operating system shutdown. Accordingly, the recovery mechanisms discussed in Nguyen are carried out in firmware, without shutting down the operating system. *Id.* ¶ XX. In my opinion, a POSITA would have been motivated to improve the efficiency of Kennedy's failover mechanism by implementing PAL-based firmware, as taught by Nguyen, in order to quickly fail over to an alternate boot processor in a seamless and prompt manner.

Claim 2: ***[2] wherein the spare processor is a hot spare that is held in idle until switched the role of boot processor***

141. In my opinion, to the extent that Nguyen and Kennedy do not explicitly teach a hot spare that is held in idle until switched to the role of boot processor, Nguyen and Kennedy, when combined with Safford, teach that the spare processor is a hot spare that is held in idle until switched the role of boot processor by disclosing a spare processor that is initially idled and becomes active upon a first processor unit losing lock step. *Safford* (Ex. 1007) at 2:26-32. Safford states: "The processor 125 may be designated as a "**hot standby**" and is **sitting idle** in lock step mode with the processor 127. Should one of the processors 111, 113, 121, and 123 suffer an error, the hot standby processors 125, 127 may be used to **speed recovery** from the resulting loss of lock step. *Id.* at 4:25-29). A POSITA would have combined Safford with Nguyen and Kennedy because each is concerned with providing processor failover mechanisms in high-availability environments, and because Nguyen and Kennedy would perform the failover more quickly when using a hot spare as taught by Safford. *Id.*

Claim 3: ***[3] wherein the spare processor is an active spare***

142. In my opinion, to the extent that Nguyen and Kennedy to not explicitly teach an active spare that is used as a spare processor, Nguyen and Kennedy, when combined with Safford, teach that the spare processor is an active spare by disclosing "moving the architected state comprises **copying the architected state directly** from the loss of lock step processor unit **to the spare processor unit**." *Safford* (Ex. 1007) at 5:23-26). A POSITA would have

combined Safford with Nguyen and Kennedy because each is concerned with providing processor failover mechanisms in high-availability environments, and because Nguyen and Kennedy would be able to perform additional processing when using an active spare, as taught by Safford. *Id.*

Claim 4: ***[4] wherein said switching further comprises: causing the system's operating system to idle the active spare***

143. In my opinion, to the extent that Nguyen and Kennedy to not explicitly teach causing the system's operating system to idle the active spare, Nguyen and Kennedy, when combined with Safford, teach that switching further includes causing the system's operating system to idle the active spare by disclosing a "means for moving an architected state of a processor unit having the loss of lock step initiating event to a **spare processor unit**, the spare processor unit **operating idle** in lock step." *Safford* (Ex. 1007) at 6:2-5). In my opinion, a POSITA would have combined Safford with Nguyen and Kennedy because each is concerned with providing processor failover mechanisms in high-availability environments, and because Nguyen and Kennedy would be able to reduce processing resource usage by idling the active spare, as taught by Safford. *Id.*

Claim 5: ***[5] wherein the spare processor is a processor module that includes a master and slave processors that operate in lockstep.***

144. I agree that Kennedy, in combination with Nguyen and Safford, teaches a multiprocessor system that includes boot processors and alternative processors. *Kennedy* (Ex. 1006) at 2:62-64 and 7:7-13. Kennedy, in combination with Nguyen and Safford, further discloses a pair of processors that operate in lockstep by disclosing that:

> In FRC mode, **one of execution cores 120 may be designated as the master**. The master refers to the execution core that is responsible for updating the resources shared by execution cores 120. **The other execution core 120 may be designated as the slave**. The slave is responsible for generating results from the same code sequence to be checked against those of the master."
>
> *Nguyen* (Ex. 1004) ¶ [0027].

145. These execution cores 120 "operate in lock step during normal FRC mode…" *Id.* ¶ [0024]. In my opinion, a POSITA would have combined Nguyen with Kennedy and Safford

because each is concerned with providing reliable processor failover mechanisms in high-availability environments, and because Kennedy and Safford would perform processor failovers more reliably when using a spare processor with master and slave execution cores as taught by Nguyen. *Id.*, *see also*

Claim 6: ***[6] wherein said determining that the processor for which said LOL is detected is assigned the role of boot processor comprises accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor.***

146. I agree that Kennedy, as modified by Nguyen and Safford, teaches accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor by disclosing that "The CPU1 120 has a corresponding **non-volatile memory 123 which contains configuration information and CPU specific initialize and self test code (ISTC) for the CPU1 120**..." *Kennedy* (Ex. 1006) at 5:15-18. Kennedy further indicates that: "The slot select logic 148 is described in more detail in reference to the CPU1 120 installed in slot 1 as the default boot processor and the CPU2 132 installed in slot 2 as the alternate boot processor. *Id.* at 10:55-58, see also Table 2. I agree that it would have been obvious to combine Kennedy with Nguyen and Safford to increase the dependability of the system by storing configuration information in non-volatile memory, as taught by Kennedy. *Id.*

Claim 7: ***[7] further comprising: determining the processor that is assigned the role of spare processor.***

147. I agree that Kennedy as modified by Nguyen and Safford teaches determining the processor that is assigned the role of spare processor by disclosing that "...provision needs to be made to **determine which processor will boot the system,** and if this "default" boot processor fails to boot the system, **an alternative processor**, if one is installed, **should take over boot operations** to enhance the availability of the system." *Kennedy* (Ex. 1006) at 7:5-13, *see also* Nguyen (Ex. 1004) at ¶ [0026]. In my opinion, it would have been obvious to a POSITA to combine Kennedy with Nguyen and Safford to increase the reliability of the system by identifying which processors are assigned certain roles, as taught by Kennedy. *Id.*

Claim 8: ***[8] wherein said determining the processor that is assigned the role of spare processor comprises: accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of spare processor.***

148. I agree that Kennedy as modified by Nguyen and Safford teaches accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor by disclosing that "The CPU1 120 has a corresponding **non-volatile memory 123 which contains configuration information and CPU specific initialize and self test code (ISTC) for the CPU1 120**..." *Kennedy* (Ex. 1006) at 5:15-18. Kennedy further indicates that: "The slot select logic 148 is described in more detail in reference to the CPU1 120 installed in slot 1 as the default boot processor and the CPU2 132 installed in slot 2 as the alternate boot processor. *Id.* at 10:55-58, *see also* Table 2. In my opinion, it would have been obvious to a POSITA to combine Kennedy with Nguyen and Safford to avoid system errors by identifying which processors are assigned certain roles, as taught by Kennedy. *Id.*, *see also*

Claim 9: ***[9] wherein said determining step is performed by system firmware.***

149. I agree that Kennedy as modified by Nguyen teaches determining whether an error has occurred for which recovery methods are to be used. *Kennedy,* (Ex. 1006) at 7:7-13. Kennedy as modified by Nguyen further teaches that a firmware-based processor abstraction layer is usable in such a recovery process. *Nguyen* (Ex. 1004) ¶ [0045]. In recovery scenarios, Kennedy as modified by Nguyen implements a firmware-based processor abstraction layer (PAL), which "**provides an abstraction of the processor** to the rest of the computing system." *Id.* "**Implementing recovery in the PAL** hides the recovery process from system level code, such as the system abstraction layer (SAL), e.g. BIOS, and the operating system." *Id.* As such, Kennedy as modified by Nguyen and Safford indicates that the process steps involved in its recovery processes, including determining steps, is performed by firmware.

Claim 10: ***[10] wherein said switching step is performed by system firmware.***

150. I agree that Kennedy as modified by Nguyen teaches determining whether an error has occurred for which recovery methods are to be used. *Kennedy,* (Ex. 1006) at 7:7-13 Kennedy as modified by Nguyen further teaches that the switching step is performed by

system firmware by implementing a firmware-based processor abstraction layer in the recovery process. *Nguyen* (Ex. 1004) ¶ [0045]. In recovery scenarios, Nguyen implements a firmware-based processor abstraction layer (PAL), which "provides an abstraction of the processor to the rest of the computing system." *Id.* "**Implementing recovery in the PAL** hides the recovery process from system level code, such as the system abstraction layer (SAL), e.g. BIOS, and the operating system." *Id.* As such, Kennedy as modified by Nguyen and Safford indicates that the process steps involved in its recovery processes, including switching steps, is performed by firmware.

Claim 11: ***[11] wherein the processor for which said LOL is detected includes a lockstep pair of processors, and wherein said switching comprises: determining one of the lockstep pair of processors that is not the cause of said LOL; and copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to said spare processor.***

151. I agree that Kennedy as modified by Nguyen and Safford teaches determining one of the lockstep pair of processors that is not the cause of said LOL and copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to said spare processor by disclosing, "the node controller 130 may store the architected state of the processors 111, 113, 121, 123, 125, and 127, either internally in the node controller 130, or in another component of the computer system 100." *Safford*, (Ex. 1007) at 4:10-14. While nodes 111 and 113 are indicated as having potentially lost lockstep, nodes 121, 123, 125, and 127 have not lost lockstep and, according to Safford, the node controller 130 is capable of copying and storing the state information of those processors that have not caused a loss of lockstep. *Id.* at 4:6-18. In my opinion, a POSITA would have been motivated to combine Safford's copying of state information for non-failed processors (i.e., processors that have not lost lockstep) in order to further bolster the reliability of its high-availability system through the copying of processor state information for non-failed, functioning nodes.

Claim 12: ***[12] wherein said spare processor includes a lockstep pair of processors, and wherein said copying comprises: copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to each of said lockstep pair of processors of said spare processor.***

152. I agree that Kennedy as modified by Nguyen and Safford teaches copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to each of said lockstep pair of processors of said spare processor by disclosing, "the node controller 130 may store the architected state of the processors 111, 113, 121, 123, 125, and 127, either internally in the node controller 130, or in another component of the computer system 100." *Safford*, (Ex. 1007) at 4:10-14. While nodes 111 and 113 are indicated as having potentially lost lockstep, nodes 121, 123, 125, and 127 have not lost lockstep and, according to Safford, the node controller 130 is capable of copying and storing the state information of those processors that have not caused a loss of lockstep. *Id.* at 4:6-18. Safford specifically notes that the state information is copied to each processor in the pair of processors (e.g., to processor pairs 121/123 or 125/127). *Id.* In my opinion, a POSITA would have been motivated to combine Safford's copying of state information for non-failed (working) nodes in order to further increase the reliability of its high-availability system by keeping a store of state information copied from working nodes.

Claim 13: ***[13P] A method comprising:***

153. *See* Limitation [1A].

[13A] system firmware determining that loss of lockstep (LOL) is detected for a lockstep pair of processors that are assigned the role of boot processor in a system

154. *See* Limitation [1A].

[13B] determining one of said lockstep pair of processors that is not the cause of said LOL

155. I agree that Kennedy as modified by Nguyen and Safford teaches determining one of said lockstep pair of processors that is not the cause of said LOL by disclosing detecting an error that is localized to one of the execution cores. Kennedy as modified by Nguyen and Safford states: "As long as the error is detected before it triggers an FRC reset, the corrupted data can be localized to one of the execution cores, leaving the **machine state data** of the other execution core **available for recovery**." *Nguyen* (Ex. 1004) ¶ [0046]).

[13C] copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to a spare processor

156. I agree that Kennedy as modified by Nguyen and Safford teaches copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to a spare processor by disclosing "the node controller 130 may store the architected state of the processors 111, 113, 121, 123, 125, and 127, either internally in the node controller 130, or in another component of the computer system 100." *Safford*, (Ex. 1007) at 4:10-14. While nodes 111 and 113 are indicated as having potentially lost lockstep, nodes 121, 123, 125, and 127 have not lost lockstep and, according to Safford, the node controller 130 is capable of copying and storing the state information of those processors that have not caused a loss of lockstep. *Id.* at 4:6-18. In my opinion, a POSITA would be motivated to combine Safford's copying of state information for non-failed, working nodes in order to further increase the reliability of its high-availability system through the storing of processor state information copied from working nodes.

[13Di] changing a system identifier, used by the system's operating system to recognize the boot processor, of said lockstep pair of processors that are assigned the role of boot processor, to a different value

157. *See* Limitation [1Ci].

[13Dii] which causes the system's operating system to not recognize the lock step pair of processors for which said LOL is detected as the processor

158. *See* Limitation [1Cii].

[13E] changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to said spare processor.

159. *See* Limitation [1D1].

Claim 14: (I agree that Claim 14 recites substantially the same subject matter as Claim 2 ([2]) and is therefore obvious for the same reasons as Claim 2 above).

Claim 15: (I agree that Claim 15 recites substantially the same subject matter as Claim 3 ([3]) and is therefore obvious for the same reasons as Claim 3 above).

Claim 16: (I agree that Claim 16 recites substantially the same subject matter as Claim 4 ([4]) and is therefore obvious for the same reasons as Claim 4 above).

Claim 17: (I agree that Claim 17 recites substantially the same subject matter as Claim 1 ([1Dii]) and is therefore obvious for the same reasons as Claim 1 above).

Claim 18: (I agree that Claim 18 recites substantially the same subject matter as Claim 9 and is therefore obvious for the same reasons as Claim 9 above).

Claim 19: (I agree that Claim 19 recites substantially the same subject matter as Claim 11 and is therefore obvious for the same reasons as Claim 11 above).

Claim 20: (I agree that Claim 20 recites substantially the same subject matter as Claim 10 and is therefore obvious for the same reasons as Claim 10 above).

Claim 21: (I agree that Claim 21 recites substantially the same subject matter as Claim 1 and is therefore obvious for the same reasons as Claim 1 above).

Claim 22: ***[22] wherein the another processor module is a processor module assigned a role of spare processor.***

160. I agree that Kennedy as modified by Nguyen and Safford teaches the other processor module is a processor module assigned a role of spare processor by disclosing that "…provision needs to be made to **determine which processor will boot the system**, and if this "default" boot processor **fails to boot the system, an alternative processor**, if one is installed, **should take over boot operations** to enhance the availability of the system." *Kennedy* (Ex. 1006) at 7:5-13. To the extent that Kennedy is not interpreted to disclose a processor module, Nguyen discloses a processor 110 that includes execution cores 120(a) and 120(b), which a POSITA would recognize as a processor module. *Nguyen* (Ex. 1004) ¶ [0027], *see also* In my opinion, it would have been obvious to a POSITA to combine Kennedy with Nguyen and Safford to reduce system errors by assigning processor roles, as taught by Kennedy, and by changing system identifiers for those processors, as taught by Safford. *Id.*, *see also*

Claim 23: (I agree that Claim 23 recites substantially the same subject matter as Claims 2 and 14 ([2] and [14]) and is therefore obvious for the same reasons as Claims 2 and 14 above).

Claim 24: (I agree that Claim 24 recites substantially the same subject matter as Claims 2 and 14 ([2] and [14]) and is therefore obvious for the same reasons as Claims 2 and 14 above).

Claim 25: (I agree that Claim 25 recites substantially the same subject matter as Claim 8 and is therefore obvious for the same reasons as Claim 8 above).

Claim 26: (I agree that Claim 26 recites substantially the same subject matter as Claim 1 and is therefore obvious for the same reasons as Claim 1 above).

Claim 27: ***[27] wherein said processor in said multi-processor system is an Itanium Processor Family (IPF) processor.***

161. I agree that Kennedy in combination with Nguyen and Safford teaches a multiprocessor system that includes boot and alternative processors. *Kennedy* (Ex. 1006) at 2:13-15. Kennedy in combination with Nguyen and Safford further teaches that a processor in such a multi-processor system is an Itanium Processor Family (IPF) processor by specifically disclosing use of an Itanium processor in its system. *Nguyen* (Ex. 1004) ¶ [0045]. Kennedy in combination with Nguyen specifically mentions that Itanium processors are used in its system: "For example, the **Itanium(R) Processor Family** of Intel(R) Corporation, employs a layer of firmware called the processor abstraction layer (PAL), which provides an abstraction of the processor to the rest of the computing System." *Id.* In view of the above citations, Kennedy as modified by Nguyen and Safford teaches that the processors would have been IPF processors. *See* In my opinion, a POSITA would have been motivated to combine these disclosures from Kennedy, Nguyen, and Safford to improve the spatial efficiency of the system by consolidating the processors into the multicore processor package used by the IPF. *Id.* In my opinion, it was well within the skill of a POSITA to modify Kennedy's system by Nguyen and Safford in this manner, and thus this combination of Kennedy, Nguyen, and Safford would have been expected to succeed. *Id.* Accordingly, Kennedy as modified by Nguyen and Safford teaches this claim limitation.

H. Ground 4 (Safford in view of Cepulis) Presents Substantial New Questions of Patentability

162. It is my opinion that Safford in view of Cepulis, together with the knowledge of a POSITA, discloses or renders obvious claims 1-4 of the '302 Patent. In addition to the opinions expressed in this Declaration, I have reviewed and agree with the corresponding obviousness contentions set forth in the Request. I have also reviewed and assisted in the

preparation of Appendices A-D. Appendix D provides a detailed mapping of Nguyen in view of Kennedy to claims 1-4 that accurately reflects my opinions regarding obviousness

163. An overview of Safford is provided in the Request. I have reviewed and agree with that overview of Safford.

164. In my opinion, Safford is analogous art to the '302 Patent because Nguyen is both from the same field of endeavor (e.g., providing recovery from failure in a high-availability computing system) as the '302 Patent and is reasonably pertinent to a problem faced by the inventors of the '302 Patent (e.g., system crashes caused by loss of lockstep). *See Safford* (Ex. 1007) at 4:52-54; *see also '302 Patent* (Ex. 1001) at 3:16-18.

165. An overview of Cepulis is provided in the Request. I have reviewed and agree with that overview of Cepulis.

166. In my opinion, Cepulis is analogous art to the '302 Patent because Cepulis is from the same field of endeavor (e.g., providing recovery from failure in a high-availability computing system) and is reasonably pertinent to a problem faced by the inventors of the '302 Patent (e.g., detecting and/or recovering from processor failures in multiprocessor systems). *See* Ex. 1014, Abstract; *see also* Ex. 1001 at 4:34-53, 3:16-18.

1. Motivation to Combine Safford and Cepulis

167. I agree that the combination of Safford in light of Cepulis' explanation of the meaning of the "logical CPU 0" boot-processor designator renders obvious, each and every element of claims 9, 10, 11, 12, and 23-25, and therefore presents a substantial new question of patentability. A POSITA would have considered the combined teachings and explanations of these references important in determining whether claims 9, 10, 11, 12, and 23-25 are patentable because Safford, in light of Cepulis' explanation of the meaning of the "logical CPU 0" boot-processor designator, was not expressly considered or discussed during prosecution. Exhibit A shows in greater detail how Safford in light of Cepulis' explanation renders obvious each and every element of claims 9, 10, 11, 12, and 23-25.

Claim 1: ***[1P] A method comprising:***

168. In my opinion, to the extent the preamble is limiting, Safford discloses a "method . . . used to enhance recovery from loss of lock step in a multi-processor computer system." Safford (Ex. 1007), Abstract.

[1A] detecting loss of lockstep (LOL) for a processor in a multi-processor system

169. I agree that Safford discloses, or at least renders obvious, this limitation by disclosing (1) two "processors [that] may operate . . . in a lock step mode" and (2) associated "error detection and signaling logic" that are configure to "detect an error that will cause a loss of lock step." Safford (Ex. 1007) at 3:1-45. Additionally, Safford discloses "[detecting] a loss of lock step. . . [by comparing] outputs from [two] microprocessor cores" and detecting "[a] difference in processing" that "is by definition a loss of lock step." Id. at 1:51-55. Safford further discloses "detecting a loss of lock step initiating event in a processor." Id., claim 13.

170. More specifically, I agree that Safford discloses that "the processor 111 detects an error event that indicates an impending loss of lock step. In block 215, the processor 111 signals the node controller 130 that the first processor pair 111/113 has broken lock step and that the first processor pair 111/113 should be taken 'off-line.'" *Safford* (Ex. 1007) at 4:35-40.

[1B] determining that the processor for which said LOL is detected is assigned a role of boot processor

171. I agree that Safford discloses, or at least renders obvious, this limitation by disclosing a pair of processors "operating in a lock step mode" that initially "appear to [a] computer system . . . to be . . . a logical CPU 0" but, later, have the logical CPU 0 designation reassigned to another processor. See *Safford* (Ex. 1007) at 3:4-5, 4:52-53 ("The processor pair 125/127 then becomes the logical CPU 0 in the computer system 100").

172. In my opinion, a POSITA would have recognized "logical CPU 0" as being the designation used by computing system 100 to identify its boot processor for at least the reason that, in multi-processor system architectures like that described in Safford, "logical CPU 0" is commonly used to indicate that a processor has been selected as a boot processor. See *Cepulis* (Ex. 1014) at 2:52-64.

173. I agree that a POSITA would have also recognized "CPU 0" and "CPU 1" as being physical designations used by computing system 100 to identify processors 111 and 113. The use of both logical and physical designators by a computing system is generally necessary when the role of boot processor can be assigned and reassigned to a processor regardless of its physical position within a system. *See Safford* (Ex. 1007) at 4:52-53 ("The processor pair 125/127 then becomes the logical CPU 0 in the computer system 100").

174. Thus, in my opinion, a POSITA would have understood "logical CPU 0" to be a boot-processor designation used by an operating system to determine which of its processors is a boot processor and/or to distinguish its boot processor from its non-boot processors. As such, determining if a processor is assigned the role of boot processor is performed by the operating system through recognition of the boot processor designation. Accordingly, Safford discloses, or at least renders obvious, determining if a processor is assigned the role of boot processor by disclosing a pair of processors "operating in a lock step mode" that initially "appear to [a] computer system . . . to be . . . a logical CPU 0" but, later, have the logical CPU 0 designation reassigned to another processor. See *Safford* (Ex. 1007) at 3:4-5, 4:52-53 ("The processor pair 125/127 then becomes the logical CPU 0 in the computer system 100").

[1C] changing a system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected, to a different value which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor

175. I agree that Safford discloses, or at least renders obvious, this limitation by disclosing a pair of processors "operating in a lock step mode" that initially "appear to [a] computer system . . . to be . . . a logical CPU 0" but, later, have the logical CPU 0 designation reassigned to another processor. See Safford (Ex. 1007) at 3:4-5, 4:52-53 ("The processor pair 125/127 then becomes the logical CPU 0 in the computer system 100"). In my opinion, a POSITA would have recognized "logical CPU 0" as being the designation used by computing system 100 to identify its boot processor for at least the reason that, in multi-processor system architectures like that described in Safford, "logical CPU 0" is commonly used to indicate that a processor has been selected as a boot processor. *See Cepulis* (Ex. 1014) at 2:52-64.

176. In my opinion, Safford indicates that, after a first pair of processors has been taken offline, another pair of processors assumes the designation or identifier of CPU0, causing the operating system to no longer recognize processor pair 111 and 113 as the boot processor (i.e., CPU0):

> Alternatively, the node controller 130 may copy the state of the first processor pair 111/113 directly from the processors 111 and 113 to the processors 125 and 127

without any intermediate storage of the architected state in the node controller 130, or other component of the computer system 100. The processor pair 125/127 then becomes the logical CPU 0 in the computer system 100, and the computer system 100 operates without a hot standby processor pair.

Safford (Ex. 1007) at 4:47-54.

[1D] changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to a spare processor without shutting down the system's operating system

177. In my opinion, Safford indicates that, after a first pair of processors has been taken offline, another pair of processors assumes the designation or identifier of CPU0, thereby causing the operating system to recognize a new processor pair 125/127 as the boot processor (i.e., CPU0):

> Alternatively, the node controller 130 may copy the state of the first processor pair 111/113 directly from the processors 111 and 113 to the processors 125 and 127 without any intermediate storage of the architected state in the node controller 130, or other component of the computer system 100. The processor pair 125/127 then becomes the logical CPU 0 in the computer system 100, and the computer system 100 operates without a hot standby processor pair.

Safford (Ex. 1007) at 4:47-54.

178. In my opinion, to the extent that Safford in view of Cepulis does not disclose an operating system, a POSITA would recognize that "each processor [would have] its own memory and [would run] its own copy of the operating system." Commercial Fault Tolerance (Ex. 1010), section 4.2. Moreover, to the extent that Safford in view of Cepulis does not explicitly state that the transition from boot to spare processor is performed without shutting down the operating system, a POSITA would recognize that the intent of Safford's recovery system is to recover from loss of lockstep without crashing or restarting the operating system.

<u>Claim 2:</u> [2] wherein the spare processor is a hot spare that is held in idle until switched the role of boot processor.

179. I agree that Safford states: "The processor 125 may be designated as a "hot standby" and is sitting idle in lock step mode with the processor 127. Should one of the processors 111,

113, 121, and 123 suffer an error, the hot standby processors 125, 127 may be used to speed recovery from the resulting loss of lock step. Safford (Ex. 1007) at 4:25-29).

Claim 3: ***[3] wherein the spare processor is an active spare.***

180. I agree that Safford teaches that the spare processor is an active spare by disclosing "moving the architected state comprises copying the architected state directly from the loss of lock step processor unit to the spare processor unit." Id. at 5:23-26).

Claim 4: ***[4] wherein said switching further comprises: causing the system's operating system to idle the active spare.***

181. I agree that Safford teaches that switching further includes causing the system's operating system to idle the active spare by disclosing a "means for moving an architected state of a processor unit having the loss of lock step initiating event to a spare processor unit, the spare processor unit operating idle in lock step." Id. at 6:2-5

I. Secondary Considerations

It is my opinion that the Challenged Claims of the '302 Patent are unpatentable as obvious in view of the prior art references. I have considered the secondary considerations that I am aware of at this time and determined that they do not impact my opinions.

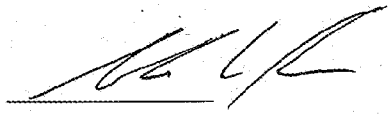
J. Declaration in Lieu of Oath

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under § 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon. If called to testify as to the truth of the matters stated herein, I could and would testify competently.

I declare under penalty of perjury that the foregoing is true and correct.

Dated: August 30, 2023

Respectfully submitted,



Andrew Wolfe, Ph.D.

US007191328B2

(12) **United States Patent**
Hobson

(10) **Patent No.: US 7,191,328 B2**
(45) **Date of Patent: Mar. 13, 2007**

(54) **SYSTEM AND METHOD FOR USING AN EXTENSIBLE FIRMWARE INTERFACE (EFI) UTILITY TO BUILD AN EFI LAYER BETWEEN AN OPERATING SYSTEM AND A LEGACY BASIC INPUT/OUTPUT SYSTEM DURING A BOOT PROCESS**

(75) Inventor: **Louis B. Hobson**, Tomball, TX (US)

(73) Assignee: **Hewlett-Packard Development Company, L.P.**, Houston, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 483 days.

(21) Appl. No.: **10/894,808**

(22) Filed: **Jul. 20, 2004**

(65) **Prior Publication Data**

US 2006/0020780 A1 Jan. 26, 2006

(51) **Int. Cl.**
G06F 15/177 (2006.01)
G06F 9/00 (2006.01)

(52) **U.S. Cl.** **713/2**; 713/1; 710/1

(58) **Field of Classification Search** 713/1, 713/2; 710/1–74
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

7,080,243	B2 *	7/2006	Ramiz et al.	713/1
2005/0071618	A1 *	3/2005	Natu	713/1
2005/0204357	A1 *	9/2005	Garg et al.	718/1
2006/0047941	A1 *	3/2006	Lai	713/2

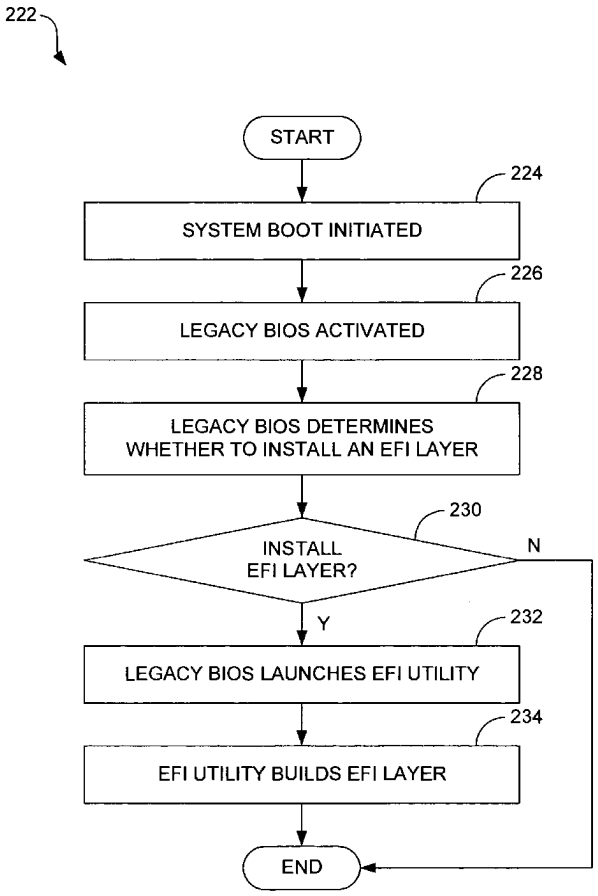
* cited by examiner

Primary Examiner—Thomas Lee
Assistant Examiner—Anand B. Patel

(57) **ABSTRACT**

Disclosed is a system and a method for implementing an extensible firmware interface (EFI). In one embodiment, a system and a method pertain to launching an EFI utility during a boot process of a computer system, and building an EFI layer that resides between an operating system of the computer system and a legacy basic input/output system (BIOS) of the computer system using the EFI utility.

20 Claims, 5 Drawing Sheets



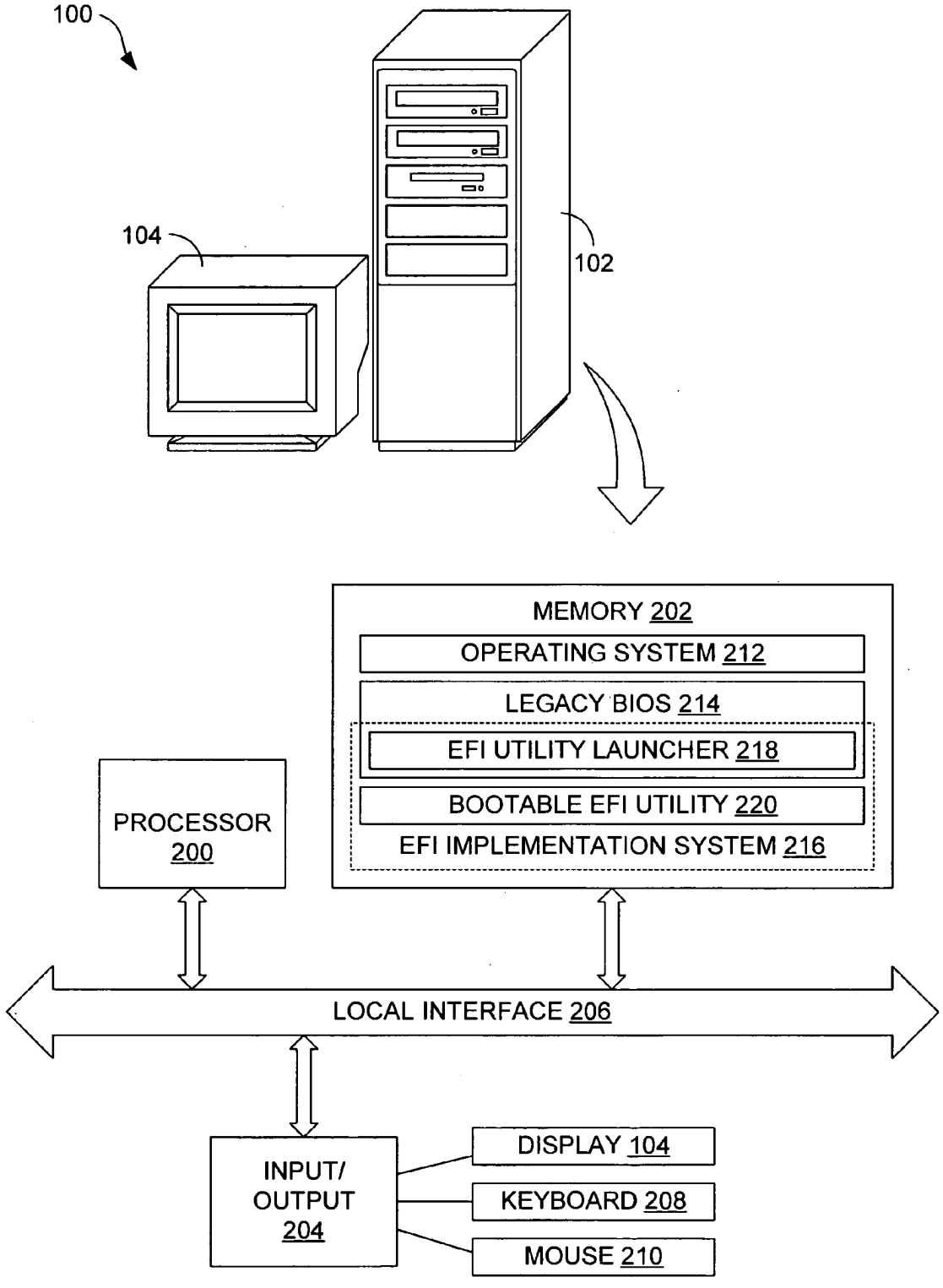

100
104
102
MEMORY 202
OPERATING SYSTEM 212
LEGACY BIOS 214
EFI UTILITY LAUNCHER 218
BOOTABLE EFI UTILITY 220
EFI IMPLEMENTATION SYSTEM 216
PROCESSOR 200
LOCAL INTERFACE 206
INPUT/ OUTPUT 204
DISPLAY 104
KEYBOARD 208
MOUSE 210

FIG. 1

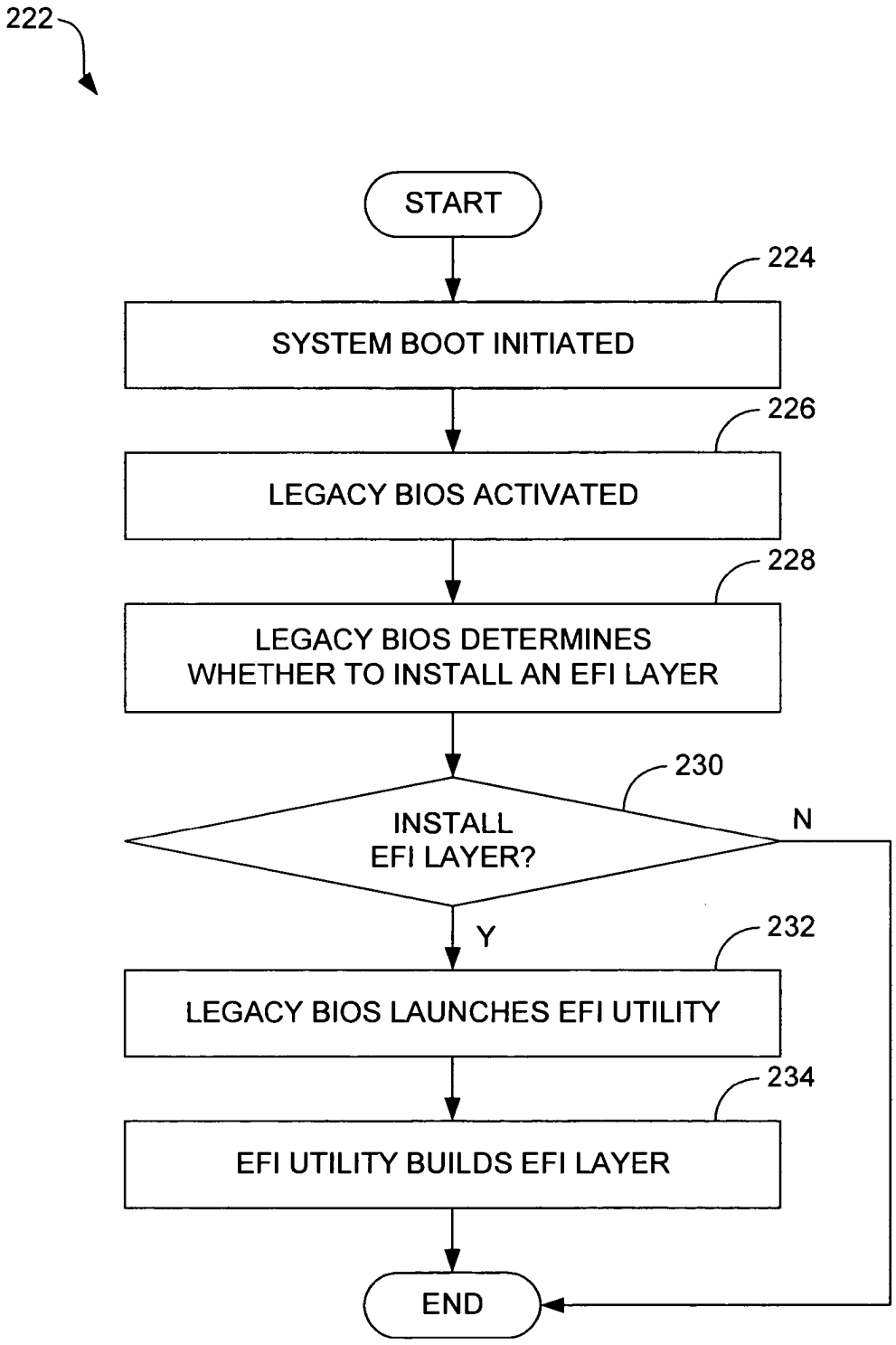

222
START
224
SYSTEM BOOT INITIATED
226
LEGACY BIOS ACTIVATED
228
LEGACY BIOS DETERMINES WHETHER TO INSTALL AN EFI LAYER
230
INSTALL EFI LAYER?
N
Y
232
LEGACY BIOS LAUNCHES EFI UTILITY
234
EFI UTILITY BUILDS EFI LAYER
END

FIG. 2

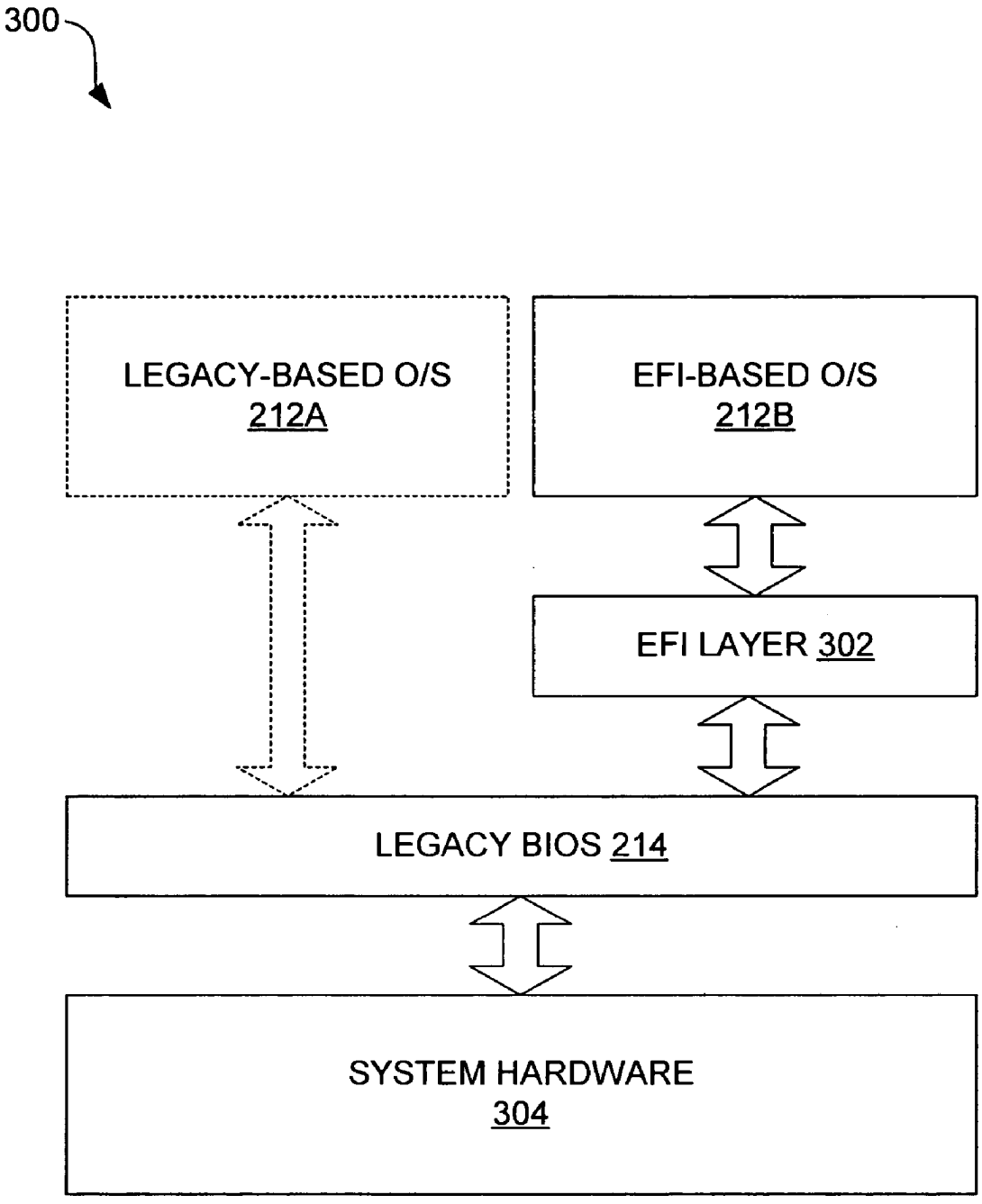

300
LEGACY-BASED O/S
212A
EFI-BASED O/S
212B
EFI LAYER 302
LEGACY BIOS 214
SYSTEM HARDWARE
304

FIG. 3

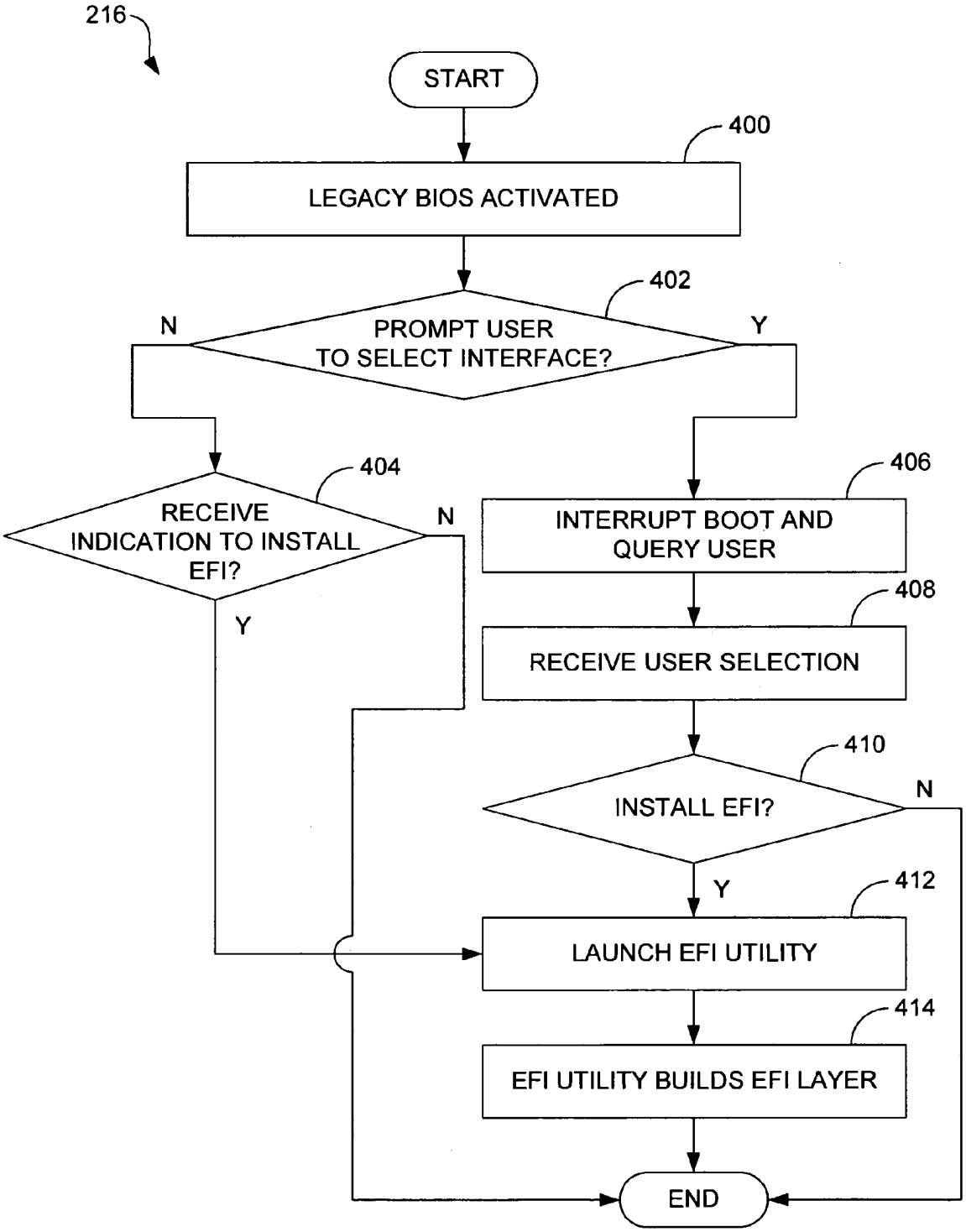

216
START
400
LEGACY BIOS ACTIVATED
402
PROMPT USER TO SELECT INTERFACE?
N
Y
404
RECEIVE INDICATION TO INSTALL EFI?
N
Y
406
INTERRUPT BOOT AND QUERY USER
408
RECEIVE USER SELECTION
410
INSTALL EFI?
N
Y
412
LAUNCH EFI UTILITY
414
EFI UTILITY BUILDS EFI LAYER
END

FIG. 4

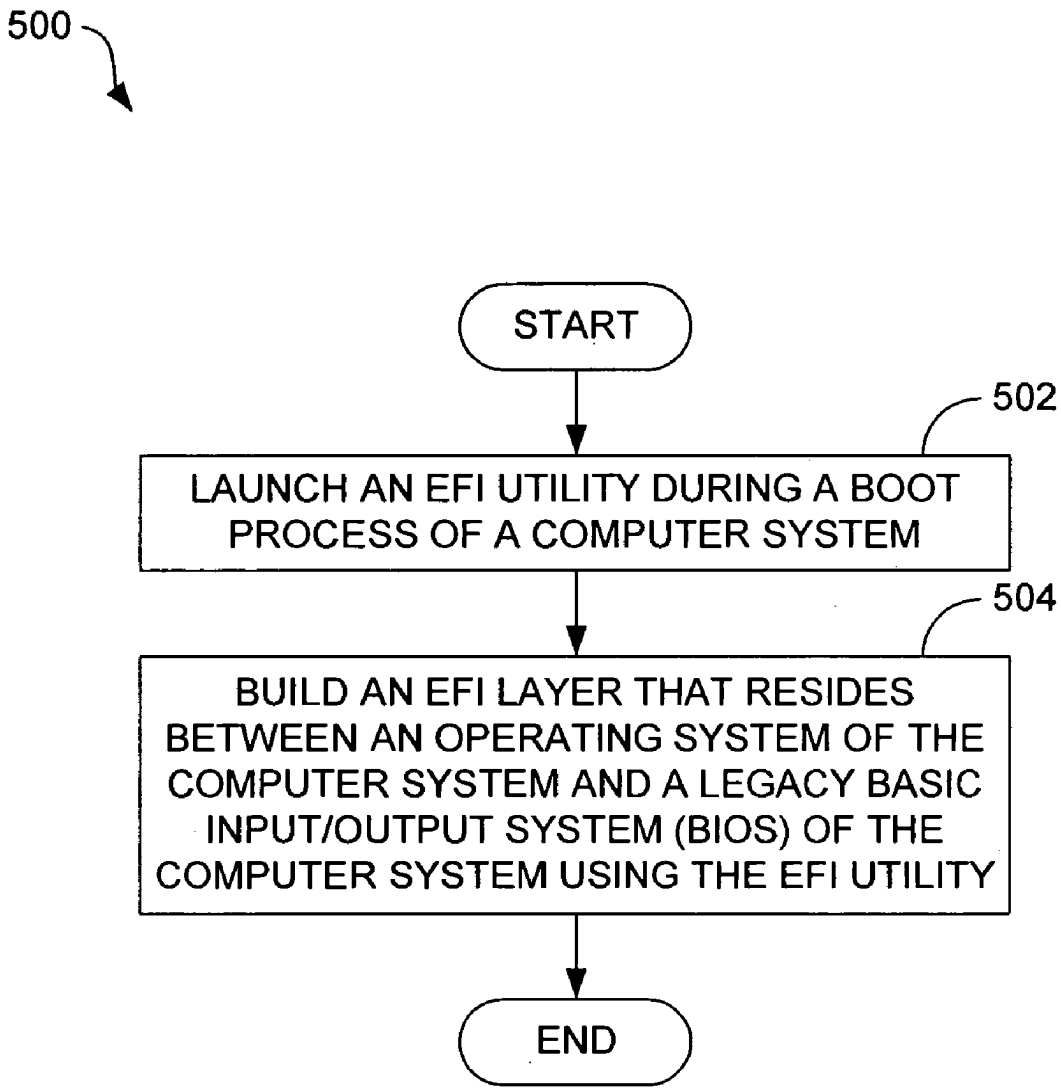

500
START
502
LAUNCH AN EFI UTILITY DURING A BOOT PROCESS OF A COMPUTER SYSTEM
504
BUILD AN EFI LAYER THAT RESIDES BETWEEN AN OPERATING SYSTEM OF THE COMPUTER SYSTEM AND A LEGACY BASIC INPUT/OUTPUT SYSTEM (BIOS) OF THE COMPUTER SYSTEM USING THE EFI UTILITY
END

FIG. 5

SYSTEM AND METHOD FOR USING AN EXTENSIBLE FIRMWARE INTERFACE (EFI) UTILITY TO BUILD AN EFI LAYER BETWEEN AN OPERATING SYSTEM AND A LEGACY BASIC INPUT/OUTPUT SYSTEM DURING A BOOT PROCESS

BACKGROUND

For years, personal computers (PCs) have used the so-called legacy basic input/output system (BIOS) to facilitate booting and to provide an interface between the computer operating system and various input/output (I/O) devices such as displays, keyboards, mice, and disk drives. Over the past several years, however, the Microsoft Corporation has been developing a new software interface, called the extensible firmware interface (EFI), that is expected to replace the legacy BIOS. The EFI provides a new model for the interface between operating systems and platform firmware and comprises data tables that contain platform-related information and boot and runtime service calls that are available to the operating system and its loader. In order to communicate with the EFI, the operating system must be specifically configured for use with the EFI, or other appropriate software must be developed to enable such communication.

Given that it appears that the PC industry will be transitioning to EFI-based systems, it is likely that future PCs will comprise the EFI. At least for the next few years, however, there will be many persons that will still use legacy-based operating systems that cannot communicate with the EFI. Accordingly, needed is an interim solution that supports both legacy BIOS and EFI so as to enable the use of either a legacy-based operating system or an EFI-based operating system.

Although both interface systems could be supported by installing both the legacy BIOS and the EFI in the system memory so that either is available for use, such a solution typically would require a larger system memory device (e.g., flash chip) to store both interface systems. Although such memory devices are available, their use would undesirably increase the cost of the PCs in which they are installed.

The Intel Corporation has proposed an alternative solution. Specifically, proposed is a backward-compatible solution in which the EFI layer is configured to emulate certain aspects of legacy BIOS operation so as to enable interface with legacy-based operating systems. In that solution, a compatibility support module (CSM) comprising a series of drivers chooses a legacy BIOS or EFI booting route depending upon the boot device(s) that is/are selected. Although this solution avoids the memory space problem identified above, such emulation is relatively complex and therefore requires significant time and resources to develop.

SUMMARY OF THE DISCLOSURE

Disclosed is a system and a method for implementing an extensible firmware interface (EFI). In one embodiment, a system and a method pertain to launching an EFI utility during a boot process of a computer system, and building an EFI layer that resides between an operating system of the computer system and a legacy basic input/output system (BIOS) of the computer system using the EFI utility.

BRIEF DESCRIPTION OF THE DRAWINGS

The disclosed system and method can be better understood with reference to the following drawings. The components in the drawings are not necessarily to scale.

FIG. **1** is a schematic and block diagram of a computer system that incorporates a system for implementing an extensible firmware interface.

FIG. **2** is a flow diagram that illustrates an embodiment of a method for implementing an extensible firmware interface.

FIG. **3** is a block diagram representing a system configuration that results when an extensible firmware interface is implemented.

FIG. **4** is a flow diagram that illustrates an embodiment of operation of the EFI implementation system shown in FIG. **1**.

FIG. **5** is a flow diagram that illustrates a further embodiment of a method for implementing an extensible firmware interface.

DETAILED DESCRIPTION

Disclosed herein are embodiments of a system and method for implementing an extensible firmware interface (EFI). More particularly, disclosed are a system and method that comprise a legacy BIOS interface that is configured to, if desired and/or necessary, launch an EFI utility that installs an EFI layer between the BIOS and an EFI-based operating system during the boot process. With such operation, the system supports both legacy BIOS operation and EFI operation.

Referring now to the drawings, in which like numerals indicate corresponding parts throughout the several views, FIG. **1** illustrates a computer system **100** that incorporates a system for implementing an EFI and, more particularly, supporting both legacy BIOS and EFI operation. The computer system **100** comprises a central computing unit (CPU) **102** a display **104**. Also illustrated in FIG. **1** is an example architecture for the CPU **102**. In the illustrated embodiment, the CPU includes a processor **200**, memory **202**, and input/output components **204**, each of which is connected to a local interface **206**.

The processor **200** can include one or more microprocessors that control the logic of the digital devices of the computer system **100**. The memory **202** includes any one of or a combination of volatile memory elements (e.g., RAM) and nonvolatile memory elements (e.g., read only memory (ROM), flash memory, hard disk, etc.).

The I/O components **204** comprise the components with which inputs and outputs can be exchanged between the CPU **102** and a user input/output device such as the display **104**, a keyboard **208**, and a mouse **210**. In addition, the components **204** may, for example, comprise various ports, a modem, wireless transceiver, a telephonic interface, and a network card that enable communication with other devices.

The memory **202** comprises various programs including an operating system **212**, a legacy BIOS **214**, and an EFI implementation system **216**. The operating system **212** controls the execution of various computer system software and provides scheduling, input-output control, file and data management, memory management, and communication control and related services. As is described in the following, the operating system **212** may comprise a legacy-based operating system or an EFI-based operating system depending upon the particular implementation. Indeed, the CPU **102** may, in some embodiments, comprise both a legacy-based operating system and an EFI-based operating system.

The EFI implementation system **216** comprises the various programs that are used to enable interface with an EFI-based operating system. As is indicated in FIG. **1**, the EFI implementation system **216** comprises an EFI launcher **218** that is comprised by the legacy BIOS **214**, and a

bootable EFI utility **220** that is separate from the legacy BIOS **214**. By way of example, the legacy BIOS **214** resides within a non-volatile memory device provided on the system circuit board of the CPU **102**. The memory device may comprise a flash-based memory chip or an electrically-erasable, programmable read only memory (EEPROM). The memory device may be relatively small, for instance having a capacity of about 512 kilobytes (KB) so as to lower the cost of the device, and the CPU **102** in which it is provided.

The bootable EFI utility **220** is a utility that, when launched, builds an EFI as an interface layer between an operating system and a legacy BIOS. By way of example, the EFI utility **220** may comprise a publicly-available utility such as the EFI sample implementation source code (version 1.10.14.62) that is currently available from the Intel Corporation. Although use of that utility may be desirable due to its ready availability, other (e.g., later designed) versions of that utility or a custom-designed utility that is configured to build an EFI layer between an operating system and a legacy BIOS could be used instead.

Irrespective of its particular configuration, the EFI utility **220** resides outside of the legacy BIOS **214**. With such a configuration, the legacy BIOS **214** may be installed on the aforementioned memory device without the need to increase the size of that device. By way of example, the EFI utility **220** resides within a host protected area (HPA) of a hard disk of the CPU **102**. As is known, the HPA is an area of a computer hard disk that is available to the system BIOS but which is unknown and inaccessible to the operating system. In alternative arrangements, the EFI utility **220** may reside in another location, for instance on a bootable removable disk, such as a floppy disk, CD or DVD, that is inserted into a disk drive of the CPU **102**.

Various programs (logic) have been described above. These programs can be stored on any computer-readable medium for use by or in connection with any computer-related system or method. In the context of this disclosure, a computer-readable medium is an electronic, magnetic, optical, or other physical device or means that contains or stores a computer program for use by or in connection with a computer-related system or method. Programs can be embodied in any computer-readable medium for use by or in connection with an instruction execution system, apparatus, or device, such as a computer-based system, processor-containing system, or other system that can fetch the instructions from the instruction execution system, apparatus, or device and execute the instructions.

FIG. **2** is a flow diagram that describes a method **222** for implementing an EFI. Process steps or blocks in the flow diagrams of this disclosure may represent modules, segments, or portions of code that include one or more executable instructions for implementing specific logical functions or steps in the process. Although particular example process steps are described, alternative implementations are feasible. Moreover, steps may be executed out of order from that shown or discussed, including substantially concurrently or in reverse order, depending on the functionality involved.

Beginning with block **224** of FIG. **2**, a computer system boot is initiated, for instance when the user turns on the power of the computer system or resets the computer system. As a result of the boot initiation, the legacy BIOS is activated, as indicated in block **226**. After that activation, the legacy BIOS determines whether to install an EFI layer, as indicated in block **228**. That determination can be input in various ways. By way of example, the determination is made in relation to a selection input by the user. For instance, if the computer system comprises an EFI-based operating system, the user may elect to install the EFI layer.

Referring next to decision block **230**, flow from this point depends upon whether an EFI layer is to be installed. If not, flow for the EFI implementation portion of the boot process is terminated. In such a case, the legacy BIOS will complete the boot process and will operate as an interface between a legacy-based operating system and the various hardware of the computer system as is currently conventional. If, on the other hand, an EFI layer is to be installed, flow continues to block **232** at which the legacy BIOS launches the EFI utility that, for example, resides in the HPA. When the EFI utility is launched, the EFI utility takes control over the boot process and, as indicated in block **234**, builds an EFI layer between the operating system (presumably an EFI-based operating system) and the legacy BIOS. In such a case, the operating system will communicate with the EFI layer, but the legacy BIOS will still be available for communicating with the computer system hardware.

FIG. **3** illustrates an example implementation of an EFI. More particularly, FIG. **3** illustrates the system configuration that can result when the legacy BIOS launches the EFI utility as described above in relation to FIG. **2**. As is indicated in FIG. **3**, the system **300** comprises one or both of a legacy-based operating system **212**A and an EFI-based operating system **212**B. In cases in which the legacy-based operating system **212**A is present and is to be used, the operating system communicates directly with the legacy BIOS **214**. In cases in which the EFI-based operating system **212**B is to be used, however, the EFI utility builds an EFI layer **302** that resides between the operating system and the legacy BIOS **214**. Therefore, the operating system **212**B communicates with the EFI layer **302**. If the operating system **212**B needs to communicate with the system hardware **304**, for instance to read data from a system hard disk, a command is communicated to the EFI layer **302**, and the EFI layer makes an appropriate request of the legacy BIOS **214**, which is configured for direct communication with the system hardware.

FIG. **4** describes an example of operation of the EFI implementation system **214** shown in FIG. **1**. Beginning with block **400**, the legacy BIOS **214** is activated in a similar manner to that described above in relation to FIG. **2**. Once activated, the legacy BIOS **214** controls the boot process and, as indicated in decision block **402**, determines whether to prompt the user to select an interface to use. More particularly, the legacy BIOS **214** determines whether to prompt the user to elect between the default legacy BIOS interface or the EFI.

If the legacy BIOS **214** is not configured to prompt the user to choose, flow continues to decision block **404** at which it is determined whether an indication to install the EFI has been received. By way of example, such an indication can be input by the user by selecting an appropriate keyboard button, such as a function key (e.g., the "F10" key). If such an indication were made during the boot process, the legacy BIOS **214** could launch the EFI utility such that the EFI layer will be built and an EFI-based operating system can be used.

If no such indication is received, the system **216** defaults to the legacy BIOS interface and no EFI is installed. However, if an indication that the EFI is desired (e.g., if an EFI-based operating system will be used), flow continues to block **412** described below.

With reference back to decision block **402**, if the legacy BIOS **214** is configured to prompt the user to select an interface, flow continues to block **406** at which the BIOS

interrupts the boot process and queries the user to elect one of a legacy-based boot or an EFI-based boot. The legacy BIOS **214** then awaits a response to the query and, presumably, a user selection is received, as indicated in block **408**.

Referring next to decision block **410**, if the user selection indicates that the EFI is not to be installed, the legacy BIOS **214** completes the boot process and no EFI layer will be installed. If, on the other hand, an EFI layer is to be installed, flow continues to block **412** at which the EFI launcher **218** (e.g., of the legacy BIOS **214**) launches the EFI utility **220** that, for example, resides in the HPA. When the EFI utility **220** is launched, the EFI utility takes control over the boot process and, as indicated in block **414**, builds an EFI layer (**302**, FIG. **3**) between the operating system and the legacy BIOS in the manner described above and depicted in FIG. **3**.

With the above-described functionality, a single computer system can be created that is compatible with both legacy-based and EFI-base operating systems. Given that the EFI layer is not installed within the non-volatile memory device that comprises the legacy BIOS, the memory device can be kept relatively small. In addition, given that the legacy BIOS still resides within the computer system, the EFI need not emulate operation of the legacy BIOS, thereby providing a simple solution to the problem of supporting both legacy-based and EFI-based operating systems.

Although implementation of the EFI has been described in the foregoing as being user selected, alternative embodiments are possible. For instance, the legacy BIOS can be configured to determine the configuration of the operating system that will be used during the boot process (i.e., legacy-based or EFI-based) and automatically launch the EFI utility **220** if an EFI-based operating system is detected.

In view of the foregoing, a method **500** for implementing an extensible firmware interface can be described as is provided in FIG. **5**. As is indicated in that figure, the method **500** comprises launching an EFI utility during a boot process of a computer system (**502**), and building an EFI layer that resides between an operating system of the computer system and a legacy basic input/output system (BIOS) of the computer system using the EFI utility (**504**).

What is claimed is:

1. A method for implementing an extensible firmware interface (EFI), the method comprising:

launching an EFI utility during a boot process of a computer system; and

building an EFI layer that resides between an operating system of the computer system and a legacy basic input/output system (BIOS) of the computer system using the EFI utility.

2. The method of claim **1**, wherein launching an EFI utility comprises launching an EFI utility with the legacy BIOS.

3. The method of claim **1**, wherein launching an EFI utility comprises launching an EFI utility with the legacy BIOS that resides on a non-volatile memory device of the computing system.

4. The method of claim **1**, wherein launching an EFI utility comprises launching an EFI utility that resides in a host protected area (HPA) of a hard disk of the computing system.

5. The method of claim **1**, wherein launching an EFI utility comprises launching an EFI utility that resides on a removable disk that is insertable into a disk drive of the computing system.

6. The method of claim **1**, wherein building an EFI layer comprises building an EFI layer that is configured to communicate with an EFI-based operating system.

7. The method of claim **1**, further comprising prompting a user during the boot process to elect between installing the EFI layer and not installing the EFI layer.

8. The method of claim **7**, wherein prompting a user comprises interrupting the boot process with the legacy BIOS.

9. The method of claim **1**, further comprising receiving an indication from a user as to whether to install the EFI layer.

10. A system for implementing an extensible firmware interface (EFI), the system comprising:

means for determining whether to build an EFI layer on a computer system;

means for launching an EFI utility during a boot process of the computer system; and

means for building an EFI layer that resides between an operating system of the computer system and a legacy basic input/output system (BIOS) of the computer system using the EFI utility.

11. The system of claim **10**, wherein the means for determining comprise part of the legacy BIOS.

12. The system of claim **10**, wherein the means for launching comprise an EFI utility launcher of the legacy BIOS.

13. The system of claim **10**, wherein the means for building comprise an EFI utility that resides in a host protected area (HPA) of a hard disk of the computing system.

14. An extensible firmware interface (EFI) implementation system stored on a computer-readable medium, the system comprising:

logic configured to launch an EFI utility during a boot process of a computer system; and

logic configured to build an EFI layer that resides between an operating system of the computer system and a legacy basic input/output system (BIOS) of the computer system using the EFI utility.

15. The system of claim **14**, wherein the logic configured to launch comprises an EFI utility launcher of the legacy BIOS.

16. The system of claim **14**, wherein the logic configured to build comprises the EFI utility that resides in a host protected area (HPA) of a hard disk of the computer system.

17. The system of claim **14**, wherein the logic configured to build comprises the EFI utility that resides on a removable disk that is insertable into a disk drive of the computer system.

18. The system of claim **14**, further comprising logic configured to determine whether to install an EFI layer.

19. A computer system, comprising:

a processor; and

a non-volatile memory device that comprises a legacy basic input/output system (BIOS), the legacy BIOS including an extensible firmware interface (EFI) utility launcher that is configured to launch an EFI utility during a boot process of the computer system; and

a hard disk that comprises an unprotected area that comprises an operating system and a host protected area (HPA) that comprises a bootable EFI utility that is configured to build an EFI layer that resides between the operating system and the legacy BIOS, the EFI layer being configured to communicate with the operating system and the legacy BIOS.

20. The computer system of claim **19**, wherein the non-volatile memory device comprises a flash-based memory chip.

* * * * *

Appendix B

This claim chart compares claims 1-27 to U.S. Patent No. 5,450,576 to Kennedy ("*Kennedy*") (Ex. 1006) in view of U.S. Patent Publication No. 2004/0123201 to Nguyen et al. ("*Nguyen*") (Ex. 1004) and further in view of U.S. Patent No. 7,085,959 to Safford ("*Safford*") (Ex. 1007). Kennedy qualifies as prior under pre-AIA § U.S.C. 102(b) because it was filed in the United States on April 26, 1993, and was issued as a patent on September 12, 1995, over a year before the '302 patent was filed. Nguyen qualifies as prior art under pre-AIA § U.S.C. 102(a) because it was filed in the United States on December 19, 2002, and was published on June 24, 2004, before the '302 patent was filed. Safford qualifies as prior art under pre-AIA § U.S.C. 102(a) because it was filed in the United States on July 3, 2002, and was published on January 8, 2004, less than one year before the '302 patent was filed.

The table below reflects SNQ2 presented in the corresponding Request for Reexamination, which includes the pertinent motivations to combine the relevant art. Further, corroborated evidence reflecting the perspective knowledge of a person having ordinary skill in the art at the time of the invention of the '302 Patent (a "POSITA") is provided through the declaration of Andrew Wolfe ("Wolfe Decl."), (Ex. 1003).

References
Kennedy (Ex. 1006)
Nguyen (Ex. 1004)
Safford (Ex. 1007)

Claim Language	Relevant Prior Art Disclosures and Argument
[1P] A method comprising:	Kennedy describes a method outlined in as follows: "The architecture provides a method and apparatus to interface multiple processors to the backplane system bus such that upon start-up of the computer system, one default boot CPU is designated to boot the system, and if this CPU fails, control transfers automatically to an alternate processor."." *Kennedy* (Ex. 1006) at 2:15-21.

[1A] detecting loss of lockstep (LOL) for a processor in a multi-processor system;	Kennedy, as modified by Nguyen and Safford, teaches detecting loss of lockstep for a processor in a multi-processor system by disclosing "execution cores 120 [that] operate in lock step" and an FRC checker that compares "signals from execution cores 120 . . . to detect soft errors in either execution core." *Nguyen* (Ex. 1004) ¶ [0029]. "Execution cores 120 operate in lock step during normal FRC mode..." *Id.* ¶ [0024]. "In FRC mode, signals from execution cores 120 may be processed by FRC checker 130, which compares them **to detect soft errors in either execution core**." *Id.* ¶ [0029]. These soft errors or mismatches that are detected between processor pair outputs are equivalent to detecting a loss of lockstep. *Wolfe Decl.* (Ex. 1003) ¶ 130. Indeed, Kennedy as modified by Nguyen further notes, "For this reason, FRC checker 130 is temporarily disabled **if error detector 140 detects an error in either error pipeline 128**, since execution cores 120 are **no longer in lock step**." *Id.* ¶ [0023]. As such, Kennedy as modified by Nguyen detects errors that indicate that two execution cores are no longer operating in lock step. This is equivalent to "detecting a loss of lockstep" as claimed in claim 1 of the '302 Patent. *Wolfe Decl.* (Ex. 1003) ¶ 130. To the extent that Nguyen is not held to teach a multi-processor system, Kennedy states, "The present invention provides a **multiprocessor interconnection architecture** with a backplane system bus for use with multiple processors." *Kennedy* (Ex. 1006) at 2:13-15. A POSITA would have been motivated to improve the reliability and robustness of Kennedy's multiprocessor system by detecting loss of lockstep, as taught by Nguyen, in order to fail over to an alternate boot processor in scenarios where a processor's failure is due to loss of lockstep. *Wolfe Decl.* (Ex. 1003) ¶ 130.
[1B] determining that the processor for which said LOL is detected is assigned a role of boot processor;	Kennedy, Nguyen, and Safford teach determining that the processor for which said LOL is detected is assigned a role of boot processor by disclosing detecting a recoverable error and allowing a processor designated as "alternative processor" to take over for a processor designated as "boot processor." *Kennedy* (Ex. 1006) at 7:9-12. Kennedy discloses that: "...provision needs to be made to determine which processor will boot the system, and if this "default" boot processor fails to boot the system, an alternative processor, if one is installed, should take over boot operations to enhance the availability of the system." *Id.* at 7:5-13. Kennedy further states: "When multiple CPUs are installed in the system, if the **default boot processor exceeds the time allotted for it to boot (it fails)**, an alternative boot processor can take over boot operations as explained below. *Id.* at 9:65-68. Thus, Kennedy determines both that a recoverable failure has occurred, and that the failure occurred at a default boot processor. *Wolfe Decl.* (Ex. 1003) ¶ 131.

	To the extent that Kennedy does not specifically indicate that the error detected was a loss of lockstep, Nguyen determines that an LOL occurs on its processor cores. *Nguyen*, (Ex. 1004) ¶ [0027]. "Execution cores 120 operate in lock step during normal FRC mode..." *Id.* ¶ [0024]. "In FRC mode, signals from execution cores 120 may be processed by FRC checker 130, which compares them **to detect soft errors in either execution core**." *Id.* ¶ [0029]. These soft errors or mismatches that are detected between processor pair outputs are equivalent to detecting a loss of lockstep. Indeed, Nguyen further notes, "For this reason, FRC checker 130 is temporarily disabled **if error detector 140 detects an error in either error pipeline 128**, since execution cores 120 are **no longer in lock step**." *Id.* ¶ [0023]. As such, Nguyen detects errors that indicate that its two execution cores are no longer operating in lock step. This is equivalent to "detecting a loss of lockstep" as claimed in claim 1 of the '302 Patent. *Wolfe Decl.* (Ex. 1003) ¶ 132. In this manner, the combination of Kennedy, Nguyen, and Safford determines whether a recoverable failure has occurred and whether the failure has occurred at a designated default boot processor. *Kennedy* (Ex. 1006) at 7:5-13. Nguyen indicates that such recoverable failures may include a loss of lockstep. *Nguyen*, (Ex. 1004) ¶ [0027]. Because Kennedy, Nguyen, and Safford describe high-availability computing architectures, a POSITA would be motivated to combine these references and would have a reasonable expectation that combining the two references would lead to a more reliable multiprocessor system. *Wolfe Decl.* (Ex. 1003) ¶ 133.
[1Ci] changing a system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected, to a different value	Kennedy as modified by Nguyen and Safford teaches this limitation by disclosing that, "To improve reliability of processing assets, a computer system employs lock stepped processor cores that operate in a **master/check pair**." *Safford* (Ex. 1007) at 2:55-57. Safford further describes one pair of these processors "operating in a lock step mode" as initially "appear[ing] to [a] computer system...to be...a **logical CPU 0,**" but later that **logical CPU 0** designation is reassigned to another processor upon failover. *See Safford* (Ex. 1007) at 3:4-5, 4:52-53. For example, "The processor pair 125/127 then becomes the **logical CPU 0** in the computer system 100". *Id.* A POSITA would have recognized Safford's logical CPU 0 as being the designation used by computing system 100 to identify its boot processor at least because, in multi-processor system architectures like Safford's, "**logical CPU 0**" was commonly used to indicate that a processor had been selected as a boot processor. *See, e.g., Cepulis* (Ex. 1014) at 2:52-64, *see also Wolfe Decl.* (Ex. 1003) ¶ 134. For example, during boot-processor selection,

	If CPU P0 is not functioning properly and cannot perform even the most basic functions, it is designated as inoperative. The hardware system then awakens CPU P1 and repeats the process of testing the CPU. If CPU P1 is operational, then it is designated as CPU L0, and it boots the remainder of the system. On the other hand, if CPU P1 also fails, it is also given an inoperative designation. The computer system then turns on CPU P2, and repeats the process. The process repeats until an operational microprocessor is found to perform the CPU L0 functions.[1] *Cepulis* (Ex. 1014) at 2:52-64. A POSITA would have also recognized "CPU 0" (and other designators taught by Safford, such as "CPU 1") as being system designations used by computing system 100 to identify processors 111 and 113 in Safford's system (e.g., processors 111/113 are initially recognized as "logical CPU 0" and, after failure of one of these processors, processors 125/127 are designated as "logical CPU 0." *Safford* (Ex. 1007) at 3:4-5, 4:52-53. The use of system designators by a computing system is generally necessary when the role of boot processor can be assigned and reassigned to a processor regardless of its physical position within a system. *See Wolfe Decl.* (Ex. 1003) ¶ 135; *see also Safford* (Ex. 1007) at 4:52-53 ("The processor pair 125/127 then becomes the **logical CPU 0** in the computer system 100"). Thus, a POSITA would have understood "**logical CPU 0**" to be the boot-processor designation used by an operating system to determine which of its processors is the boot processor and/or to distinguish its boot processor from its non-boot processors. *See Wolfe Decl.* (Ex. 1003) ¶ 135. Moreover, Safford teaches the storing of state information for failed processors. *Safford*, (Ex. 1007) at 4:40-51. Because the state information of Safford's processors is copied from memory, a POSITA would have understood the memory to be storing the boot-processor designation (*e.g.*, logical CPU 0) for the processors so assigned by Safford's system. *See Id.* In view of these citations and the knowledge that a boot processor was a common component identified as "logical CPU 0" in computer systems (*e.g.*, Safford's system), a POSITA would have understood Safford as disclosing data storage that stores information assigning the role of boot processor to one of the processors and changing that assignment by changing the identifier "logical CPU 0" during a failover. *See supra* § I.C.2; *see also Wolfe Decl.* (Ex. 1003) ¶ 135. Still further, A POSITA would have combined Safford with Kennedy and Nguyen because each is concerned with providing processor failover mechanisms in high-

[1] Cepulis is provided herein to explain the meaning of the term "boot processor" used in Safford. *See* M.P.E.P. § 2131.01.

	availability environments, and because Kennedy and Nguyen would perform the failover more reliably when changing a system identifier as taught by Safford. *Id., see also Wolfe Decl.* (Ex. 1003) ¶ 135.
[1Cii] which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor	Kennedy, Nguyen, and Safford teach causing the system's operating system to not recognize the processor for which said LOL is detected as the boot processor by disclosing that a failed processor pair 111/113 is taken off-line, that its state information is copied to another processor pair (e.g., a "hot standby" processor 125/127), and that [t]he processor pair 125/127 then becomes the logical CPU 0 in the computer system 100, and the computer system 100 operates without a hot standby processor pair. *Safford* (Ex. 1007) at 4:38-53. Thus, in Kennedy's system as modified by Nguyen and Safford, in the event of a processor failure (such as a loss of lockstep, as taught by Nguyen), a new processor is designated as the logical CPU 0, and the old (failed) processor no longer retains that designation. *Id.*, see also *Wolfe Decl.* (Ex. 1003) ¶ 136. A POSITA would have been motivated to improve the reliability of Kennedy's system by incorporating a change in system identifiers, as taught by Safford, to ensure that its operating system would no longer refer to failed processors as "logical CPU 0" when booting. *Wolfe Decl.* (Ex. 1003) ¶ 136.
[1Di] changing a system identifier of the spare processor to that of the boot processor,	Kennedy as modified by Safford teaches that "To improve reliability of processing assets, a computer system employs lock stepped processor cores that operate in a **master/check pair**." *Safford* (Ex. 1007) at 2:55-57. Safford further describes one pair of these processors "operating in a lock step mode" as initially "appear[ing] to the computer system…to be…a **logical CPU 0,**" but later that **logical CPU 0** designation is reassigned to another processor. *See Safford* (Ex. 1007) at 3:4-5, 4:52-53 ("The processor pair 125/127 then becomes the **logical CPU 0** in the computer system 100"). *Id.*, *see also Wolfe Decl.* (Ex. 1003) ¶ 137. By changing the former standby processor 125/127 to the "logical CPU 0" processor, the system identifier of the spare processor (e.g., CPU 1) is changed to that of the boot processor (e.g., CPU 0). *Id.*, *see also Wolfe Decl.* (Ex. 1003) ¶ 137. In light of the above citations, a POSITA would have understood "**logical CPU 0**" to be the boot-processor designation used by an operating system to determine which of its processors is the boot processor and/or to distinguish its boot processor from its non-boot processors. *See Wolfe Decl.* (Ex. 1003) ¶ 137. Moreover, a POSITA would have been motivated to improve the reliability of Kennedy's system by changing these system identifiers during failover, as taught by Safford, to ensure that its operating system would properly refer to the designated "logical CPU 0" when operating the processor to which failover occurred. *Id.*

[1Dii] thereby switching the role of boot processor to a spare processor without shutting down the system's operating system.	Kennedy as modified by Nguyen and Safford teaches this limitation. Kennedy notes that an error can lead to switching the role of boot processor from one processor to another: "The architecture provides a method and apparatus to interface multiple processors to the backplane system bus such that upon start-up of the computer system, one default boot CPU is designated to boot the system, and if this CPU fails, **control transfers automatically to an alternate processor**." *Kennedy* (Ex. 1006) at 2:15-21. Moreover, Kennedy states that: "...provision needs to be made to determine which processor will boot the system, and if this "default" boot processor fails to boot the system, an alternative processor, if one is installed, should take over boot operations to enhance the availability of the system." *Id.* at 7:5-13. As such, Kennedy as modified by Nguyen and Safford teaches an automatic switching of roles from default boot processor to alternative processor or vice versa. *Wolfe Decl.* (Ex. 1003) ¶ 138. To the extent that Kennedy is not interpreted to teach role switching of a boot processor to a spare processor without shutting down the system's operating system, Nguyen teaches switching from a processor to a spare processor without shutting down the system's operating system. In combination, Kennedy would be modified by Nguyen and Safford to include PAL-based implementations that do not require a system shutdown. Specifically, Kennedy, as modified by Nguyen and Safford, would switch from a boot processor to an alternative processor using a PAL-based, firmware implementation that would allow the switch between processors to be performed quickly enough to avoid trigging a timeout period that would otherwise be enforced by the operating system. *Wolfe Decl.* (Ex. 1003) ¶ 139. Nguyen notes that "PAL-based implementations of the recovery mechanism should be able to complete quickly enough to avoid triggering a time-out period enforced by the operating system. *Nguyen* (Ex. 1004) ¶ [0045]." Nguyen states: > As noted above, the recovery mechanism for errors detected within the FRC boundary may be handled by **various combinations of hardware, software and firmware modules**. One embodiment of the recovery mechanism employs code that is closely associated with the processor. For example, the Itanium® Processor Family of Intel® Corporation, employs a **layer of firmware** called the processor abstraction layer (PAL), which **provides an abstraction of the processor to the rest of the computing system**. Implementing recovery in the PAL **hides the recovery process from system level code**, such as the system abstraction layer (SAL), e.g. BIOS, **and the operating system**. PAL-based implementations of the recovery mechanism

	should be able to complete quickly enough to **avoid triggering a time-out period enforced by the operating system**. *Id.* In this example, implementing PAL-based firmware to implement the recovery mechanism quickly enough to avoid triggering a time-out period enforced by the operating system is equivalent to performing the recovery mechanism without shutting down the system's operating system. *Wolfe Decl.* (Ex. 1003) ¶ 140. Using a firmware-based PAL layer hides the recovery process from system level code such as the operating system. Changes in system level code would require an operating system shutdown to implement. *Id.* PAL-based implementations, however, provide recovery without triggering a time-out period or other operations enforced by the operating system. This is equivalent to avoiding an operating system shutdown. Accordingly, the recovery mechanisms discussed in Nguyen are carried out in firmware, without shutting down the operating system. *Id.* A POSITA would have been motivated to improve the efficiency of Kennedy's failover mechanism by implementing PAL-based firmware, as taught by Nguyen, in order to quickly fail over to an alternate boot processor in a seamless and prompt manner. *Id.*
[**2**] wherein the spare processor is a hot spare that is held in idle until switched the role of boot processor	To the extent that Kennedy and Nguyen to not explicitly teach a hot spare that is held in idle until switched to the role of boot processor, Kennedy and Nguyen, when combined with Safford, teach that the spare processor is a hot spare that is held in idle until switched the role of boot processor by disclosing a spare processor that is initially idled and becomes active upon a first processor unit losing lock step. *Safford* (Ex. 1007) at 2:26-32. Safford states: "The processor 125 may be designated as a "**hot standby**" and is **sitting idle** in lock step mode with the processor 127. Should one of the processors 111, 113, 121, and 123 suffer an error, the hot standby processors 125, 127 may be used to **speed recovery** from the resulting loss of lock step. *Id.* at 4:25-29). A POSITA would have combined Safford with Kennedy and Nguyen because each is concerned with providing processor failover mechanisms in high-availability environments, and because Kennedy and Nguyen would perform the failover more quickly when using a hot spare as taught by Safford. *Id.*, *see also Wolfe Decl.* (Ex. 1003) ¶ 141.

[3] wherein the spare processor is an active spare	To the extent that Kennedy and Nguyen to not explicitly teach an active spare that is used as a spare processor, Kennedy and Nguyen, when combined with Safford, teach that the spare processor is an active spare by disclosing "moving the architected state comprises **copying the architected state directly** from the loss of lock step processor unit **to the spare processor unit**." *Safford* (Ex. 1007) at 5:23-26). A POSITA would have combined Safford with Kennedy and Nguyen because each is concerned with providing processor failover mechanisms in high-availability environments, and because Kennedy and Nguyen would be able to perform additional processing when using an active spare, as taught by Safford. *Id.*, *see also Wolfe Decl.* (Ex. 1003) ¶ 142.
[4] wherein said switching further comprises: causing the system's operating system to idle the active spare	To the extent that Kennedy and Nguyen to not explicitly teach causing the system's operating system to idle the active spare, Kennedy and Nguyen, when combined with Safford, teach that switching further includes causing the system's operating system to idle the active spare by disclosing a "means for moving an architected state of a processor unit having the loss of lock step initiating event to a **spare processor unit**, the spare processor unit **operating idle** in lock step." *Safford* (Ex. 1007) at 6:2-5). A POSITA would have combined Safford with Kennedy and Nguyen because each is concerned with providing processor failover mechanisms in high-availability environments, and because Kennedy and Nguyen would be able to reduce processing resource usage by idling the active spare, as taught by Safford. *Id.*, *see also Wolfe Decl.* (Ex. 1003) ¶ 143.
[5] wherein the spare processor is a processor module that includes a master and slave processors that operate in lockstep.	Kennedy, in combination with Nguyen and Safford, teaches a multiprocessor system that includes boot processors and alternative processors. *Kennedy* (Ex. 1006) at 2:62-64 and 7:7-13. Kennedy, in combination with Nguyen and Safford, further discloses a pair of processors that operate in lockstep by disclosing that: In FRC mode, **one of execution cores 120 may be designated as the master**. The master refers to the execution core that is responsible for updating the resources shared by execution cores 120. **The other execution core 120 may be designated as the slave**. The slave is responsible for generating results from the same code sequence to be checked against those of the master." *Nguyen* (Ex. 1004) ¶ [0027].

	These execution cores 120 "operate in lock step during normal FRC mode..." *Id.* ¶ [0024]. A POSITA would have combined Nguyen with Kennedy and Safford because each is concerned with providing reliable processor failover mechanisms in high-availability environments, and because Kennedy and Safford would perform processor failovers more reliably when using a spare processor with master and slave execution cores as taught by Nguyen. *Id.*, *see also Wolfe Decl.* (Ex. 1003) ¶ 145.
[6] wherein said determining that the processor for which said LOL is detected is assigned the role of boot processor comprises accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor.	Kennedy, as modified by Nguyen and Safford, teaches accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor by disclosing that "The CPU1 120 has a corresponding **non-volatile memory 123 which contains configuration information and CPU specific initialize and self test code (ISTC) for the CPU1 120**..." *Kennedy* (Ex. 1006) at 5:15-18. Kennedy further indicates that: "The slot select logic 148 is described in more detail in reference to the CPU1 120 installed in slot 1 as the default boot processor and the CPU2 132 installed in slot 2 as the alternate boot processor. *Id.* at 10:55-58, see also Table 2. It would have been obvious to combine Kennedy with Nguyen and Safford to increase the dependability of the system by storing configuration information in non-volatile memory, as taught by Kennedy. *Id.*, *see also Wolfe Decl.* (Ex. 1003) ¶ 146.
[7] further comprising: determining the processor that is assigned the role of spare processor.	Kennedy as modified by Nguyen and Safford teaches determining the processor that is assigned the role of spare processor by disclosing that "...provision needs to be made to **determine which processor will boot the system,** and if this "default" boot processor fails to boot the system, **an alternative processor**, if one is installed, **should take over boot operations** to enhance the availability of the system." *Kennedy* (Ex. 1006) at 7:5-13, *see also* Nguyen (Ex. 1004) at ¶ [0026]. It would have been obvious to a POSITA to combine Kennedy with Nguyen and Safford to increase the reliability of the system by identifying which processors are assigned certain roles, as taught by Kennedy. *Id.*, *see also Wolfe Decl.* (Ex. 1003) ¶ 147.

[8] wherein said determining the processor that is assigned the role of spare processor comprises: accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of spare processor.	Kennedy as modified by Nguyen and Safford teaches accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor by disclosing that "The CPU1 120 has a corresponding **non-volatile memory 123 which contains configuration information and CPU specific initialize and self test code (ISTC) for the CPU1 120**..." *Kennedy* (Ex. 1006) at 5:15-18. Kennedy further indicates that: "The slot select logic 148 is described in more detail in reference to the CPU1 120 installed in slot 1 as the default boot processor and the CPU2 132 installed in slot 2 as the alternate boot processor. *Id.* at 10:55-58, *see also* Table 2. It would have been obvious to a POSITA to combine Kennedy with Nguyen to avoid system errors by identifying which processors are assigned certain roles, as taught by Kennedy. *Id.*, *see also Wolfe Decl.* (Ex. 1003) ¶ 148.
[9] wherein said determining step is performed by system firmware.	Kennedy as modified by Nguyen teaches determining whether an error has occurred for which recovery methods are to be used. *Kennedy,* (Ex. 1006) at 7:7-13. Kennedy as modified by Nguyen further teaches that a firmware-based processor abstraction layer is usable in such a recovery process. *Nguyen* (Ex. 1004) ¶ [0045]. In recovery scenarios, Kennedy as modified by Nguyen implements a firmware-based processor abstraction layer (PAL), which "**provides an abstraction of the processor** to the rest of the computing system." *Id.* "**Implementing recovery in the PAL** hides the recovery process from system level code, such as the system abstraction layer (SAL), e.g. BIOS, and the operating system." *Id.* As such, Kennedy as modified by Nguyen indicates that the process steps involved in its recovery processes, including determining steps, is performed by firmware. *Wolfe Decl.* (Ex. 1003) ¶ 149.
[10] wherein said switching step is performed by system firmware.	Kennedy as modified by Nguyen teaches determining whether an error has occurred for which recovery methods are to be used. *Kennedy,* (Ex. 1006) at 7:7-13 Kennedy as modified by Nguyen further teaches that the switching step is performed by system firmware by implementing a firmware-based processor abstraction layer in the recovery process. *Nguyen* (Ex. 1004) ¶ [0045]. In recovery scenarios, Nguyen implements a firmware-based processor abstraction layer (PAL), which "provides an abstraction of the processor to the rest of the computing system." *Id.* "**Implementing recovery in the PAL** hides the recovery process from system level code, such as the system abstraction layer (SAL), e.g. BIOS, and the operating system." *Id.* As such, Kennedy as modified by Nguyen indicates that the process steps involved in its recovery processes, including switching steps, is performed by firmware.

[11] wherein the processor for which said LOL is detected includes a lockstep pair of processors, and wherein said switching comprises: determining one of the lockstep pair of processors that is not the cause of said LOL; and copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to said spare processor.	Kennedy as modified by Nguyen and Safford teaches determining one of the lockstep pair of processors that is not the cause of said LOL and copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to said spare processor by disclosing, "the node controller 130 may store the architected state of the processors 111, 113, 121, 123, 125, and 127, either internally in the node controller 130, or in another component of the computer system 100." *Safford*, (Ex. 1007) at 4:10-14. While nodes 111 and 113 are indicated as having potentially lost lockstep, nodes 121, 123, 125, and 127 have not lost lockstep and, according to Safford, the node controller 130 is capable of copying and storing the state information of those processors that have not caused a loss of lockstep. *Id.* at 4:6-18. A POSITA would have been motivated to combine Safford's copying of state information for non-failed processors (i.e., processors that have not lost lockstep) in order to further bolster the reliability of its high-availability system through the copying of processor state information for non-failed, functioning nodes. *Wolfe Decl.* (Ex. 1003) ¶ 151.
[12] wherein said spare processor includes a lockstep pair of processors, and wherein said copying comprises: copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to each of said lockstep pair of processors of said spare processor	Kennedy as modified by Nguyen and Safford teaches copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to each of said lockstep pair of processors of said spare processor by disclosing, "the node controller 130 may store the architected state of the processors 111, 113, 121, 123, 125, and 127, either internally in the node controller 130, or in another component of the computer system 100." *Safford*, (Ex. 1007) at 4:10-14. While nodes 111 and 113 are indicated as having potentially lost lockstep, nodes 121, 123, 125, and 127 have not lost lockstep and, according to Safford, the node controller 130 is capable of copying and storing the state information of those processors that have not caused a loss of lockstep. *Id.* at 4:6-18. Safford specifically notes that the state information is copied to each processor in the pair of processors (e.g., to processor pairs 121/123 or 125/127). *Id.* A POSITA would have been motivated to combine Safford's copying of state information for non-failed (working) nodes in order to further increase the reliability of its high-availability system by keeping a store of state information copied from working nodes. *Wolfe Decl.* (Ex. 1003) ¶ 152.
[13P] A method comprising:	*See* Limitation [1P].
[13A] system firmware determining that loss of lockstep (LOL) is detected for a lockstep pair of processors that are assigned the role of boot processor in a system;	*See* Limitation [1A].

[13B] determining one of said lockstep pair of processors that is not the cause of said LOL;	Kennedy as modified by Nguyen and Safford teaches determining one of said lockstep pair of processors that is not the cause of said LOL by disclosing detecting an error that is localized to one of the execution cores. Kennedy as modified by Nguyen and Safford states: "As long as the error is detected before it triggers an FRC reset, the corrupted data can be localized to one of the execution cores, leaving the **machine state data** of the other execution core **available for recovery**." *Nguyen* (Ex. 1004) ¶ [0046]).
[13C] copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to a spare processor;	Kennedy as modified by Nguyen and Safford teaches copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to a spare processor by disclosing "the node controller 130 may store the architected state of the processors 111, 113, 121, 123, 125, and 127, either internally in the node controller 130, or in another component of the computer system 100." *Safford*, (Ex. 1007) at 4:10-14. While nodes 111 and 113 are indicated as having potentially lost lockstep, nodes 121, 123, 125, and 127 have not lost lockstep and, according to Safford, the node controller 130 is capable of copying and storing the state information of those processors that have not caused a loss of lockstep. *Id.* at 4:6-18. A POSITA would be motivated to combine Safford's copying of state information for non-failed, working nodes in order to further increase the reliability of its high-availability system through the storing of processor state information copied from working nodes. *Wolfe Decl.* (Ex. 1003) ¶ 156.
[13Di] changing a system identifier, used by the system's operating system to recognize the boot processor, of said lockstep pair of processors that are assigned the role of boot processor, to a different value,	*See* Limitation [1Ci].
[13Dii] which causes the system's operating system to not recognize the lock step pair of processors for which said LOL is detected as the processor; and	*See* Limitation [1Cii].
[13E] changing a system identifier of the spare processor to that of the boot processor, thereby switching the role	*See* Limitation [1Di].

of boot processor to said spare processor.	
[**14**] wherein the spare processor is a hot spare that is held in idle until switched the role of boot processor	*See* Limitation [2].
[**15**] wherein the spare processor is an active spare	*See* Limitation [3].
[**16**] further comprising: causing the system's operating system to idle the active spare	*See* Limitation [4].
[**17**] wherein said switching causes the system's operating system to recognize said spare processor as said boot processor without shutting down said operating system.	*See* Limitation [1Dii].
[**18**] wherein said determining step is performed by system firmware.	*See* Limitation [9].
[**19**] wherein said copying step is performed by system firmware	*See* Limitation [11].
[**20**] wherein said switching step is performed by system firmware.	*See* Limitation [10].
[**21**] A system comprising:	*See* Limitations [1A]-[1Dii].

[**22**] wherein the another processor module is a processor module assigned a role of spare processor.	Kennedy as modified by Nguyen and Safford teaches the other processor module is a processor module assigned a role of spare processor by disclosing that "…provision needs to be made to **determine which processor will boot the system**, and if this "default" boot processor **fails to boot the system, an alternative processor**, if one is installed, **should take over boot operations** to enhance the availability of the system." *Kennedy* (Ex. 1006) at 7:5-13. To the extent that Kennedy is not interpreted to disclose a processor module, Nguyen discloses a processor 110 that includes execution cores 120(a) and 120(b), which a POSITA would recognize as a processor module. *Nguyen* (Ex. 1004) ¶ [0027], *see also Wolfe Decl.* (Ex. 1003) ¶ 160. It would have been obvious to a POSITA to combine Kennedy with Nguyen and Safford to reduce system errors by assigning processor roles, as taught by Kennedy, and by changing system identifiers for those processors, as taught by Safford. *Id.*, *see also Wolfe Decl.* (Ex. 1003) ¶ 160.
[**23**] wherein the another processor module is held in idle until said system firmware causes said operating system to recognize said another processor module as said system boot processor	*See* Limitations [2] and [14].
[**24**] wherein the system firmware causes the operating system to idle the another processor module before causing said operating system to recognize said another processor module as said system boot processor	*See* Limitations [2] and [14].
[**25**] further comprising: non-volatile data storage storing information that assigns the role of system boot processor to one of said plurality of processor modules	*See* Limitation [8].
[**26**] A system comprising:	*See* Limitations [1A]-[1Dii].

[27] wherein said processor in said multi-processor system is an Itanium Processor Family (IPF) processor.	Kennedy in combination with Nguyen and Safford teaches a multiprocessor system that includes boot and alternative processors. *Kennedy* (Ex. 1006) at 2:13-15. Kennedy in combination with Nguyen and Safford further teaches that a processor in such a multi-processor system is an Itanium Processor Family (IPF) processor by specifically disclosing use of an Itanium processor in its system. *Nguyen* (Ex. 1004) ¶ [0045]. Kennedy in combination with Nguyen specifically mentions that Itanium processors are used in its system: "For example, the **Itanium(R) Processor Family** of Intel(R) Corporation, employs a layer of firmware called the processor abstraction layer (PAL), which provides an abstraction of the processor to the rest of the computing System." *Id.* In view of the above citations, Kennedy as modified by Nguyen and Safford teaches that the processors would have been IPF processors. *See Wolfe Decl.* (Ex. 1003) ¶ 161. A POSITA would have been motivated to combine these disclosures from Kennedy, Nguyen, and Safford to improve the spatial efficiency of the system by consolidating the processors into the multicore processor package used by the IPF. *Id.* It was well within the skill of a POSITA to modify Kennedy's system by Nguyen and Safford in this manner, and thus this combination of Kennedy, Nguyen, and Safford would have been expected to succeed. *Id.* Accordingly, Kennedy as modified by Nguyen and Safford teaches this claim limitation.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re patent of Michaelis *et al.*	§	Attorney Docket No.: 185945.012600
	§	
U.S. Patent No. 7,624,302	§	Customer No.: 67395
	§	
Issue Date: November 24, 2009	§	
	§	
Filing Date: October 25, 2004	§	
	§	
For: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR	§	

Mail Stop "*Ex Parte* Reexam"
Attn: Central Reexamination Unit
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

CERTIFICATE OF SERVICE

It is certified that, pursuant to 37 C.F.R. § 1.510(b)(5), copies of the following documents have been served in their entireties on the patent owner at the correspondence address provided for in 37 C.F.R. § 1.33(c):

1. Request for *Ex Parte* Reexamination of U.S. Patent No. 7,624,302 Transmittal Form, PTO/SB/57

2. Request for *Ex Parte* Reexamination of U.S. Patent No. 7,624,302 Pursuant to 37 C.F.R. § 1.510 and accompanying appendices and exhibits:

Ex. 1001	U.S. Patent No. 7,624,302 to Michaelis et al. ("the '302 Patent" or "the Challenged Patent")

Ex. 1002	U.S. File History of the '302 Patent ("U.S. File History")
Ex. 1004	U.S. Patent Application Pub. No. 2004/0123201 to Nguyen et al. ("Nguyen")
Ex. 1005	U.S. Patent No. 7,328,371 to Kalyanasundharam et al. ("Kalyanasundharam")
Ex. 1006	U.S. Patent No. 5,450,576 to Kennedy et al. ("Kennedy")
Ex. 1007	U.S. Patent No. 7,085,959 to Safford ("Safford")
Ex. 1008	U.S. Patent No. 6,625,749 to Quach ("Quach")
Ex. 1009	U.S. Patent Pub. No. 2005/0172164 to Fox ("Fox")
Ex. 1010	Wendy Bartlett, et. al., *Commercial Fault Tolerance: A Tale of Two Systems, IEEE Transactions on Dependable and Secure Computing*, Vol. 1, No. 1 (January-March 2004) ("Bartlett")
Ex. 1011	Intel, *Extensible Firmware Interface Specification*, Version 1.10 (Dec. 1, 2002) ("Intel EFI Specification")
Ex. 1012	U.S. Patent No. 7,191,328 to Hobson ("Hobson")
Ex. 1013	Bozidar Levi, *UNIX Administration, A Comprehensive Sourcebook for Effective Systems and Network Management* (2002) ("Levi")
Ex. 1014	U.S. Patent No. 5,491,788 to Cepulis et al. (Cepulis)
Ex. 1015	*Intel IA-64 Architecture Software Developer's Manual*, Revision 1.1 (July 2000) ("Intel Itanium Manual")
Ex. 1016	*Intel Multiprocessor Specification*, Version 1.4 (May 1997) ("Intel Multiprocessor Specification")
Ex. 1017	U.S. Patent No. 5,915,082 to Marshall ("Marshall")
Ex. AA	Appendix A: Claim chart comparing claims to Kalyanasundharam in view of Kennedy, as set forth in Ground 1.

Ex. BB	Appendix B: Claim chart comparing claims to Kennedy in view of Nguyen and Safford, as set forth in Ground 2.
Ex. CC	Appendix C: Claim chart comparing claims to Safford in light of Cepulis, as set forth in Ground 3.

3. Information Disclosure Statement, PTO/SB/08, listing references cited in the Request for Ex Parte Reexamination of U.S. Patent No. 7,624,302 pursuant to 35 U.S.C. § 302 and 37 C.F.R. § 1.510.

The copy has been served on September 1, 2023, by causing the aforementioned documents to be deposited in the United States Postal Service as priority mail postage pre-paid in a package addressed to:

FORAS TECHNOLOGIES LTD.
c/o 56436 - Hewlett Packard Enterprise
3404 E. Harmony Road
Mail Stop 79
Fort Collins, CO 80528

Dated: September 1, 2023

/Gregory R. Lunt/
Reg. No. 57,354

GREENBERG TRAURIG, LLP
1750 Tysons Boulevard
McLean, VA 22102
Tel: (703) 749-1370

Counsel for Requester Unified Patents, LLC

US007251749B1

(12) **United States Patent**
Fong et al.

(10) **Patent No.: US 7,251,749 B1**
(45) **Date of Patent: Jul. 31, 2007**

(54) **EFFICIENT TRUE IMAGE RECOVERY OF DATA FROM FULL, DIFFERENTIAL, AND INCREMENTAL BACKUPS**

(75) Inventors: **Yinfung Fong**, Campbell, CA (US); **Stephen Manley**, Pleasanton, CA (US)

(73) Assignee: **Network Appliance, Inc.**, Sunnyvale, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 590 days.

(21) Appl. No.: **10/778,518**

(22) Filed: **Feb. 12, 2004**

(51) **Int. Cl.**
G06F 11/00 (2006.01)

(52) **U.S. Cl.** **714/20**; 714/15; 714/21

(58) **Field of Classification Search** 714/6, 714/15, 20, 21
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,691,245 B1 * 2/2004 DeKoning 714/6
6,981,177 B2 * 12/2005 Beattie 714/20
7,069,401 B1 * 6/2006 Noonan et al. 711/162
7,152,183 B2 * 12/2006 Fujibayashi 714/6
2004/0030852 A1 * 2/2004 Coombs et al. 711/162
2004/0163009 A1 * 8/2004 Goldstein et al. 714/6

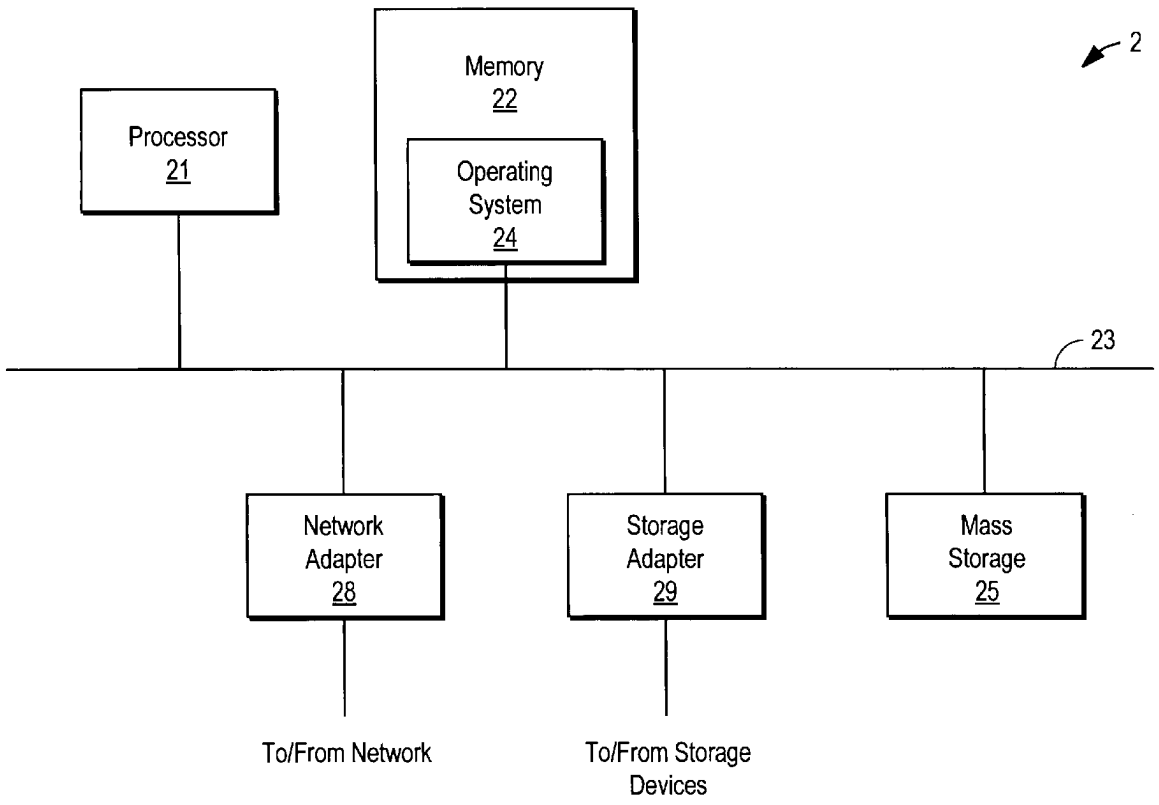
* cited by examiner

Primary Examiner—Scott Baderman
Assistant Examiner—Tim Bonura
(74) *Attorney, Agent, or Firm*—Blakely Sokoloff Taylor & Zafman LLP

(57) **ABSTRACT**

A method and apparatus to recover a set of data from a full backup and an incremental or differential backup are described. This includes a source directory that was modified between the full backup and the incremental or differential backup. The full backup is restored, including the source directory, and a new directory is created to replace the restored directory when the incremental or differential backup is applied to the restored full backup. Content for an entry from the modified source directory in the incremental or differential backup is created in the new directory if the corresponding content is present in the incremental or differential backup. If the content for the entry is not present in the incremental or differential backup, then the entry in the new directory is linked to corresponding content in the restored directory from the full backup.

18 Claims, 10 Drawing Sheets



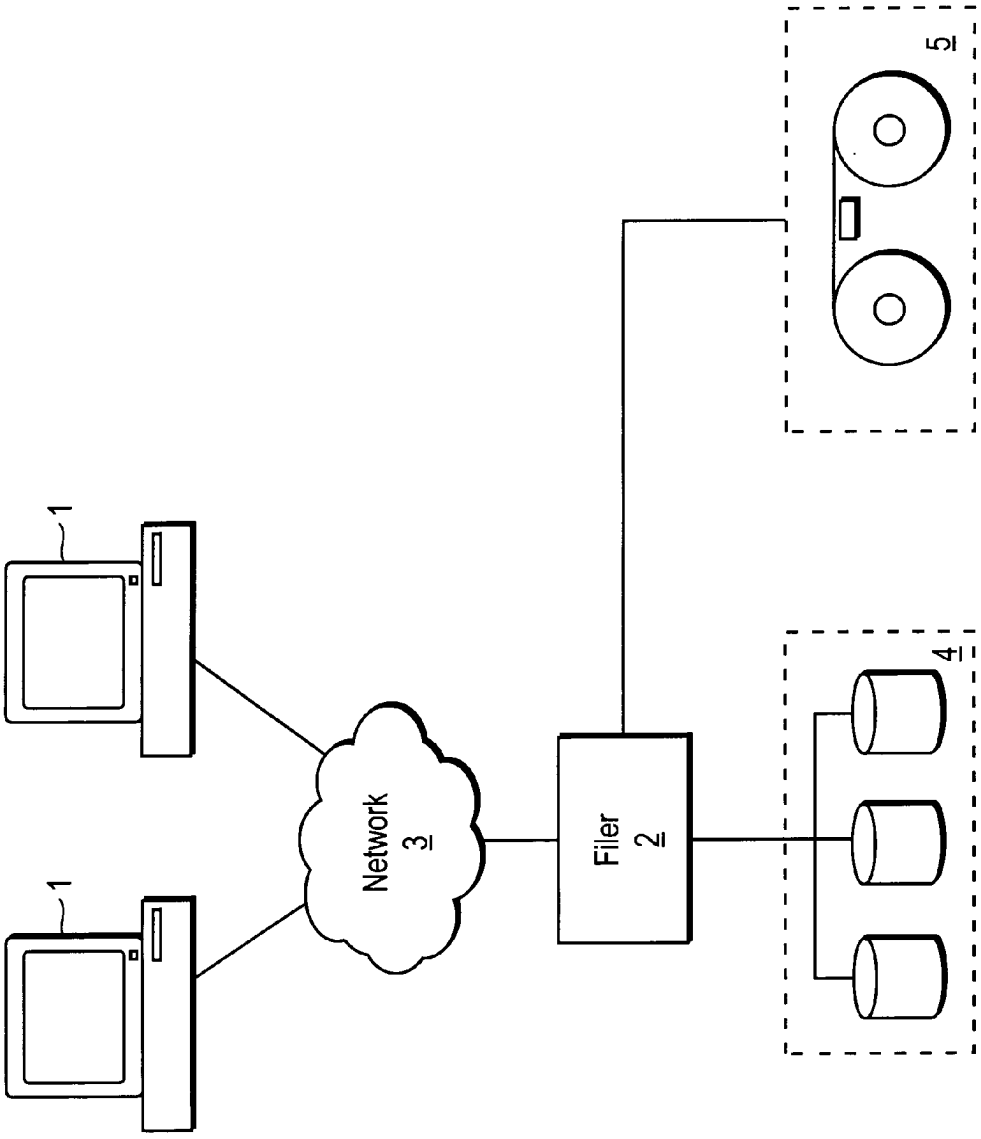


FIG. 1

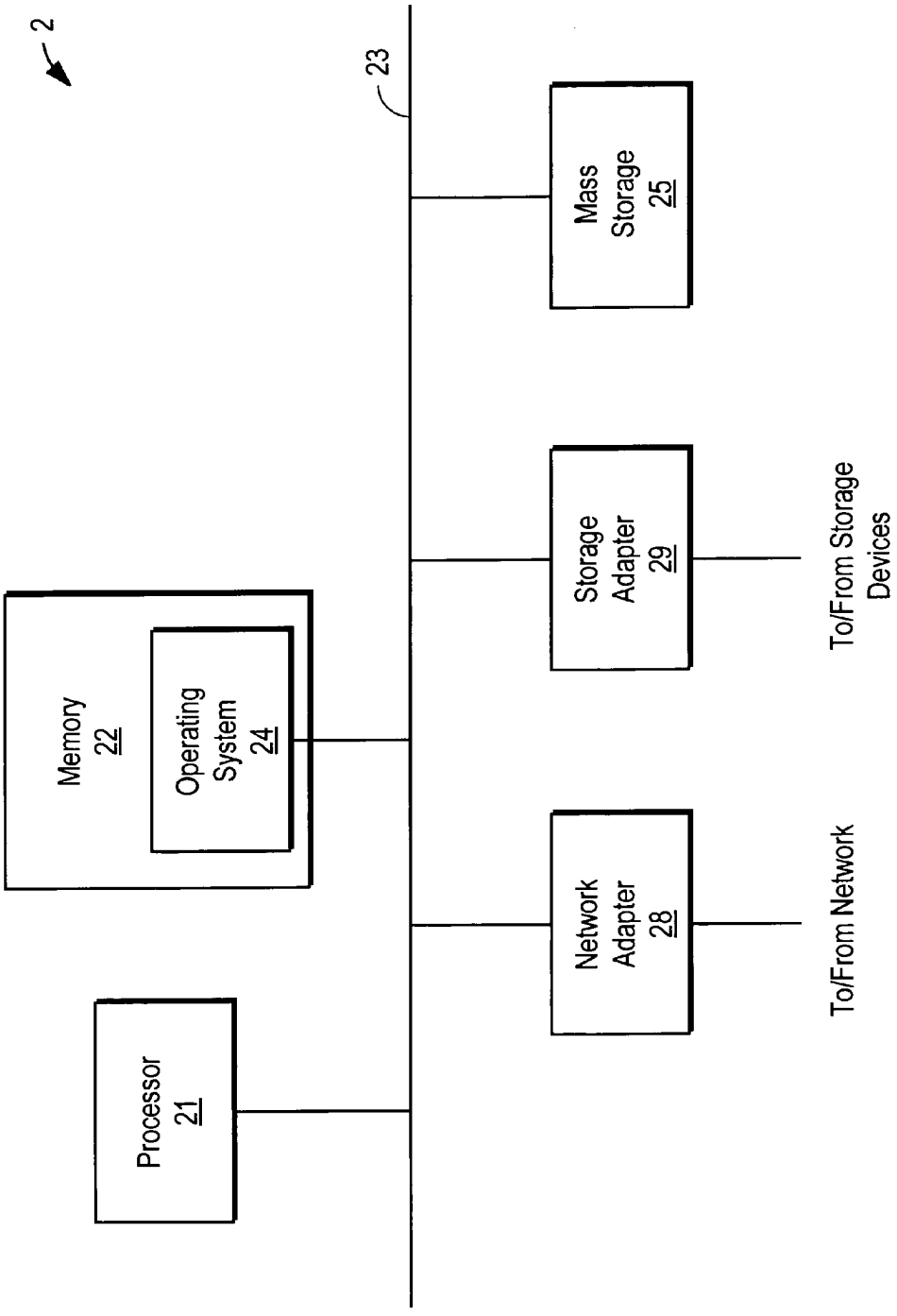

2
23
Mass Storage 25
Storage Adapter 29
To/From Storage Devices
Memory 22
Operating System 24
Network Adapter 28
To/From Network
Processor 21

FIG. 2

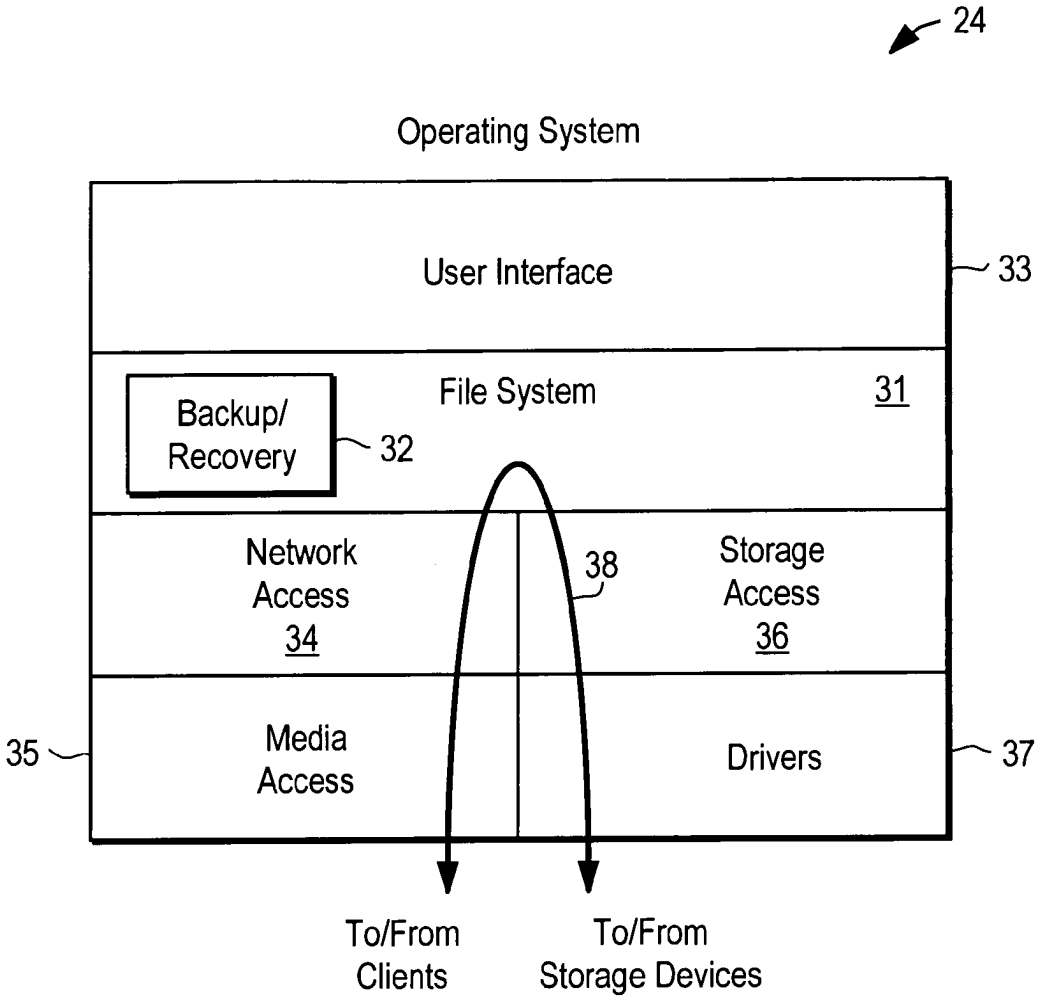

24
Operating System
User Interface
33
File System
31
Backup/
Recovery
32
Network
Access
34
38
Storage
Access
36
Media
Access
35
Drivers
37
To/From
Clients
To/From
Storage Devices

FIG. 3

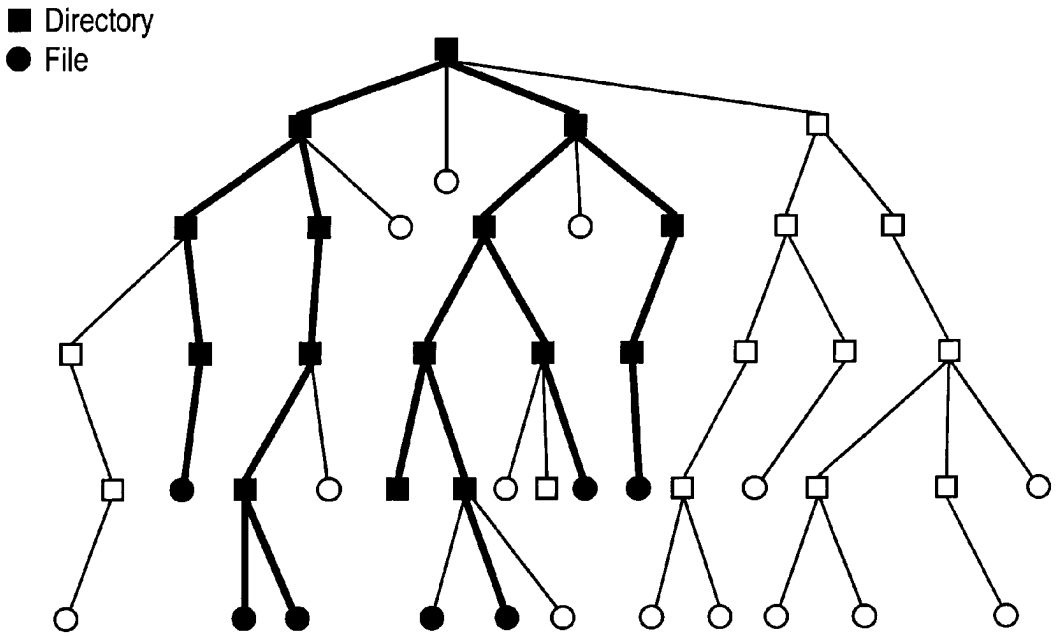


FIG. 4

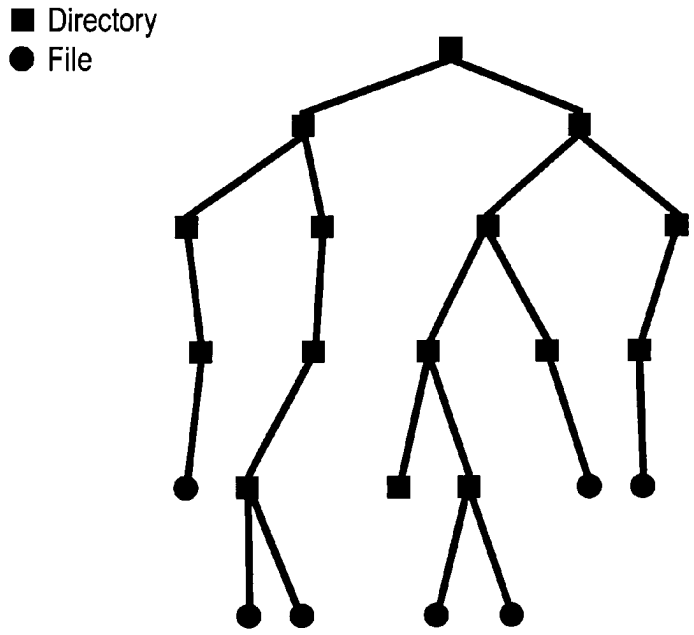


FIG. 5

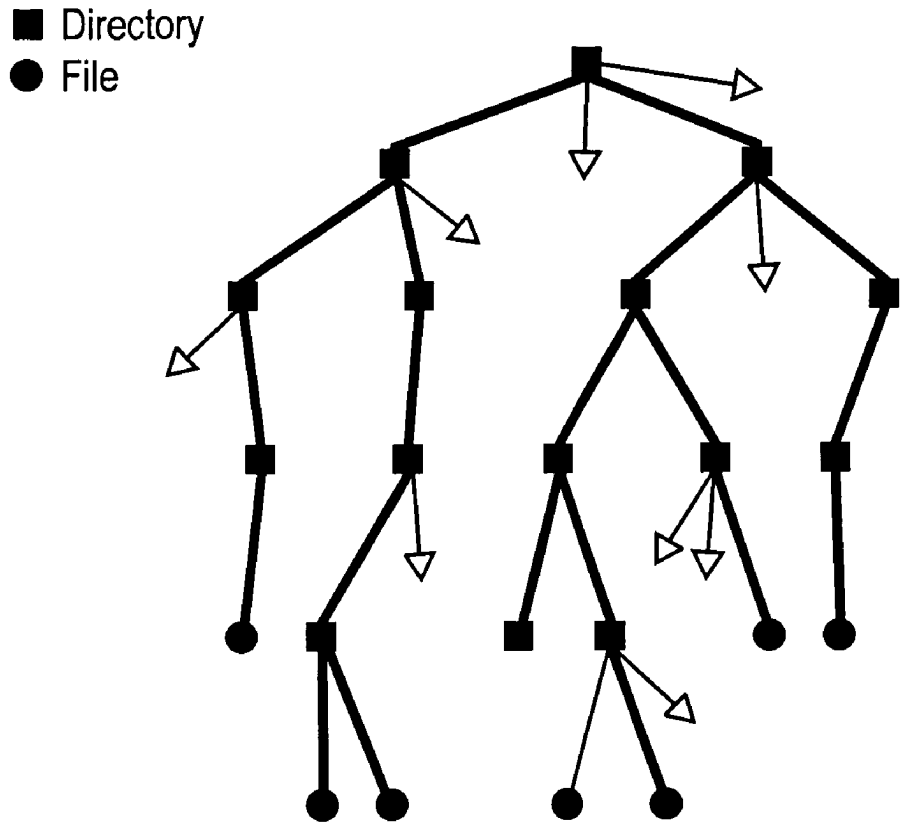


FIG. 6

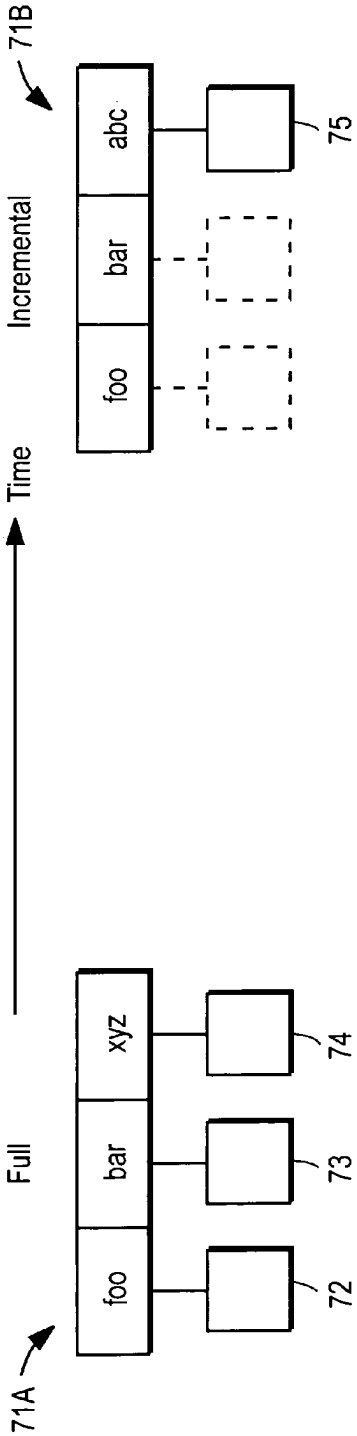

71A
Full
foo
bar
xyz
72
73
74
Time
71B
Incremental
foo
bar
abc
75

FIG. 7

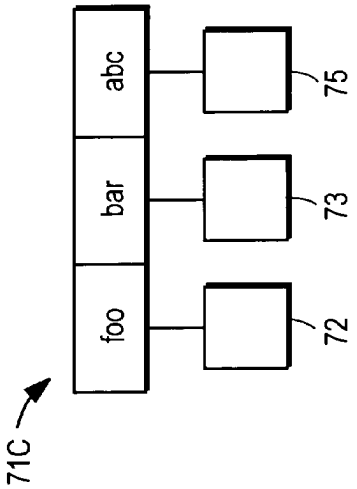

71C
foo
bar
abc
72
73
75

FIG. 8

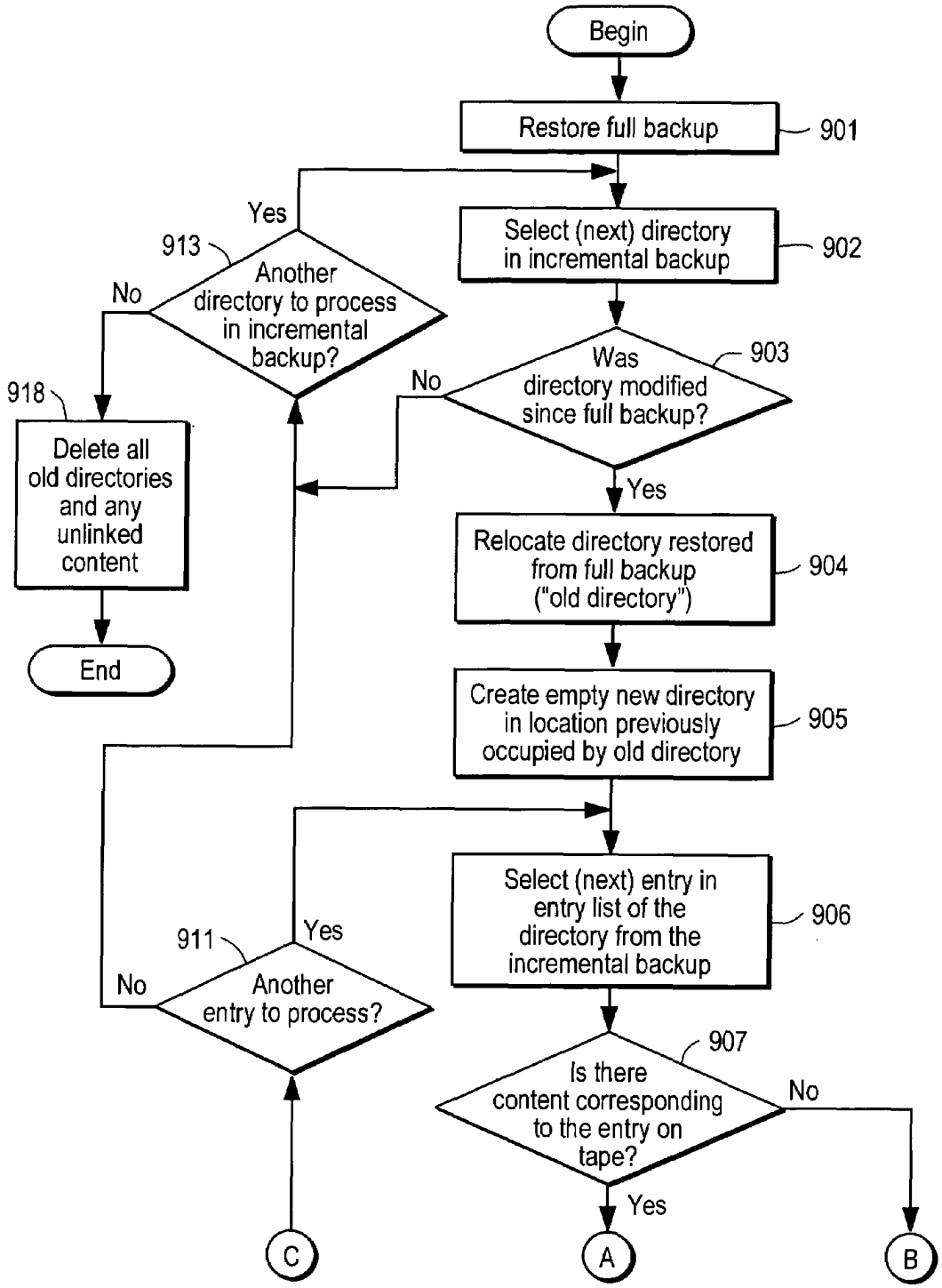

Begin
Restore full backup
901
Select (next) directory in incremental backup
902
Was directory modified since full backup?
903
No
Yes
Relocate directory restored from full backup ("old directory")
904
Create empty new directory in location previously occupied by old directory
905
Select (next) entry in entry list of the directory from the incremental backup
906
Is there content corresponding to the entry on tape?
907
No
Yes
A
B
Another entry to process?
911
Yes
No
C
Another directory to process in incremental backup?
913
Yes
No
Delete all old directories and any unlinked content
918
End

FIG. 9A

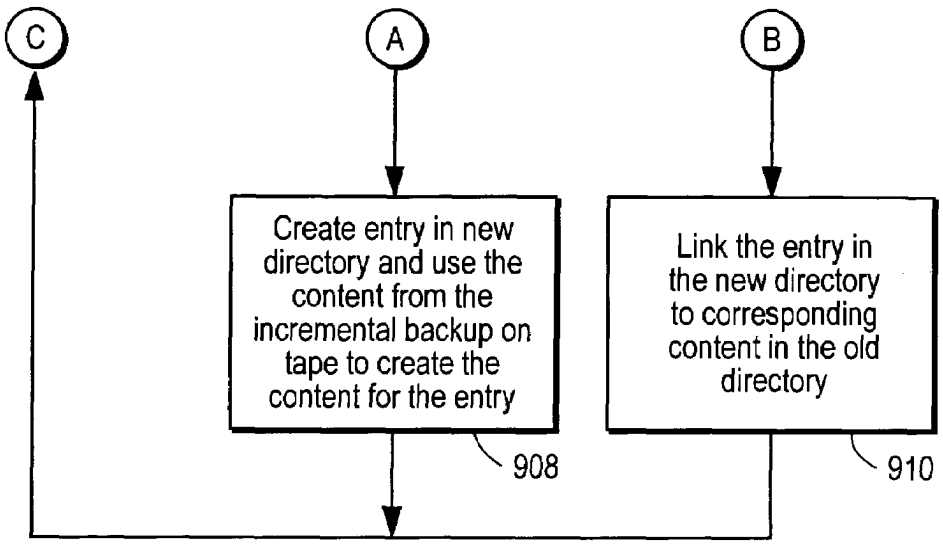

C
A
B
Create entry in new directory and use the content from the incremental backup on tape to create the content for the entry
908
Link the entry in the new directory to corresponding content in the old directory
910

FIG. 9B

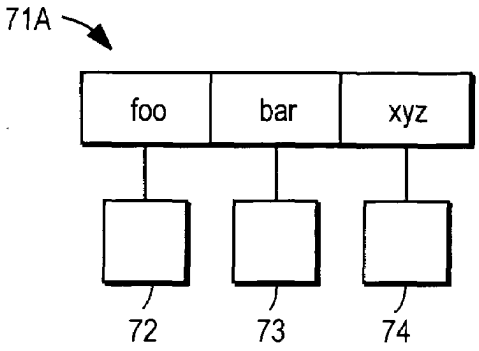


FIG. 10A

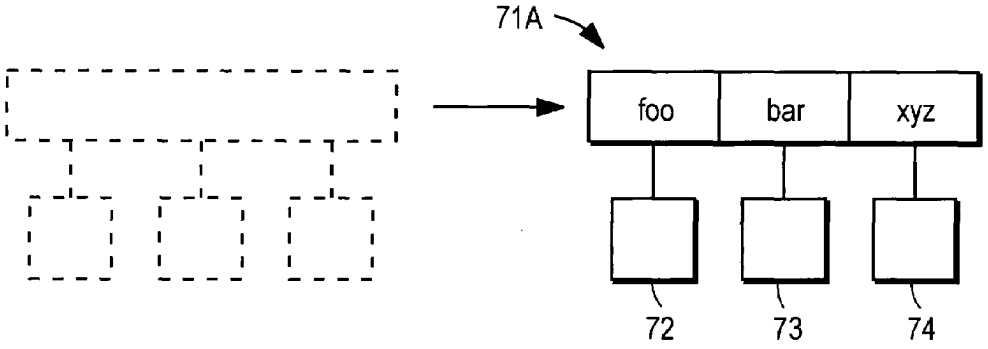


FIG. 10B

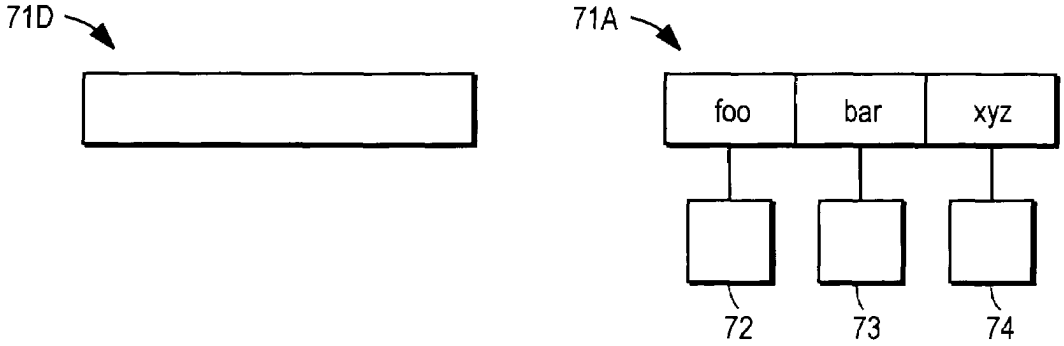


FIG. 10C

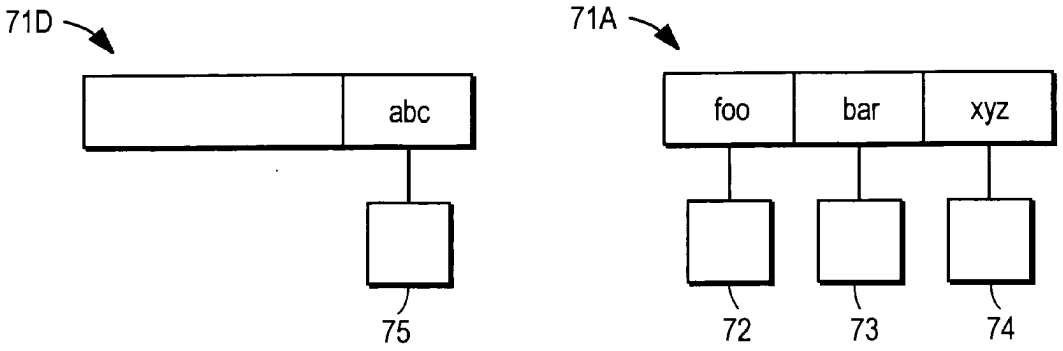


FIG. 10D

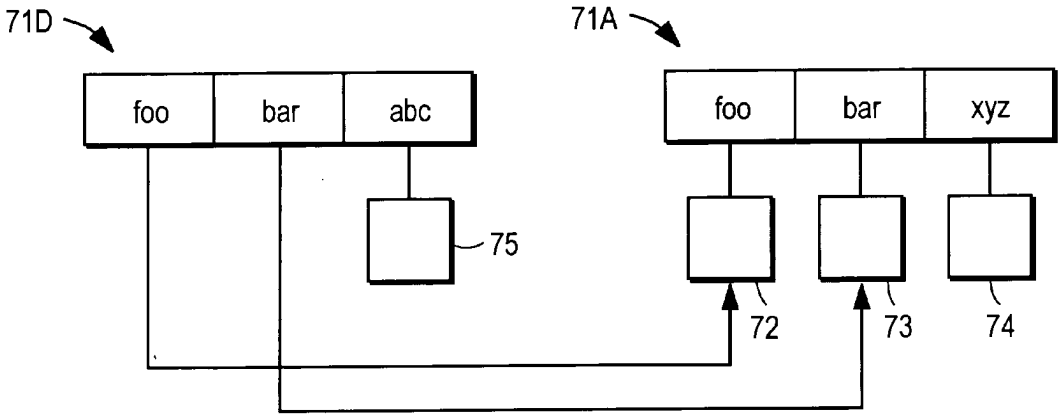


FIG. 10E

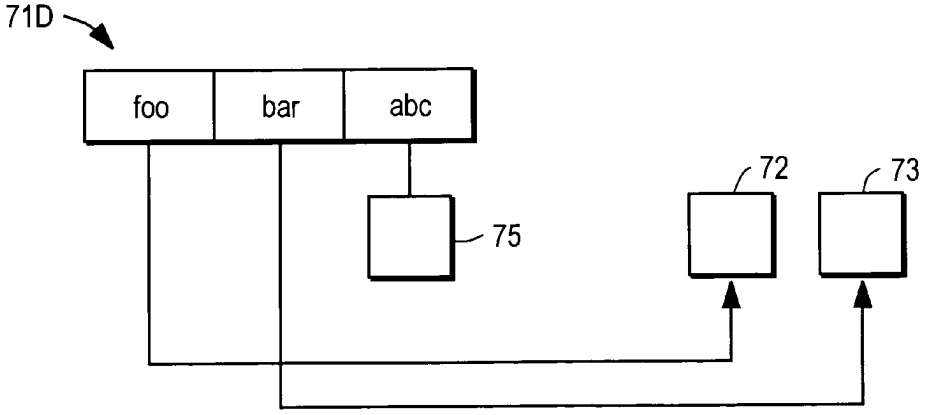


FIG. 10F

EFFICIENT TRUE IMAGE RECOVERY OF DATA FROM FULL, DIFFERENTIAL, AND INCREMENTAL BACKUPS

FIELD OF THE INVENTION

At least one embodiment of the present invention pertains to data storage systems, and more particularly, to a technique for recovering data from full and incremental backups.

BACKGROUND

Practical file systems used for mission critical computing need protection against catastrophic failures, such as system crashes and data loss. Such systems also need protection from unintended corruption of data and user/application errors. A common practice to protect data from permanent loss is to make backup copies of the data periodically. For example, users may copy the data to another file system where the data can be accessed online (typically in a read-only manner) or to backup storage devices such as tapes, where the data stays offline until mounted again. These two approaches do not necessarily exclude each other.

A simple approach to making backups of a file system is to copy the entire contents of the file system to tapes (or other non-volatile mass storage devices) every time a backup is performed. To recover the file system to its state at the time of a particular backup, the entire particular backup is simply restored back to the target file system. However, due to a characteristic typical of most file systems, this approach is not the most desirable. A typical file system does not change rapidly over time. Between two consecutive backups, only a very small percentage (e.g., 5%) of the data may have changed. Most of the data is identical between any two consecutive backups.

The aforementioned characteristic raises two problems. First, on a second backup, a significant amount of tape space tends to be allocated to data that already exists in the first backup. This decreases the efficiency of tape space and raises costs. Second, a significant amount of time tends to be spent saving the same data again in the second backup.

Based on these considerations, the concept of an incremental backup has been introduced. An incremental backup is based on and follows a full backup. After a full backup, the subsequent incremental backup will capture an image of only the data that has changed since the full backup. This process can continue indefinitely, wherein there are an unlimited number of subsequent incremental backups, each based on the previous incremental backup. On recovery, the full backup is first restored. Then the changes represented by the first incremental backup are distributed to the appropriate locations on the file system to bring the file system up to the state at the time the incremental backup was done. This process is iterated for each of the incremental backups.

Note that a similar process can be run with differential backups. Differential backups work similarly to incremental backups, except that differential backups are not always based on the most recent incremental backup. To minimize the number of backups to restore, differential backups may be based on an older incremental, differential or full backup, thus eliminating the need to retain the incremental and differential backups taken between the base backup and the new differential. The cost of this approach is that the differential backup, by aggregating multiple incremental backups into one backup, is additional system load and consumption of backup storage.

One problem with existing recovery algorithms based on full and incremental backups is that they tend to be very slow. It is desirable to have a data recovery algorithm that will efficiently apply an incremental backup to a restored full backup, to transform a file system to its state at the point of the incremental backup. This is known as a true image recovery, in which the recovered data exactly matches the state of the data at the time of backup. This contrasts with extraction recoveries, in which data is recovered from incremental or differential backups without renaming or removing the files that were renamed or removed between the full and incremental backup.

Some file systems are tree-structured. An example of such a file system is the Write Anywhere File Layout (WAFL) file system used in Filer products made by Network Appliance, Inc. of Sunnyvale, Calif. A file system is commonly made up of directories and individual files. The files usually contain the valuable data, while the directories provide a hierarchy/organization of the files. Every directory contains zero or more "entries", which can be files or sub-directories. There is commonly a root directory, under which there can be multiple sub-directories. Each sub-directory can hold its own sub-directories, and so on. This forms a tree structure organization in which files are distributed among the directories in their defined relationship. On backup and restore, not only is it desirable to recover the data in files, it is also desirable to recover the exact organization of data/files.

In some file servers, a full backup gathers all files and directories and their content from the file system and writes them out to tapes. The incremental backup that follows will only write out files and directories that are new or have been modified since the full backup. The modifications to a directory include addition, removal and renaming of files and sub-directories, as well as content update of an existing file in the directory.

In a typical tree-structured file system such as described above, for every modified directory there is a path that leads to the root directory. Therefore, to generate the full pathname on a recovery, during incremental backup all directories along that path also need to be written out to tapes. These directories might not have been modified; they are in the incremental backup only because one or more of their descendants has been modified. As a result, time is unnecessarily spent processing unmodified directories when the incremental backup is applied to the full backup during recovery.

Another problem with at least one known data recovery algorithm is that in the process of determining which entries to add, delete and keep, every entry in the new directory is compared to every entry in the old directory. The algorithm, therefore, has an order of m×n comparisons between entries in the old directory and entries in the new directory, where m is the number of entries in the old directory and n is the number of entries in the new directory. As a directory grows, the amount of time needed to process the directory during recovery increases exponentially. The algorithm, therefore, does not scale with the directory size.

What is needed, therefore, is a data recovery algorithm, based on full and incremental backups, which overcomes these and other disadvantages of the prior art.

SUMMARY OF THE INVENTION

The present invention includes a method and apparatus to recover a set of data from a full backup and an incremental

backup. In certain embodiments, the method is as follows: A first directory is restored based on the full backup. A new, empty second directory is then created, and a determination is then made of whether content corresponding to an entry is present in the incremental backup. The entry is created in the second directory. If content corresponding to the entry is present in the incremental backup, then content for the entry is created in the second directory from the incremental backup. If, however, content corresponding to the entry is not present in the incremental backup, then the entry is linked to corresponding content in the first directory.

Other aspects of the invention will be apparent from the accompanying figures and from the detailed description which follows.

BRIEF DESCRIPTION OF THE DRAWINGS

One or more embodiments of the present invention are illustrated by way of example and not limitation in the figures of the accompanying drawings, in which like references indicate similar elements and in which:

FIG. **1** illustrates a storage network configuration in which the invention can be implemented;

FIG. **2** is a block diagram of the architecture of a filer;

FIG. **3** is a block diagram of the operating system of a filer;

FIG. **4** illustrates an example of a tree-structured file system at the time of an incremental backup;

FIG. **5** illustrates an image of a portion of the file system of FIG. **4**, as captured by the incremental backup;

FIG. **6** is another illustration of the partial file system as captured by the incremental backup;

FIG. **7** illustrates a directory in a full backup and the same directory as modified in a subsequent incremental backup;

FIG. **8** illustrates the result of applying the incremental backup of FIG. **7** to the full backup of FIG. **7** during a data recovery process;

FIG. **9** shows a process of recovering data from a full backup and an incremental backup in accordance with the invention; and

FIG. **10** illustrates the process of FIG. **9** with respect to a particular directory that has been modified since the full backup.

DETAILED DESCRIPTION

A method and apparatus to recover a set of data from a full backup and an incremental or differential backup in which the data is an exact image of the system at the time of the incremental or differential backup are described. Note that in this description, references to "one embodiment" or "an embodiment" mean that the feature being referred to is included in at least one embodiment of the present invention. Further, separate references to "one embodiment" or "an embodiment" in this description do not necessarily refer to the same embodiment; however, such embodiments are also not mutually exclusive unless so stated, and except as will be readily apparent to those skilled in the art from the description. For example, a feature, structure, act, etc. described in one embodiment may also be included in other embodiments. Thus, the present invention can include a variety of combinations and/or integrations of the embodiments described herein.

There are two observations on backup format that help to solve the problems mentioned above. First, there is normally a last-modified timestamp on every directory in any backup. The timestamp is updated every time a directory is modified directly. For a directory that exists in both the full backup and an incremental backup, to determine if it has been modified since full backup, we can simply compare the timestamps. (To that end, it is important to ensure that the timestamp of a directory is updated only in response to user modifications to the contents of the directory, not in response to simply moving a directory or purely internal file system operations.) If the timestamps are different, the directory has been modified. If the timestamps are identical, the directory has not been modified. Thus, unmodified directories can be skipped in the process of applying the incremental backup during recovery.

Second, for a directory with a modified set of entries in an incremental backup, we update the restored directory in the file system with a new set of entries. At the end of the process, the restored directory should be exactly the same as the one in the incremental backup. It is not necessary to transform the old directory to reflect the new set of entries. We can simply construct the new directory from the content in the incremental backup, and replace the old directory with the new directory.

There are two types of entries in a modified directory in an incremental backup:

1. Entries that have changed since the full backup. Since these entries had changed, the incremental backup picked up the entries as well as their content. During recovery, we can recreate the entries out of the incremental backup alone. In this description, we call these entries "on-tape" entries.
2. Entries that have not changed since the full backup. Since these entries are unchanged, the copy of the content we have restored from the full backup is still valid. We simply have to pull the copy from the old directory into the new directory. In this description, we call these entries "off-tape" entries.

Despite the use of the terms "on tape" and "off tape" in this description, it is to be understood that types of storage media other than tape can be used to store the full and incremental backups in connection with the data recovery technique introduced herein.

Thus, as described further below, the data recovery technique introduced herein is as follows, according to at least one embodiment of the invention. To facilitate description, the technique will be described with respect to incremental backups. Note, however, that the technique is also applicable to differential backups in essentially the same manner as incremental backups and should be understood to apply also to differential backups wherever incremental backups are discussed.

First, the full backup is restored. Next, the following process is repeated for each directory in the set of data that was modified between the full backup and a subsequent backup, e.g., an incremental backup: The directory from the restored full backup (the "old directory") is relocated to temporary storage, and a new empty directory is created in its place. For each entry in the entry list of the directory in the incremental backup, a corresponding entry and content are created in the new directory from the incremental backup if the content is present in the incremental backup (i.e., the content is "on tape"). If, however, content for the entry is not present in the incremental backup (i.e., the content is "off tape"), then the entry is created in the new directory and linked to corresponding content in the restored old directory (from the restored full backup). All of the entries in the directory from the incremental backup are processed in this way, and the process is then repeated for all directories in the

incremental backup. After all of the directories in the incremental backup have been processed in this way, all of the directories from the restored full backup are deleted, along with any of their contents which are not linked to an entry in another directory.

If the old directory (the directory restored from the full backup) has entries or content that was removed between backups, no action is taken on those entries. At the conclusion of processing, the original directory contains only old or eliminated data. Thus, without complex calculation, the original directory can be removed, which automatically and completely cleans up the state of the restored system without complicated comparisons and lookups of directory entries.

In contrast with the existing data recovery algorithms discussed above, to restore a modified directory this technique requires no more than n lookups of directory entries between the directory in the incremental backup and the old directory in the full backup, even when m and n are both greater than one (where m and n represent the number of entries in the old and modified directory, respectively). The technique introduced herein, therefore, is much more scalable than the prior technique. In addition, the technique introduced herein, with fewer lookups and computations, tends to be more reliable than prior restoration techniques.

FIG. **1** illustrates an example of a network configuration in which the data recovery technique introduced herein may be employed. Of course, the technique can also be employed in various other types of environments. The illustrated system includes a filer (file server) **2**, which may implement the data recovery technique. A number of client processing systems ("clients") **1** are coupled to the filer **2** through a first network **3**, such as a LAN. Each of the clients **1** may be, for example, a conventional personal computer (PC), workstation, or the like. The filer **2** provides the clients **1** with access to files stored on a first set of mass storage devices **4**, such as magnetic or optical disks, which may be configured as one or more RAID groups.

The filer **2** is also coupled to a backup storage facility **5**, which may include one or more tapes. The filer **2** is programmed to save backup copies of its file system to backup storage facility **5** in the form of a full backup and, subsequently, one or more incremental backups. The full backup and the incremental backup(s) may or may not be stored on the same physical storage media.

In certain embodiments, the filer **2** may also be connected (e.g., through a network) to another, remote filer (not shown), such that the two filers cooperate to perform data mirroring. To facilitate explanation, this description is presented in terms of recovering data onto the same filer which created the backups, i.e., where the source is the same as the destination. However, it is to be understood that essentially the same technique can be applied where the destination is different from the source, such as to use backup data of one filer to recover data for another filer.

FIG. **2** shows the architecture of the filer **2**, according to certain embodiments of the invention. Note that certain standard and well-known components which are not germane to the present invention are not shown. The filer **2** includes one or more processors **21** and memory **22**, coupled together by a bus system **23**. The bus system **23** in FIG. **2** is an abstraction that represents any one or more separate physical buses and/or point-to-point connections, connected by appropriate bridges, adapters and/or controllers. The bus system **23**, therefore, may include, for example, a system bus, a Peripheral Component Interconnect (PCI) bus, a HyperTransport or industry standard architecture (ISA) bus, a small computer system interface (SCSI) bus, a universal serial bus (USB), or an Institute of Electrical and Electronics Engineers (IEEE) standard 1394 bus (sometimes referred to as "Firewire").

The processor **21** is the central processing unit (CPU) of the filer **2** and, thus, controls the overall operation of the filer **2**. In certain embodiments, the processor **21** accomplishes this by executing software stored in main memory **22**. The processor **21** may be, or may include, one or more programmable general-purpose or special-purpose microprocessors, digital signal processors (DSPs), programmable controllers, application specific integrated circuits (ASICs), programmable logic devices (PLDs), or the like, or a combination of such devices.

Memory **22** represents some form of random access memory (RAM), read-only memory (ROM), or both. Memory **22** stores the operating system **24** of the filer **2**. The data recovery technique introduced herein may be implemented within the operating system **24**, as described further below.

Also connected to the processor **21** through the bus system **23** are a network adapter **28**, a storage adapter **29**, and one or more internal mass storage devices **25** (not to be confused with the mass storage devices **4** or backup mass storage facility). The network adapter **28** provides the filer **2** with the ability to communicate with remote devices, such as clients and/or another filer, over a network and may be, for example, an Ethernet adapter. The storage adapter **29** allows the filer to access the external mass storage devices and may be, for example, a Fibre Channel adapter or SCSI adapter.

FIG. **3** illustrates the operating system **24** of the filer **2**, according to certain embodiments of the invention. As can be seen, the operating system **24** includes a number of layers. The core of the operating system **24** is the file system layer **31** which, among other responsibilities, executes read and write operations on the mass storage devices **4** in response to client requests, maintains directories, etc. The file system layer **31** operates on blocks of data of a predetermined size, such as 4 kbytes. The data backup and recovery techniques described herein may be implemented by a module **32** within the file system layer **31**, as shown in FIG. **3**. Also shown in FIG. **3** is the logical data path **38** from clients to mass storage devices, through the file system **31**.

Above the file system layer **31**, the operating system **24** also includes a user interface **33**, through which a network administrator or other user can control and/or configure the filer (e.g., remotely from a management station). The user interface **33** may generate a command line interface and/or a graphical user interface for this purpose.

Below the file system layer **31**, on the client side the operating system **24** includes a network layer **34** and, at the lowest level, a media access layer **35**. The network access layer **34** implements any of various protocols used to communicate with client devices, such as network file system (NFS), common Internet file system (CIFS) and/or hypertext transport protocol (HTTP). The media access layer **35** includes one or more drivers which implemented the protocols used to communicate over the network, such as Ethernet.

Below the file system layer **31** on the storage device side, the operating system **24** includes a storage access layer **36** and, at the lowest level, a driver layer **37**. The storage access layer **36** implements a disk storage protocol such as RAID, while the driver layer **37** implements a lower-level storage device access protocol, such as Fibre Channel or SCSI.

FIG. **4** shows an example of an image of a tree-structured file system, such as may be implemented by filer **2**, at the

time of an incremental backup. The nodes connecting the various branches represent files and directories (directories are shown as rectangles, and files are shown as ovals). The files and directories represented by thicker ("black") lines either have been modified or are ancestors of those which have been modified since the full backup. The branches represented by thinner ("gray") lines are files and directories that have not been modified since the full backup. The incremental backup will capture an image of only the part of the file system shown in the thicker lines. In other words, the incremental backup will contain the image of a partial file system, such as shown in FIG. **5**.

With the incremental backup alone, it is possible to reconstruct the exact image, including up-to-date file content, of the partial file system. Note that the only difference between directory structures in FIG. **4** and FIG. **5** is in the gray branches, which remain the same between the full and incremental backups. If the full backup is restored to a separate known place, all those gray branches reside in it and are in the correct (up-to-date) state. By purging all those necessary branches out of the full backup destination and inserting them in the right places, the result is a file system reflecting the image of the source file system at the time of the incremental backup. This purging and re-attaching does not necessarily have to be done after the partial file system from the incremental backup is fully erected. It could as well be done while recreating each directory in the incremental backup.

Therefore, two issues should be addressed: 1) when to insert a gray branch from the full backup, and 2) where to locate that branch on the full backup file system tree. Regarding the first issue, FIG. **6** is a more precise representation of the incremental backup of FIG. **5**. For every unmodified branch coming out of a directory, there exists in the directory an entry that does not refer to any file/directory content (inode) on the tape. Note that this description assumes a file system that is based on the use of "nodes" such as in the Unix operating system.

In other words, there are two types of directory entries in any directory of an incremental backup:

1. Off-tape entry—a directory entry that refers to an inode not in the incremental backup.
2. On-tape entry—a directory entry that refers to an inode in the incremental backup.

An off-tape entry indicates that there is a file or sub-directory in the current directory that had not been modified since the full backup, such that the incremental backup does not have the content of the file or sub-directory. For off-tape entries, their contents are obtained from the destination of the full backup (the restored full backup).

Regarding the issue of where to locate that branch on the full backup file system tree, note that the fact that a particular file, call it abc, is untouched between the full backup and the incremental backup normally implies not only that its content has not changed, but also that its parent remains the same. The source inode of the file's parent is known. With the parent source inode, the inode map (or translation table) gives the parent destination inode on the full backup destination. The name for abc on the full backup destination is also abc (the name has to remain the same, otherwise, it violates the initial assumption that abc does not change). With a handle to its parent directory and its name in the full backup destination, we can easily move the file into the current directory. If this algorithm runs on a system that does not obey the initial assumption, the algorithm is still functional, though less optimized. The unique name of the file will need to be either stored by the baseline recovery, or the incremental recovery will need to look up the name of the file by its unique descriptor.

FIGS. **7** and **8** conceptually show an example of a directory at different points in time during the backup and recovery processes. At the time of the full backup, the directory **71**A includes three entries, foo, bar, and xyz, as shown at the left side of FIG. **7**. Each entry in a directory **71**A is assigned an inode number (not shown), which indicates the location of the content for that entry. Hence, the entries foo, bar and xyz are linked to their corresponding content **72**, **73** and **74**, respectively, by their inode assignments.

The same directory at the time of the incremental backup appears as **71**B at the right side of FIG. **7**. At the time of the incremental backup, xyz has been deleted from the directory and a new entry, abc, has been added along with its content **75**. The content for foo and bar is not present in the directory **71**B in the incremental backup (i.e., it is off tape), as indicated by the dashed lines in FIG. **7**, since that content has been neither deleted nor modified since the full backup. FIG. **8** shows the result of merging incremental backup with the full backup during the data recovery process. As shown, the resulting directory **71**C includes three entries, foo, bar and abc, which are linked to their corresponding content, **72**, **73** and **75**, respectively. The resulting directory **71**C reflects the fact that xyz has been deleted from the directory and abc has been added.

The data recovery process will now be further described with reference to FIGS. **9** and **10**. FIG. **9** shows the data recovery process according to certain embodiments of the invention. Initially, the full backup is restored at **901**. Reference numerals **902** through **912** in FIG. **9** represent a subprocess performed for each directory in the file system as represented in the incremental backup. In at least one embodiment, the partial file system tree from the incremental backup is traversed directory-by-directory starting from the root directory, using a breadth-first approach. At **902** a directory in the incremental backup is selected, and at **902** it is determined whether the directory was modified since the full backup (e.g., based on its timestamp). A directory is considered to have been modified if it has been directly modified or if it is along the path from a directly modified file or directory to the root. If the directory was not modified, the process proceeds by selecting the next directory in the incremental backup, if any (**913** and **902**). If all directories have been considered (**913**), the process ends.

Reference numerals **904** through **910** represent a subprocess that is performed for each directory that was modified since the full backup. If the currently selected directory was modified (as determined at **903**), then at **904** the old directory (i.e., the directory as restored from the full backup) is relocated to a temporary location. Then, at **905** a new empty directory is created in the location previously occupied by the old directory, to replace the old directory.

Reference numerals **906** through **910** represent a subprocess that is performed for each entry in the modified directory. Associated with each modified directory in the incremental backup is an entry list (not shown) indicating the entries in that directory. At **906** an entry in the entry list is selected, and at **907** it is determined whether the content for that entry is on tape (in the incremental backup). If the content for that entry is on tape, then at **908** the entry is created in the new directory and the corresponding content from the incremental backup is used to create the content for that entry. On the other hand, if the content for the entry is off tape (not in the incremental backup), then at **910** the entry

is created in the new directory and linked to the corresponding content in the old directory (i.e., from the restored full backup).

There is special case of **910** associated with a Unix based system, which should be mentioned at this point. This is the case where the entry being processed is an off tape directory (as opposed to a file). In a Unix based system, a file can be referred to by more than one name, such that there can be more than one link to the content associated with a file. However, a directory can have only one parent. Consequently, at **910** in a system with this limitation, if the entry being processed is a directory, then the entry is actually moved from the old directory to the new directory, rather than linking the entry in the new directory to the content in the old directory.

After all entries in the entry list have been processed (**911**), the foregoing process is then repeated for each remaining directory in the incremental backup (**913**). After all directories in the incremental backup have been processed, then at **912** all old directories (i.e., the directories from the restored full backup) are deleted along with any of their contents that are not linked to an entry in another directory.

FIG. **10** uses the example from FIGS. **7** and **8** to illustrate the foregoing process for a particular directory which is modified since the full backup. The old directory **71**A resulting from restoring the full backup is shown in FIG. **10**A. After restoring the full backup, the directory **71**A is moved to a temporary location, as shown in FIG. **10**B. Next, a new empty directory **71**D is created in the location previously occupied by the old directory, as shown in FIG. **10**C. As discussed above, entry abc was created after the full backup and, therefore, is on tape. Consequently, as shown in FIG. **10**D, entry abc is created in the new directory **71**D along with its content **75** from the incremental backup. On the other hand, entries foo and bar were not modified since the full backup and, therefore, are off tape. Consequently, after creating these entries in the new directory **71**D, these entries are linked to the corresponding content from the restored full backup, as shown in FIG. **10**E. Note that entry xyz never appears in the new directory **71**D, since that entry was deleted before the incremental backup. After all of the directories in the incremental backup have been processed, the old directory **71**A is deleted, excluding any of its content which has been linked with an entry in a new directory created by this process.

In contrast with certain known data recovery algorithms, the technique introduced above restores any modified directory using no more than n lookups of directory entries between the directory as represented in the incremental backup and the directory as represented in the full backup (where m and n are the number of entries in the old and modified directory, respectively), even when m and n are both greater than one. The technique introduced above, therefore, is much more scalable than the prior technique. In addition, the technique introduced above tends to be more reliable.

Thus, a method and apparatus to recover a set of data from a full backup and an incremental or differential backup in which the data is an exact image of the system at the time of the incremental or differential backup have been described. Although the present invention has been described with reference to specific exemplary embodiments, it will be recognized that the invention is not limited to the embodiments described, but can be practiced with modification and alteration within the spirit and scope of the appended claims. Accordingly, the specification and drawings are to be regarded in an illustrative sense rather than a restrictive sense.

What is claimed is:

1. A method of recovering a set of data from a full backup and a subsequent backup, the method comprising:
- restoring a first directory based on the full backup;
- creating a new empty second directory;
- determining whether content corresponding to an entry is present in the subsequent backup;
- creating the entry in the second directory;
- if content corresponding to the entry is present in the subsequent backup, then creating content for the entry in the second directory from the subsequent backup; and
- if content corresponding to the entry is not present in the subsequent backup, then linking the entry to corresponding content in the first directory, wherein said linking includes assigning a metadata identifier of the entry.

2. A method as recited in claim **1**, further comprising deleting the first directory.

3. A method as recited in claim **2**, further comprising deleting content in the first directory if the content in the first directory is not linked to an entry in another directory.

4. A method as recited in claim **1**, wherein the first directory and second directory correspond to a directory that was modified after the full backup was created and before the subsequent backup was created.

5. A method as recited in claim **1**, wherein the set of data is a file system, and the subsequent backup is an image of at least a portion of the file system.

6. A method as recited in claim **1**, wherein the metadata identifier is an inode number.

7. A method of recovering data from a full backup and a subsequent backup, the method comprising:
- restoring the full backup from a backup storage facility, including restoring a directory which was modified after the full backup but before the subsequent backup; and
- creating an empty new directory to replace the restored directory;
- for each entry in a source directory in the subsequent backup corresponding to the restored directory,
  - determining if content corresponding to the entry is present in the subsequent backup,
  - if content corresponding to the entry is present in the subsequent backup, then creating an entry in the new directory, using the content from the subsequent backup to create the content for the created entry, and
  - if content corresponding to the entry is not present in the subsequent backup, then creating an entry in the new directory and linking the entry in the new directory to corresponding content in the restored directory, wherein linking the entry in the new directory includes assigning a metadata identifier of the entry in the new directory.

8. A method as recited in claim **7**, wherein the set of data is a file system, and the subsequent backup is an image of at least a portion of the file system.

9. A method as recited in claim **7**, further comprising deleting the restored directory.

10. A method as recited in claim **7**, wherein the set of data comprises a plurality of directories, the method further comprising repeating said creating, said determining, said using, and said linking, for each of the plurality of directories.

11. A method as recited in claim 7, wherein the metadata identifier is an inode number.

12. A method of recovering a set of data from a full backup and a subsequent backup, the method comprising:

restoring the full backup from a backup storage facility, the full backup including one or more directories, each of the directories including zero or more entries, the subsequent backup also including one or more directories, each of the directories in the subsequent backup including zero or more entries and having an associated entry list identifying the zero or more entries in the directory; and

for each directory in the subsequent backup,

determining if the directory was modified since the full backup;

if the directory was not modified since the full backup, then skipping the directory; and

if the directory was modified since the full backup, then

relocating a version of the directory restored from the full backup;

creating an empty new directory in a location previously occupied by the relocated directory, to replace the relocated directory;

for each entry in the entry list,

determining if content corresponding to the entry is present in the subsequent backup,

if content corresponding to the entry is present in the subsequent backup, then using the content from the subsequent backup to create the content in the new directory, and

if content corresponding to the entry is not present in the subsequent backup, then linking the corresponding entry in the new directory to corresponding content in the relocated directory restored from the full backup, by assigning an inode number of the entry in the new directory;

and

deleting the relocated directory.

13. A method as recited in claim 12, wherein the set of data is a file system, and the subsequent backup is an image of at least a portion of the file system.

14. A storage system comprising:

a processor;

a first storage facility comprising one or more mass storage devices storing a full backup of a set of data;

a second storage facility comprising one or more mass storage devices storing a subsequent backup of the set of data;

a memory storing instructions which, when executed by the processor, cause the storage system to perform a process of recovering the set of data from the full backup and the subsequent backup, wherein the process includes

restoring the full backup, including restoring a directory, the restored directory including one or more entries, the subsequent backup including a modified version of the directory;

creating a new directory to replace the restored directory;

for each entry in the modified version of the directory,

if corresponding content is present in the subsequent backup, then using the content from the subsequent backup to create the content in the new directory and linking the content created in the new directory to a corresponding entry in the new directory, and

if corresponding content try is not present in the subsequent backup, then linking the corresponding entry in the new directory to corresponding content in the restored directory by assigning a metadata identifier of the entry in the new directory.

15. A storage system as recited in claim 14, wherein the restoration process further comprises deleting the restored directory.

16. A storage system as recited in claim 14, wherein the set of data is a file system, and the subsequent backup is an image of at least a portion of the file system.

17. A storage system as recited in claim 14, wherein the storage system is a file server and the set of data is a file system.

18. A storage system as recited in claim 14, wherein the metadata identifier is an inode number.

* * * * *

intel

Extensible Firmware Interface Specification

Version 1.10

December 1, 2002

1 Introduction

This *Extensible Firmware Interface* (hereafter known as EFI) *Specification* describes an interface between the operating system (OS) and the platform firmware. The interface is in the form of data tables that contain platform-related information, and boot and runtime service calls that are available to the OS loader and the OS. Together, these provide a standard environment for booting an OS.

The EFI specification is designed as a pure interface specification. As such, the specification defines the set of interfaces and structures that platform firmware must implement. Similarly, the specification defines the set of interfaces and structures that the OS may use in booting. How either the firmware developer chooses to implement the required elements or the OS developer chooses to make use of those interfaces and structures is an implementation decision left for the developer.

The intent of this specification is to define a way for the OS and platform firmware to communicate only information necessary to support the OS boot process. This is accomplished through a formal and complete abstract specification of the software-visible interface presented to the OS by the platform and firmware.

Using this formal definition, a shrink-wrap OS intended to run on Intel® architecture-based platforms will be able to boot on a variety of system designs without further platform or OS customization. The definition will also allow for platform innovation to introduce new features and functionality that enhance platform capability without requiring new code to be written in the OS boot sequence.

Furthermore, an abstract specification opens a route to replace legacy devices and firmware code over time. New device types and associated code can provide equivalent functionality through the same defined abstract interface, again without impact on the OS boot support code.

The EFI specification is primarily intended for the next generation of 32-bit Intel architecture (IA-32) and Itanium®-based computers. Thus, the specification is applicable to a full range of hardware platforms from mobile systems to servers. The specification provides a core set of services along with a selection of protocol interfaces. The selection of protocol interfaces can evolve over time to be optimized for various platform market segments. At the same time the specification allows maximum extensibility and customization abilities for OEMs to allow differentiation. In this, the purpose of EFI is to define an evolutionary path from the traditional "PC-AT"-style boot world into a legacy-API free environment.

1.1 EFI Driver Model Extensions

Access to boot devices is provided through a set of protocol interfaces. The *EFI 1.02 Specification* describes these protocol interfaces in detail. However, it does not describe how these protocol interfaces are produced by the system firmware. The *EFI 1.10 Specification* includes extensions to the *EFI 1.02 Specification* that describe the *EFI Driver Model* along with additional protocol interfaces that provide access to a richer set of boot devices. One purpose of the *EFI Driver Model* is to provide a replacement for "PC-AT"-style option ROMs. It is important to point out that drivers written to the *EFI Driver Model* are designed to access boot devices in the preboot environment. They are not designed to replace the high performance OS specific drivers. The *EFI 1.10 Specification* is designed to be backward compatible with the *EFI 1.02 Specification*. This means that any EFI applications or drivers written to the *EFI 1.02 Specification* will continue to function on system firmware that complies with the *EFI 1.10 Specification*.

The *EFI Driver Model* is designed to support the execution of modular pieces of code, also known as drivers that run in the preboot environment. These drivers may manage or control hardware buses and devices on the platform or they may provide some software derived platform specific service.

The *EFI Driver Model* is designed to extend the *EFI Specification* in a way that supports device drivers and bus drivers. These extensions are provided in the form of new protocols, new boot services, and updated EFI boot services that are backward compatible with their original versions. The *EFI Driver Model* also contains information required by EFI driver writers to design and implement any combination of bus drivers and device drivers that a platform may need to boot an EFI compliant OS.

The *EFI Driver Model* is designed to be generic and can be adapted to any type of bus or device. The *EFI 1.10 Specification* describes how to write PCI bus drivers, PCI device drivers, USB bus drivers, USB device drivers, and SCSI drivers. Additions details are provided that allow EFI drivers to be stored in PCI option ROMs while maintaining compatibility with legacy option ROM images.

One of the design goals in the *EFI 1.10 Specification* is to keep the driver images as small as possible. However, if a driver is required to support multiple processor architectures, a driver object file would have to be shipped for each supported processor architecture. To address this space issue, the *EFI 1.10 Specification* also defines the *EFI Byte Code Virtual Machine*. An EFI driver can be compiled into a single EFI Byte Code Virtual Machine object file. EFI 1.10 complaint firmware must contain an EFI Byte Code interpreter. This allows a single EFI Byte Code object file to be shipped that supports multiple processor architectures. Another space saving technique is the use of compression. The *EFI 1.10 Specification* defines compression and decompression algorithms that may be used to reduce the size of EFI drivers, and thus reduce the overhead when EFI drivers are stored in ROM devices.

The information contained in the *EFI 1.10 Specification* can be used by OSVs, IHVs, OEMs, and firmware vendors to design and implement EFI firmware, EFI drivers that produce standard protocol interfaces, and EFI operating system loaders that can be used to boot EFI compliant operating systems.

1.2 Overview

This EFI 1.10 Specification is organized as listed in Table 1-1.

Table 1-1. Organization of the EFI Specification

Chapter/Appendix	Description
1. Introduction	Introduces the EFI Specification and topics related to using the specification.
2. Overview	Describes the major components of EFI, including the boot manager, firmware core, calling conventions, protocols, and requirements.
3. Boot Manager	Describes the boot manager, which is used to load EFI drivers and EFI applications.
4. EFI System Table	Describes the EFI System Table that is passed to every EFI driver and EFI application.
5. Services — Boot Services	Contains the definitions of the fundamental services that are present in an EFI-compliant system before an OS is booted.
6. Services — Runtime Services	Contains definitions for the fundamental services that are present in an EFI-compliant system before and after an OS is booted.
7. Protocols — EFI Loaded Image	Defines the EFI Loaded Image Protocol that describes an EFI image that has been loaded into memory.
8. Protocols — Device Path Protocol	Defines the device path protocol and provides the information needed to construct and manage device paths in the EFI environment.
9. Protocols — EFI Driver Model	Describes a generic driver model for EFI. This includes the set of services and protocols that apply to every bus and device type. These protocols include the Driver Binding Protocol, the Platform Driver Override Protocol, the Bus Specific Driver Override Protocol, the Driver Diagnostics Protocol, the Driver Configuration Protocol, and the Component Name Protocol.
10. Protocols — Console Support	Defines the Console I/O protocols, which handle input and output of text-based information intended for the system user while executing in the EFI boot services environment. These protocols include the Simple Input Protocol, the Simple Text Output Protocol, the Universal Graphics Adapter (UGA) Protocol, the Simple Pointer Protocol, and the Serial I/O Protocol.

continued

Table 1-1. Organization of the EFI Specification (continued)

Chapter/Appendix	Description
11. Protocols — Bootable Image Support	Defines the protocols that provide access to bootable images while executing in the EFI boot services environment. It also describes the supported disk layouts including MBR, El Torito, and the Guided Partition Table (GPT). These protocols include the Load File Protocol, the Simple File System Protocol, the Disk I/O Protocol, the Block I/O Protocol, and the Unicode Collation Protocol.
12. Protocols — PCI Bus Support	Defines PCI Bus Drivers, PCI Device Drivers, and PCI Option ROM layouts. The protocols described include the PCI Root Bridge I/O Protocol and the PCI I/O Protocol.
13. Protocols — SCSI Bus Support	Defines the SCSI Pass Thru Protocol that is used to abstract access to a SCSI channel that is produced by a SCSI host controller.
14. Protocols — USB Support	Defines USB Bus Drivers and USB Device Drivers. The protocols described include the USB Host Controller Protocol and the USB I/O Protocol.
15. Protocols — Network Support	Defines the protocols that provide access to network devices while executing in the EFI boot services environment. These protocols include the Simple Network Protocol, the PXE Base Code Protocol, and the Boot Integrity services (BIS) Protocol.
16. Protocols — Debugger Support	An optional set of protocols that provide the services required to implement a source level debugger for the EFI environment. The EFI DebugPort Protocol provides services to communicate with a remote debug host. The Debug Support Protocol provides services to hook processor exceptions, save the processor context, and restore the processor context. These protocols can be used in the implementation of a debug agent on the target system that interacts with the remote debug host.
17. Protocols — Compression Algorithm Specification	Describes in detail the EFI compression/decompression algorithm, as well as the EFI Decompress Protocol. The EFI Decompress Protocol provides a standard decompression interface for use at boot time. The EFI Decompress Protocol is used by a PCI Bus Driver to decompress EFI drivers stored in PCI Option ROMs.
18. Protocols — Device I/O Protocol	Defines the Device I/O protocol, which is used by code running in the EFI boot services environment to access memory and I/O.

continued

Table 1-1. Organization of the EFI Specification (continued)

Chapter/Appendix	Description
19. EFI Byte Code Virtual Machine	Defines the EFI Byte Code virtual processor and its instruction set. It also defines how EBC object files are loaded into memory, and the mechanism for transitioning from native code to EBC code and back to native code. The information in this document is sufficient to implement an EFI Byte Code interpreter, an EFI Byte Code compiler, and an EFI Byte Code linker.
A. GUID and Time Formats	Explains format of EFI GUIDs (Guaranteed Unique Identifiers).
B. Console	Describes the requirements for a basic text-based console required by EFI-conformant systems to provide communication capabilities.
C. Device Path Examples	Examples of use of the data structures that defines various hardware devices to the EFI boot services.
D. Status Codes	Lists success, error, and warning codes returned by EFI interfaces.
E. 32/64-Bit UNDI Specification	This appendix defines the 32/64-bit H/W and S/W Universal Network Driver Interfaces (UNDIs).
F. Using the Simple Pointer Protocol	This appendix provides the suggested usage of the Simple Pointer Protocol.
G. Using the EFI SCSI Pass Thru Protocol	This appendix provides an example on how the SCSI Pass Thru Protocol can be used.
H. Compression Source Code	The C source code to an implementation of the EFI Compression Algorithm.
I. Decompression Source Code	The C source code to an implementation of the EFI Decompression Algorithm.
J. EFI Byte Code Virtual Machine Opcode Summary	A summary of the opcodes in the instruction set of the EFI Byte Code Virtual Machine.
K. Alphabetic Function List	Lists all EFI interface functions alphabetically.
References	Lists all necessary and/or useful specifications, web sites, and other documentation that is referenced in this EFI Specification.
Glossary	Briefly describes terms defined or referenced by this specification.
Index	Provides an index to the key terms and concepts in the specification.

1.3 Goals

The "PC-AT" boot environment presents significant challenges to innovation within the industry. Each new platform capability or hardware innovation requires firmware developers to craft increasingly complex solutions, and often requires OS developers to make changes to their boot code before customers can benefit from the innovation. This can be a time-consuming process requiring a significant investment of resources.

The primary goal of the EFI specification is to define an alternative boot environment that can alleviate some of these considerations. In this goal, the specification is similar to other existing boot specifications. The main properties of this specification and similar solutions can be summarized by these attributes:

- *Coherent, scalable platform environment.* The specification defines a complete solution for the firmware to completely describe platform features and surface platform capabilities to the OS during the boot process. The definitions are rich enough to cover the full range of contemporary Intel architecture-based system designs.
- *Abstraction of the OS from the firmware.* The specification defines interfaces to platform capabilities. Through the use of abstract interfaces, the specification allows the OS loader to be constructed with far less knowledge of the platform and firmware that underlie those interfaces. The interfaces represent a well-defined and stable boundary between the underlying platform and firmware implementation and the OS loader. Such a boundary allows the underlying firmware and the OS loader to change provided both limit their interactions to the defined interfaces.
- *Reasonable device abstraction free of legacy interfaces.* "PC-AT" BIOS interfaces require the OS loader to have specific knowledge of the workings of certain hardware devices. This specification provides OS loader developers with something different—abstract interfaces that make it possible to build code that works on a range of underlying hardware devices without having explicit knowledge of the specifics for each device in the range.
- *Abstraction of Option ROMs from the firmware.* This specification defines interfaces to platform capabilities including standard bus types such as PCI, USB, and SCSI. The list of supported bus types may grow over time, so a mechanism to extend to future bus types is included. These defined interfaces and the ability to extend to future bus types are components of the *EFI Driver Model*. One purpose of the *EFI Driver Model* is to solve a wide range of issues that are present in existing "PC-AT" option ROMs. Like OS loaders, drivers use the abstract interfaces so device drivers and bus drivers can be constructed with far less knowledge of the platform and firmware that underlie those interfaces.
- *Architecturally shareable system partition.* Initiatives to expand platform capabilities and add new devices often require software support. In many cases, when these platform innovations are activated before the OS takes control of the platform, they must be supported by code that is specific to the platform rather than to the customer's choice of OS. The traditional approach to this problem has been to embed code in the platform during manufacturing (for example, in flash memory devices). Demand for such persistent storage is increasing at a rapid rate. This specification defines persistent store on large mass storage media types for use by platform support code extensions to supplement the traditional approach. The definition of how this works is made clear in the

specification to ensure that firmware developers, OEMs, operating system vendors, and perhaps even third parties can share the space safely while adding to platform capability.

Defining a boot environment that delivers these attributes could be accomplished in many ways. Indeed several alternatives, perhaps viable from an academic point of view, already existed at the time this specification was written. These alternatives, however, typically presented high barriers to entry given the current infrastructure capabilities surrounding Intel architecture platforms. This specification is intended to deliver the attributes listed above while also recognizing the unique needs of an industry that has considerable investment in compatibility and a large installed base of systems that cannot be abandoned summarily. These needs drive the requirements for the additional attributes embodied in this specification:

- *Evolutionary, not revolutionary.* The interfaces and structures in the specification are designed to reduce the burden of an initial implementation as much as possible. While care has been taken to ensure that appropriate abstractions are maintained in the interfaces themselves, the design also ensures that reuse of BIOS code to implement the interfaces is possible with a minimum of additional coding effort. In other words, on IA-32 platforms the specification can be implemented initially as a thin interface layer over an underlying implementation based on existing code. At the same time, introduction of the abstract interfaces provides for migration away from legacy code in the future. Once the abstraction is established as the means for the firmware and OS loader to interact during boot, developers are free to replace legacy code underneath the abstract interfaces at leisure. A similar migration for hardware legacy is also possible. Since the abstractions hide the specifics of devices, it is possible to remove underlying hardware, and replace it with new hardware that provides improved functionality, reduced cost, or both. Clearly this requires that new platform firmware be written to support the device and present it to the OS loader via the abstract interfaces. However, without the interface abstraction, removal of the legacy device might not be possible at all.
- *Compatibility by design.* The design of the system partition structures also preserves all the structures that are currently used in the "PC-AT" boot environment. Thus it is a simple matter to construct a single system that is capable of booting a legacy OS or an EFI-aware OS from the same disk.
- *Simplifies addition of OS-neutral platform value-add.* The specification defines an open extensible interface that lends itself to the creation of platform "drivers." These may be analogous to OS drivers, providing support for new device types during the boot process, or they may be used to implement enhanced platform capabilities like fault tolerance or security. Furthermore this ability to extend platform capability is designed into the specification from the outset. This is intended to help developers avoid many of the frustrations inherent in trying to squeeze new code into the traditional BIOS environment. As a result of the inclusion of interfaces to add new protocols, OEMs or firmware developers have an infrastructure to add capability to the platform in a modular way. Such drivers may potentially be implemented using high level coding languages because of the calling conventions and environment defined in the specification. This in turn may help to reduce the difficulty and cost of innovation. The option of a system partition provides an alternative to nonvolatile memory storage for such extensions.
- *Built on existing investment.* Where possible, the specification avoids redefining interfaces and structures in areas where existing industry specifications provide adequate coverage. For example, the ACPI specification provides the OS with all the information

US 20040123201A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2004/0123201 A1

Nguyen et al. (43) Pub. Date: Jun. 24, 2004

(54) ON-DIE MECHANISM FOR HIGH-RELIABILITY PROCESSOR

(76) Inventors: **Hang T. Nguyen**, Tempe, AZ (US); **Steven J. Tu**, Phoenix, AZ (US); **Alexander J. Honcharik**, Holden, MA (US); **Sujat Jamil**, Chandler, AZ (US)

Correspondence Address:
Leo V. Novakoski
BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP
Seventh Floor
12400 Wilshire Boulevard
Los Angeles, CA 90025-1026 (US)

(21) Appl. No.: **10/324,957**

(22) Filed: **Dec. 19, 2002**

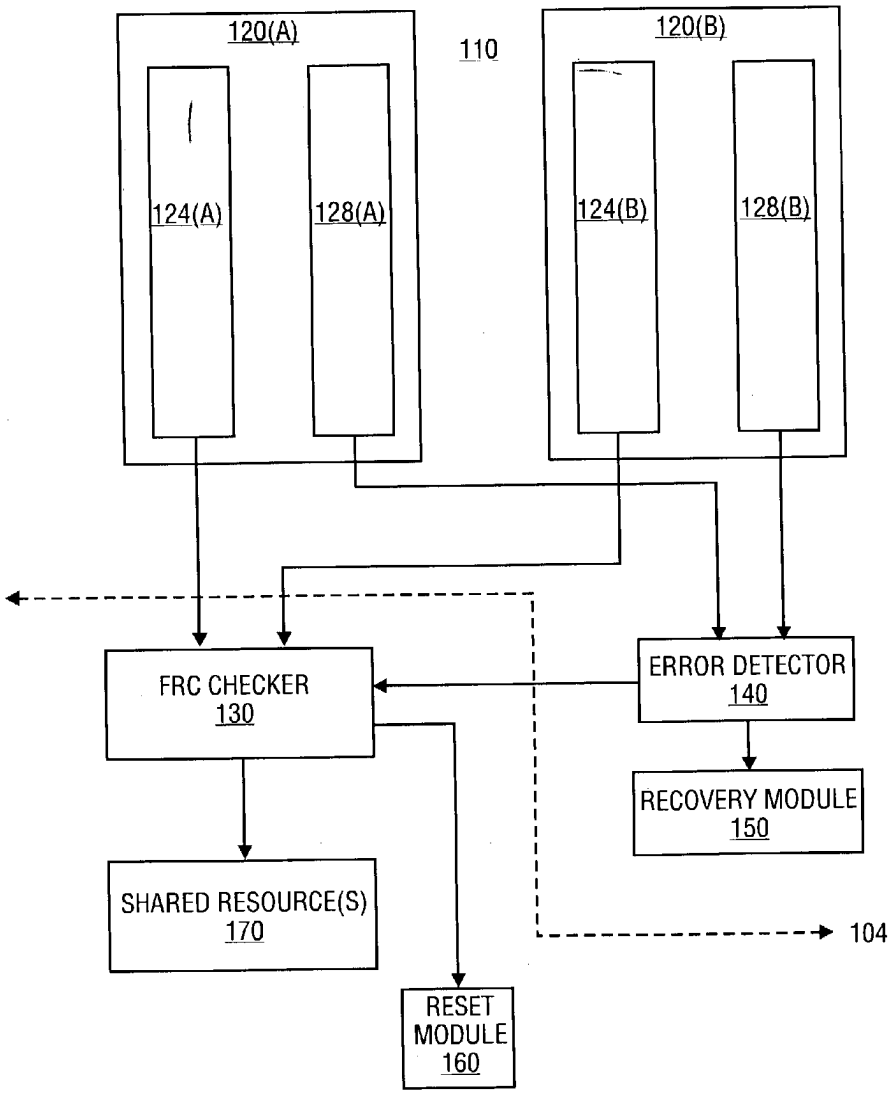
Publication Classification

(51) **Int. Cl.**[7] **G01R 31/28**; G06F 11/00

(52) **U.S. Cl.** **714/736**

(57) **ABSTRACT**

A processor includes first and second execution cores that operate in a redundant (FRC) mode, an FRC check unit to compare results from the first and second execution cores, and an error check unit to detect recoverable errors in the first and second cores. The error detector disables the FRC checker, responsive to detection of a recoverable error. A multi-mode embodiment of the processor implements a multi-core mode in addition to the FRC mode. An arbitration unit regulates access to resources shared by the first and second execution cores in multi-core mode. The FRC checker is located proximate to the arbitration unit in the multi-mode embodiment.



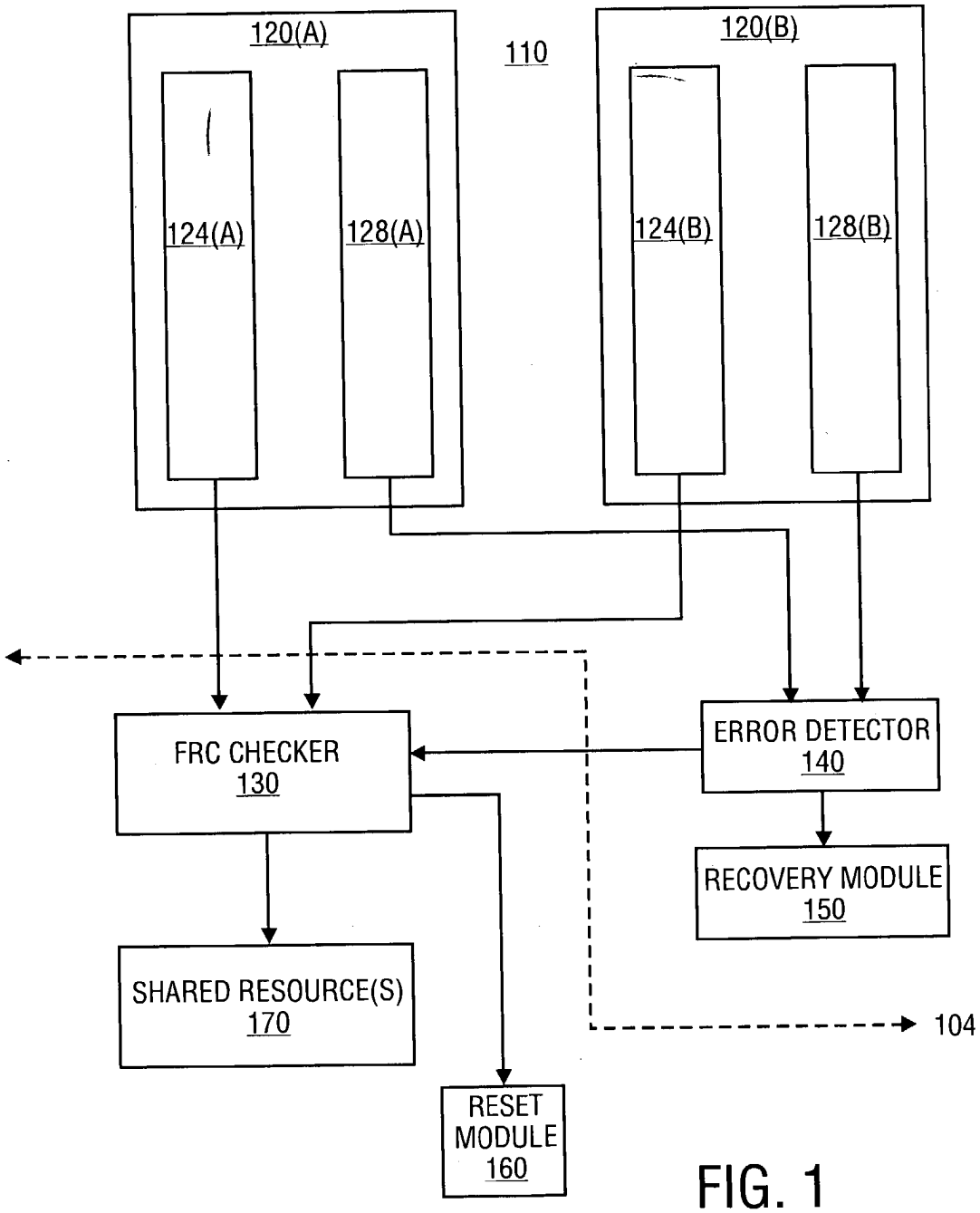

120(A)
110
120(B)
124(A)
128(A)
124(B)
128(B)
FRC CHECKER
130
ERROR DETECTOR
140
RECOVERY MODULE
150
SHARED RESOURCE(S)
170
104
RESET
MODULE
160

FIG. 1

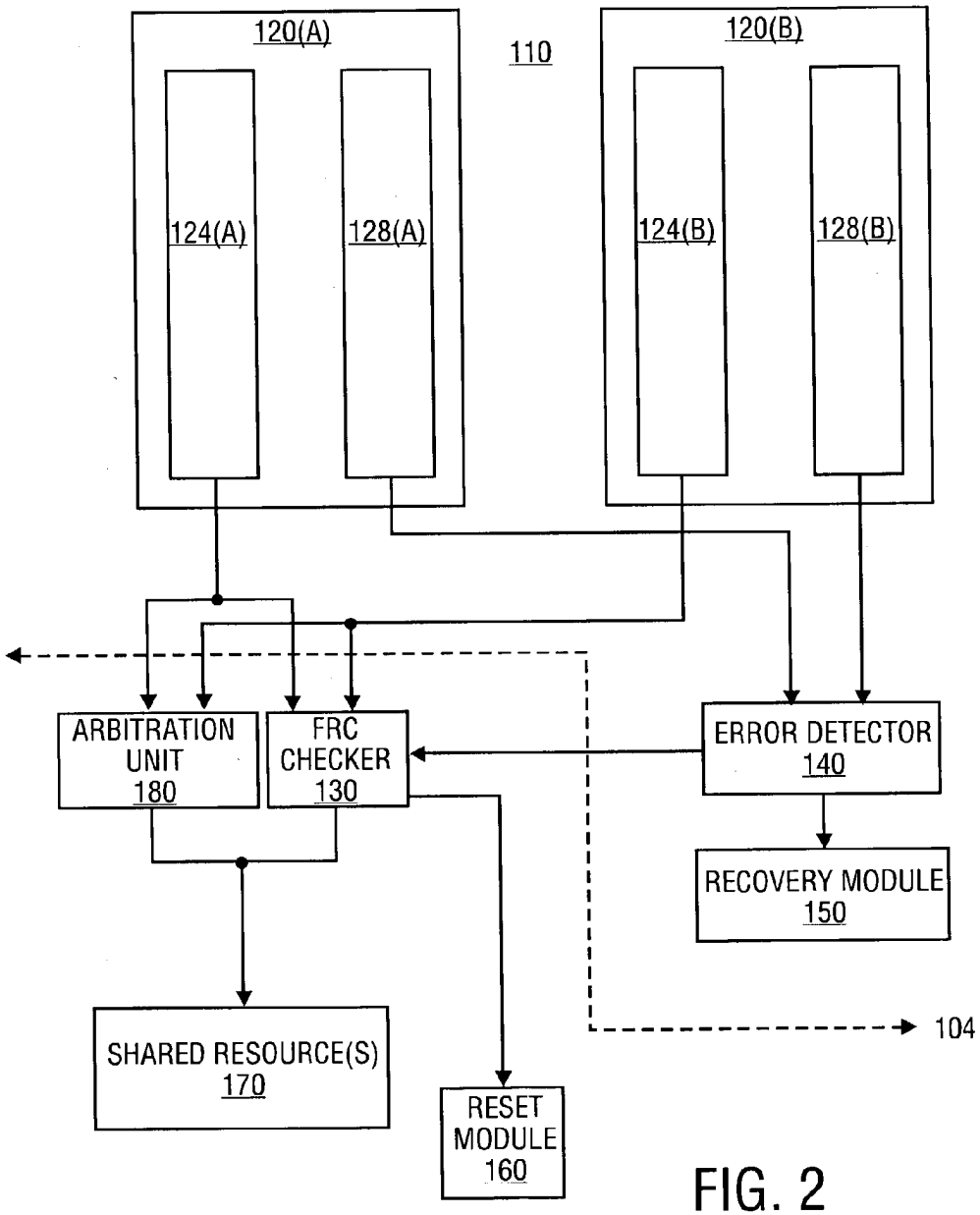

120(A)
110
120(B)
124(A)
128(A)
124(B)
128(B)
ARBITRATION UNIT 180
FRC CHECKER 130
ERROR DETECTOR 140
RECOVERY MODULE 150
SHARED RESOURCE(S) 170
RESET MODULE 160
104

FIG. 2

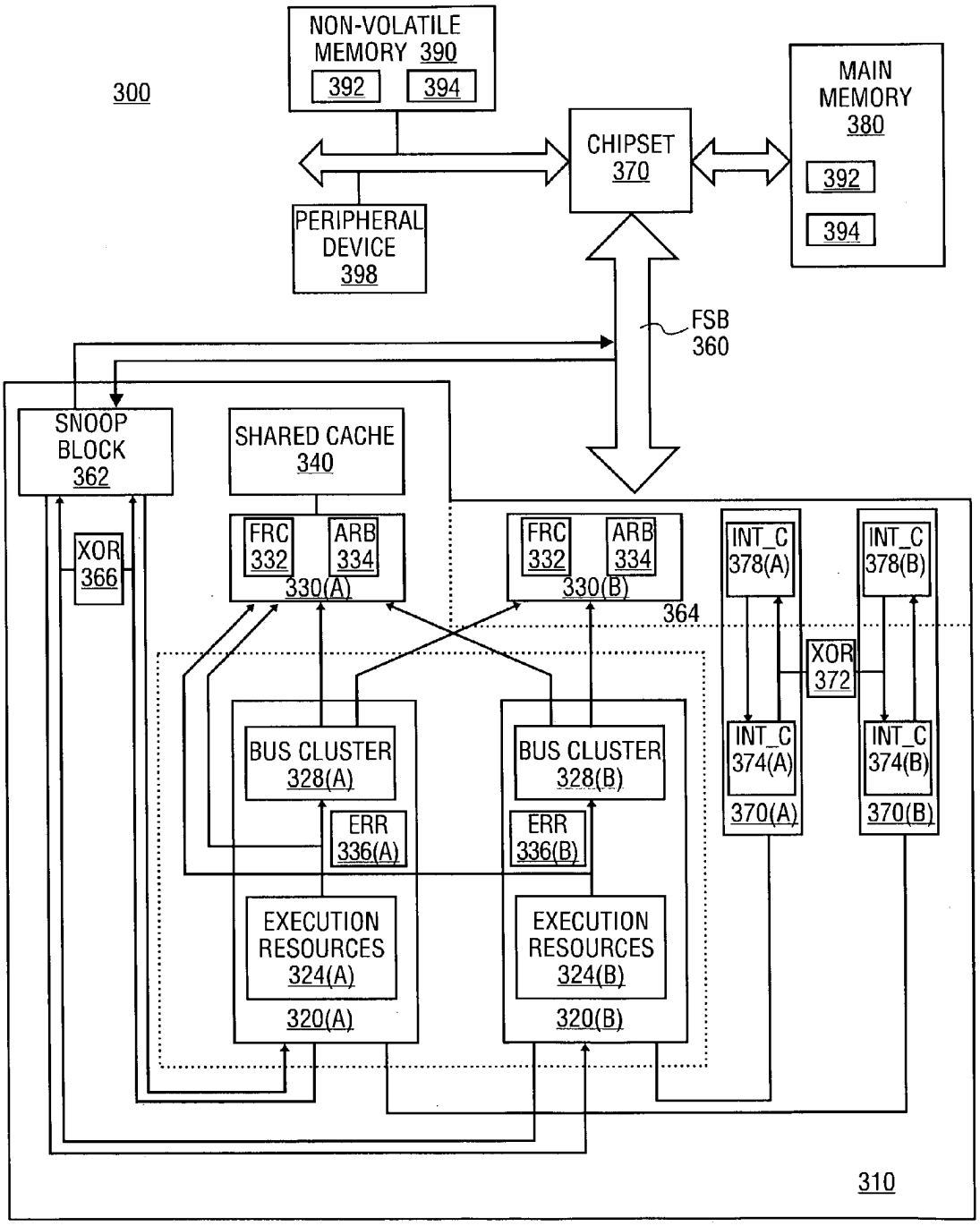

300
NON-VOLATILE MEMORY 390
392
394
MAIN MEMORY 380
392
394
CHIPSET 370
PERIPHERAL DEVICE 398
FSB 360
SNOOP BLOCK 362
SHARED CACHE 340
XOR 366
FRC 332
ARB 334
330(A)
FRC 332
ARB 334
330(B)
364
INT_C 378(A)
INT_C 378(B)
XOR 372
INT_C 374(A)
INT_C 374(B)
370(A)
370(B)
BUS CLUSTER 328(A)
BUS CLUSTER 328(B)
ERR 336(A)
ERR 336(B)
EXECUTION RESOURCES 324(A)
EXECUTION RESOURCES 324(B)
320(A)
320(B)
310

FIG. 3A

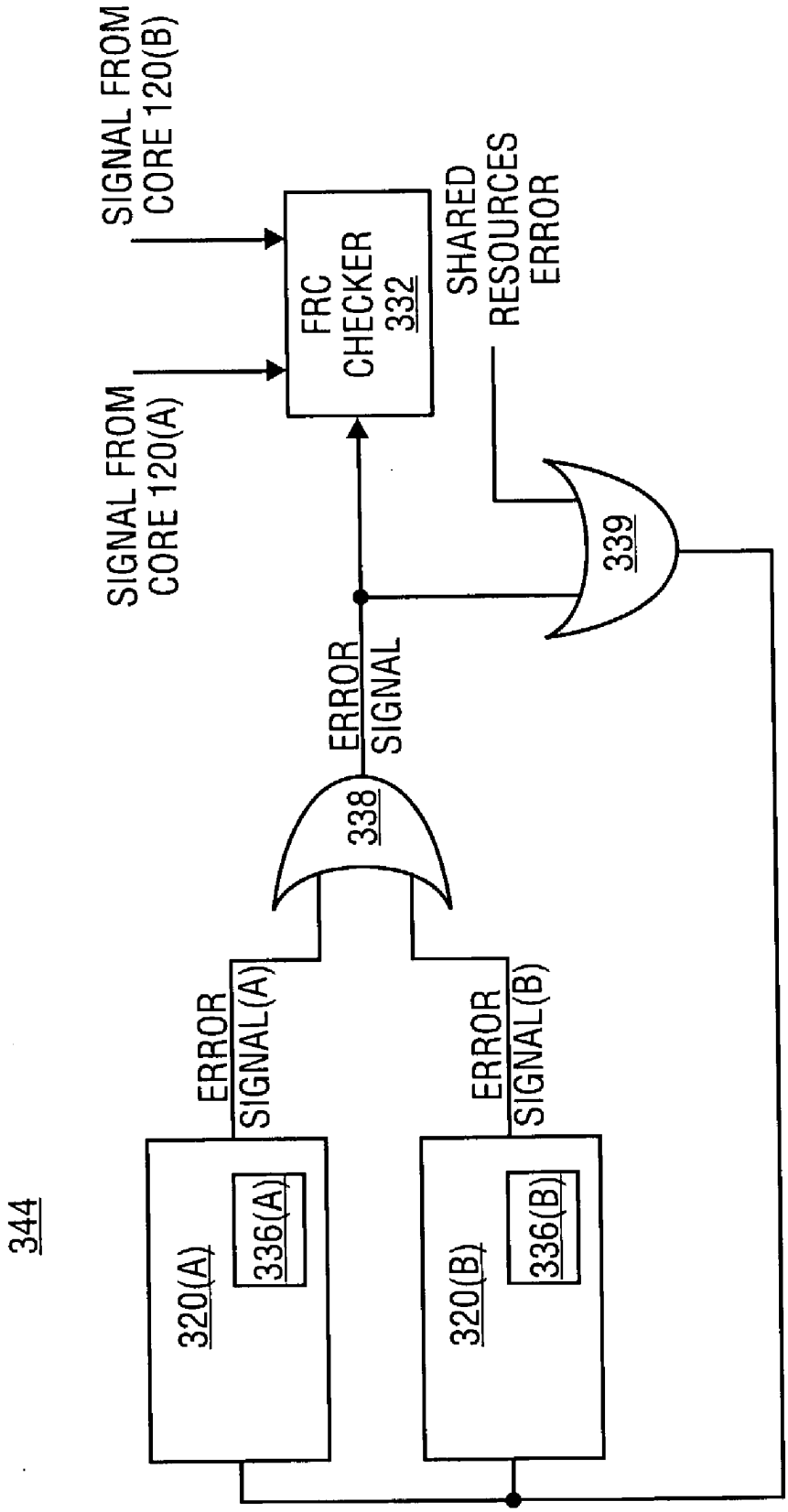

344
320(A)
336(A)
320(B)
336(B)
ERROR SIGNAL(A)
ERROR SIGNAL(B)
338
ERROR SIGNAL
339
SHARED RESOURCES ERROR
FRC CHECKER 332
SIGNAL FROM CORE 120(A)
SIGNAL FROM CORE 120(B)

FIG. 3B

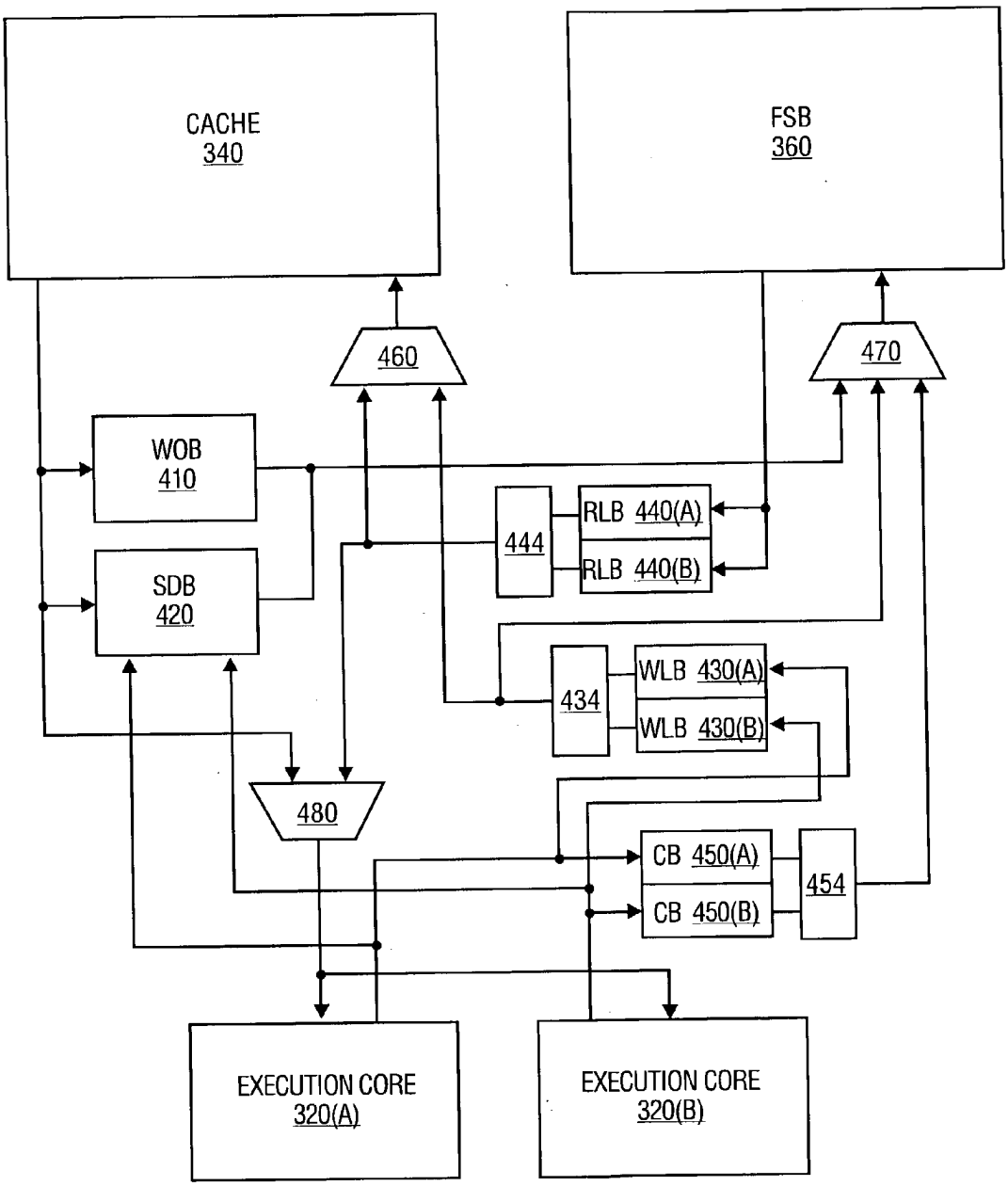

CACHE
340
FSB
360
460
470
WOB
410
SDB
420
444
RLB 440(A)
RLB 440(B)
434
WLB 430(A)
WLB 430(B)
480
CB 450(A)
CB 450(B)
454
EXECUTION CORE
320(A)
EXECUTION CORE
320(B)

FIG. 4

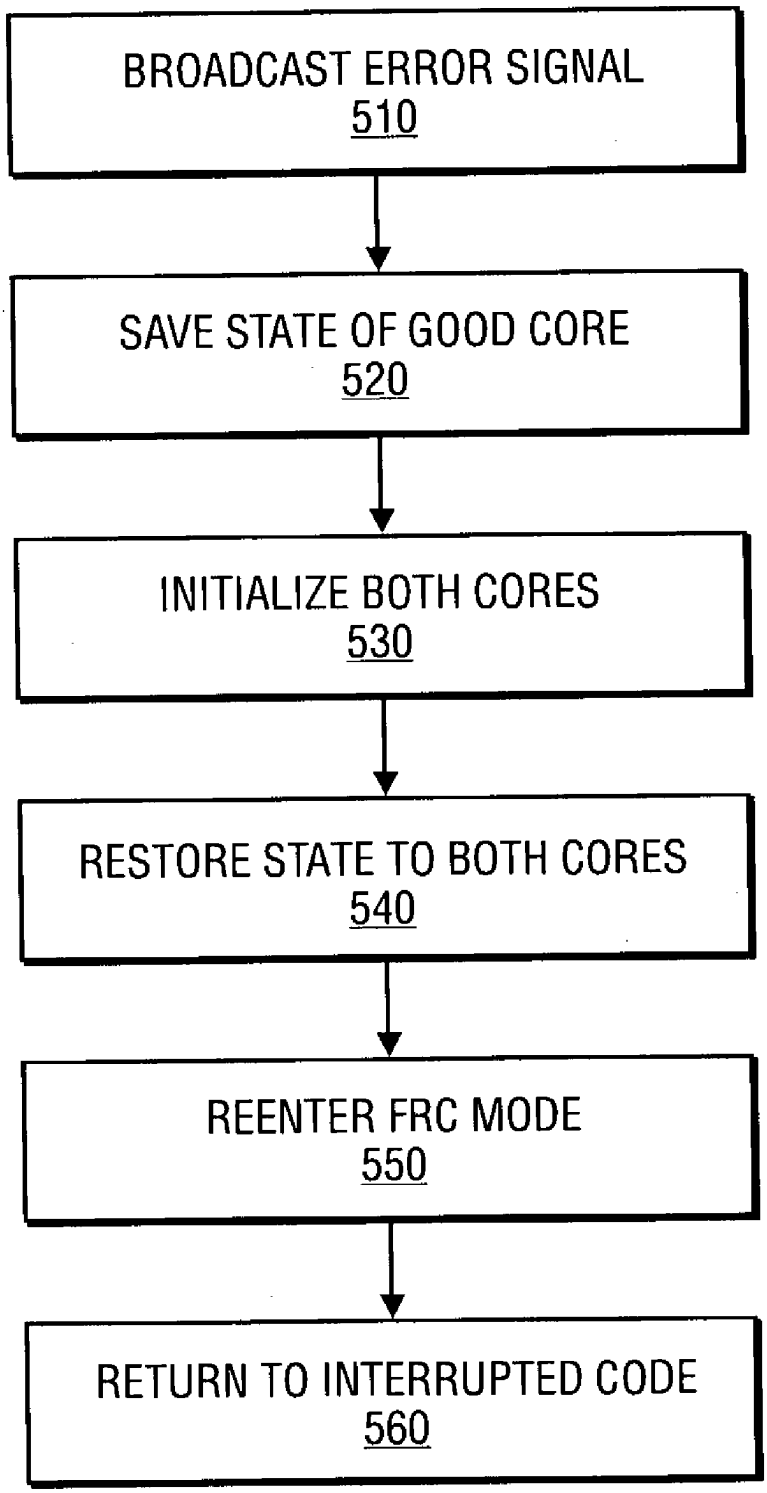

BROADCAST ERROR SIGNAL
510
SAVE STATE OF GOOD CORE
520
INITIALIZE BOTH CORES
530
RESTORE STATE TO BOTH CORES
540
REENTER FRC MODE
550
RETURN TO INTERRUPTED CODE
560

FIG. 5

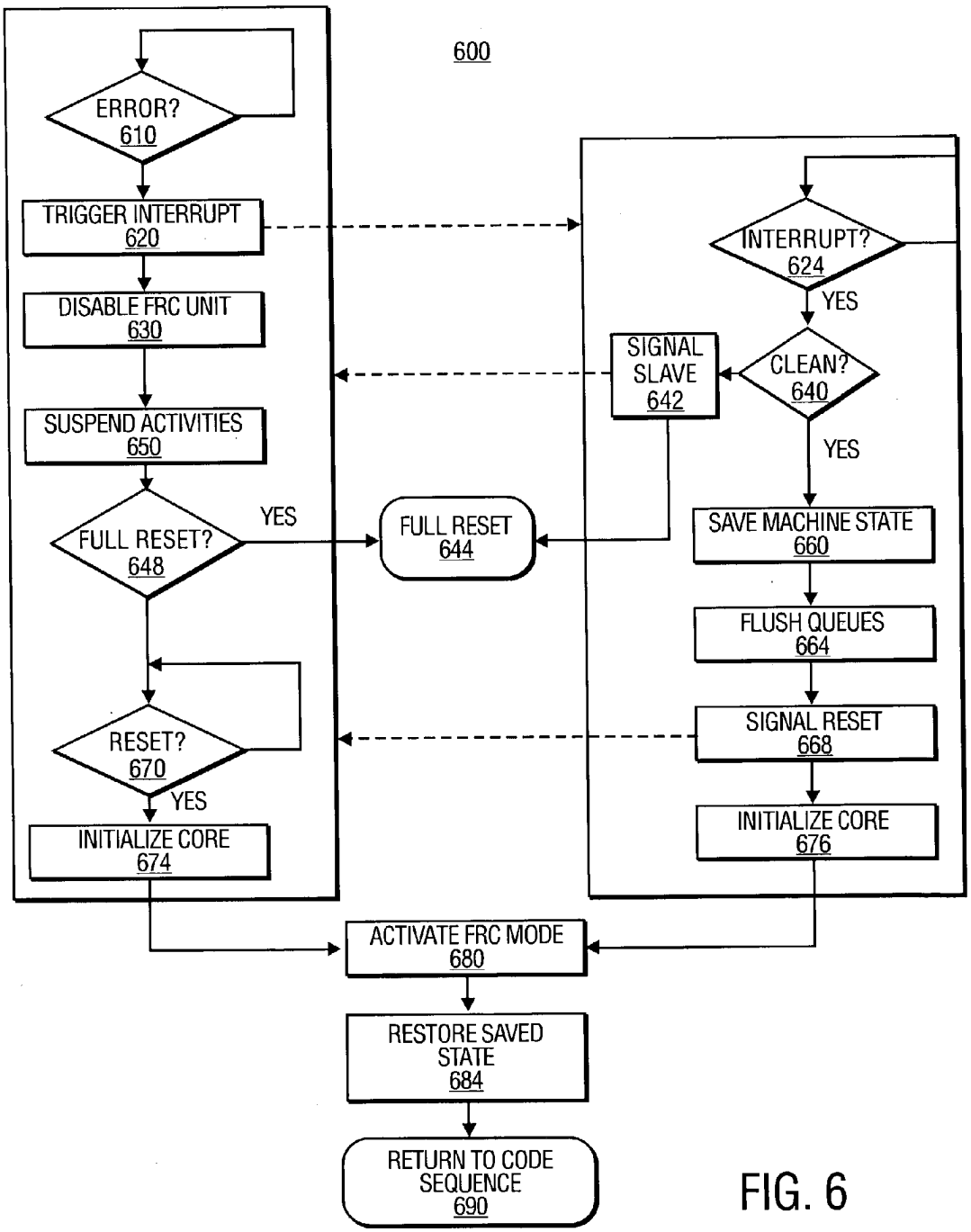

600
ERROR? 610
TRIGGER INTERRUPT 620
DISABLE FRC UNIT 630
SUSPEND ACTIVITIES 650
FULL RESET? 648
YES
FULL RESET 644
RESET? 670
YES
INITIALIZE CORE 674
INTERRUPT? 624
YES
SIGNAL SLAVE 642
CLEAN? 640
YES
SAVE MACHINE STATE 660
FLUSH QUEUES 664
SIGNAL RESET 668
INITIALIZE CORE 676
ACTIVATE FRC MODE 680
RESTORE SAVED STATE 684
RETURN TO CODE SEQUENCE 690

FIG. 6

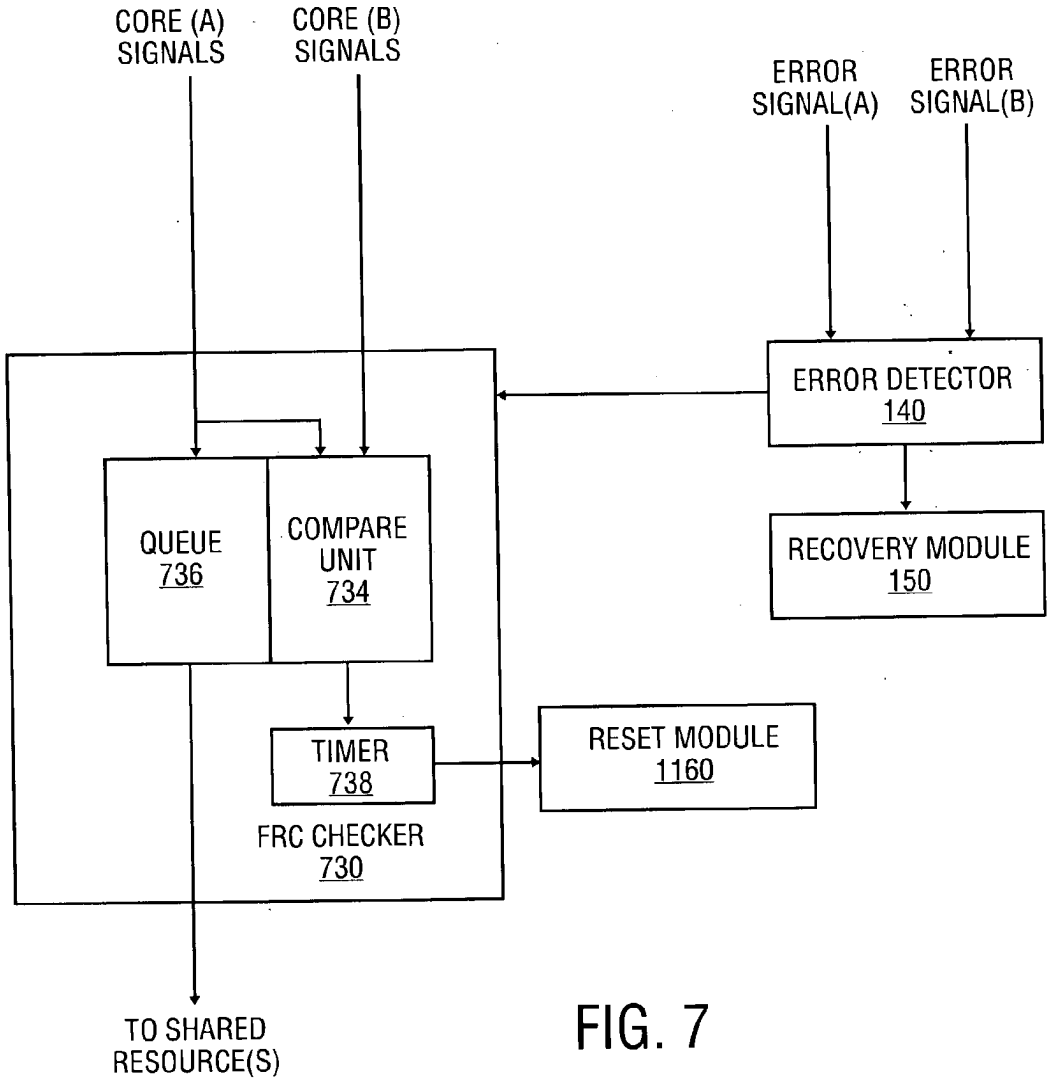

CORE (A) SIGNALS
CORE (B) SIGNALS
ERROR SIGNAL(A)
ERROR SIGNAL(B)
ERROR DETECTOR 140
QUEUE 736
COMPARE UNIT 734
RECOVERY MODULE 150
TIMER 738
RESET MODULE 1160
FRC CHECKER 730
TO SHARED RESOURCE(S)

FIG. 7

ON-DIE MECHANISM FOR HIGH-RELIABILITY PROCESSOR

BACKGROUND OF THE INVENTION

[0001] 1. Technical Field

[0002] The present invention relates to microprocessors and, in particular, to mechanisms for handling errors in FRC-enabled processors.

[0003] 2. Background Art

[0004] Servers and other high-end computing and communication systems are designed to provide high levels of reliability and availability. Soft errors pose a major challenge to both of these properties. Soft errors result from collisions between high-energy particles, e.g. alpha particles, and charge storing nodes. They are prevalent in storage arrays, such as caches, TLBs, and the like, which include large numbers of charge storing nodes. They also occur in random state elements and logic. Rates of occurrence of soft errors (soft error rates or SERs) will likely increase as device geometry decreases and device densities increase.

[0005] Highly reliable systems include safeguards to detect and manage soft errors, before they lead to silent, e.g. undetected, data corruption (SDC). However, to the extent error detection/handling mechanisms that support high-reliability operations take a system away from its normal operations, the system's availability is reduced. For example, one such mechanism resets the system to its last known valid state if an error is detected. The system is unavailable to carry out its assigned task while it is engaged in the reset operation.

[0006] One well-known mechanism for detecting soft errors is functional redundancy checking (FRC). A single processor enabled for FRC may include replicated instruction execution cores on which the same instruction code is run. Depending on the particular embodiment, each replicated execution core may include one or more caches, register files and supporting resources in addition to the basic execution units (integer, floating point, load store, etc.). FRC-hardware compares results generated by each core, and if a discrepancy is detected, the FRC system passes control to an error-handling routine. The point(s) at which results from different execution cores are compared represents the FRC-boundary for the system. Errors that are not detected at the FRC boundary can lead to SDC.

[0007] Since FRC errors indicate only that the execution cores disagree on a result, FRC errors are detectable but not recoverable. As noted above, the FRC error handling routine typically resets the system to the last known point of reliable data. This reset mechanism is relatively time consuming. It takes the system away from its normal operations, reducing system availability.

[0008] FRC is only one mechanism for handling soft errors, and for random logic and random state elements, it is the primary mechanism. Array structures present a different picture. Array structures typically include parity and/or ECC hardware, which detect soft errors by examining properties of the data. In many cases, the system can correct errors created by data corruption using relatively fast hardware or software mechanisms. However, for FRC-enabled processors, such errors are likely to be manifested as FRC errors, since they take the execution cores out of lock-step. Handling these otherwise recoverable errors through a reset mechanism reduces system availability.

[0009] The present invention addresses mechanisms for combining recoverable and non-recoverable error handling mechanisms efficiently in FRC-enabled processors.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] The present invention may be understood with reference to the following drawings, in which like elements are indicated by like numbers. These drawings are provided to illustrate selected embodiments of the present invention and are not intended to limit the scope of the invention.

[0011] **FIG. 1** is a block diagram of a processor including dual execution cores and FRC-detection and handling logic.

[0012] **FIG. 2** is a block diagram of an embodiment of the processor of **FIG. 1** that is capable of operating in multiple modes.

[0013] **FIG. 3A** is a block diagram of one embodiment of a computing system that implements the multi mode processor of **FIG. 2**.

[0014] **FIG. 3B** is a block diagram of a mechanism for signaling recoverable errors in the computing system of **FIG. 3A**.

[0015] **FIG. 4** is a block diagram representing the data paths of the computing system of **FIG. 3A**.

[0016] **FIG. 5** is a flowchart representing one embodiment of a mechanism for recovering from soft error in an execution core.

[0017] **FIG. 6** is a flowchart representing one embodiment of a mechanism for recovering from a soft error in a multi execution core processor.

[0018] **FIG. 7** is a block diagram representing one embodiment of an FRC checker that mitigates race conditions between recoverable and non-recoverable error mechanisms.

DETAILED DESCRIPTION OF THE INVENTION

[0019] The following discussion sets forth numerous specific details to provide a thorough understanding of the invention. However, those of ordinary skill in the art, having the benefit of this disclosure, will appreciate that the invention may be practiced without these specific details. In addition, various well-known methods, procedures, components, and circuits have not been described in detail in order to focus attention on the features of the present invention. For example, aspects of the present invention are illustrated using a dual-core processor, but persons skilled in the art will recognize that more than two cores may be used with appropriate modifications of the reset and recovery mechanisms.

[0020] **FIG. 1** is a block diagram representing one embodiment of an FRC-enabled processor **110** in accordance with the present invention. Processor **110** includes first and second execution cores **120**(*a*), **120**(*b*) (generically, execution core **120**), an FRC checker **130**, an error detector

140, a recovery module **150**, a reset module **160**, and shared resources **170**. A portion of the FRC-boundary is indicated by dashed line **104**. For purposes of illustration, recovery module **150** and reset module **160** are shown as part of processor **110**. These modules may be implemented in whole or in part as hardware, firmware or software and located on or off of the processor die. Similarly, shared resources **170** may include components that are on the processor die as well as components that are on one or more different die.

[**0021**] Each execution core **120** includes a data pipeline **124** and an error pipeline **128** that feed into FRC checker **130** and error detector **140**, respectively. Data pipeline **124** represents the logic that operates on various types of data as it moves through processor **110** toward FRC checker **130**. Data processed by data pipeline **124** may include result operands, status flags, addresses, instructions and the like that are generated and staged through processor **110** during code execution. Error pipeline **128** represents the logic that operates on various types of data to detect errors in the data and provide appropriate signals to error detector **140**. For example, the signals may be one or more bits (flags) representing the parity or ECC status of data retrieved from various storage arrays (not shown) of processor **110**. Soft errors in these arrays may appear as parity or ECC error flags when the corrupted data is accessed.

[**0022**] If an error reaches error detector **140** from either core **120**, recovery module **150** is activated to implement a recovery routine. Recovery can be implemented with relatively low latency by hardware, software, firmware or some combination of these. For example, there is an extremely small probability that data is corrupted in both execution cores **120** at the same (or nearly the same) time. This leaves an uncorrupted copy of the data available to restore data integrity to processor **110**. However, an FRC error will be triggered if the corrupted data from one execution core and an uncorrupted version of the data from the other execution are allowed to reach FRC checker **130** before recovery module **150** is activated. Since FRC errors are not recoverable, reset module **160** resets the system, if FRC checker **130** signals an FRC error before the underlying parity/ECC error is detected.

[**0023**] Not all FRC-errors are traceable to underlying parity/ECC or other correctible soft errors. For those that are, it is faster for error detector **140** to address the underlying soft error than for FRC-checker to address the FRC error that results when the corrupted data reaches FRC boundary **104**. As noted above, the reset process has a significantly longer latency than the recovery process, and it is to be avoided if the error can be corrected by recovery module **150**. In addition, reset usually brings the entire system down, whereas recovery only results in a temporary performance loss. For this reason, FRC checker **130** is temporarily disabled if error detector **140** detects an error in either error pipeline **128**, since execution cores **120** are no longer in lock step.

[**0024**] Execution cores **120** operate in lock step during normal FRC mode, but data pipeline **124** and error pipeline **128** may operate relatively independently. For example, ECC hardware is relatively complex and, consequently, relatively slow, especially for 2-bit errors. A flag signaling such an error may reach error detector **140** before, after, at the same time as the data with which it is associated reaches FRC checker **130**. This flexibility is generally beneficial. For example, it allows data to be used speculatively before its error status is determined. Since soft errors are relatively rare, and error pipeline **128** is generally as fast as data pipeline **124**, this flexibility is net positive. As long as the error flag arrives at error detector **140** in time to disable FRC checker **130** before it acts on a mismatch attributable to the corrupted data, the relatively low latency recovery routine is engaged.

[**0025**] As discussed below, processor **110** may implement strategies to mitigate the race between recoverable and non-recoverable error mechanisms. For example, a streamlined signaling mechanism may be used in FRC mode to speed disabling of FRC checker **130** in the event of a non-FRC error. In addition, FRC errors may be delayed for an interval prior to reset, in case a late arriving recoverable error signal obviates the need for reset.

[**0026**] For one embodiment of the invention, processor **110** is capable of operating in a high reliability (e.g. FRC) mode or a high performance (e.g. multi-core) mode. The operating mode may be selected, for example, when a computing system that includes processor **110** is booted or reset. In FRC mode, execution cores **120**(*a*) and **120**(*b*) may appear to the operating system as a single logical processor. Execution cores **120**(*a*) and **120**(*b*) process the same code sequence, and the results generated are compared by FRC checker **130**. If the results agree, a machine state corresponding to the code sequence is updated.

[**0027**] In FRC mode, one of execution cores **120** may be designated as the master. The master refers to the execution core that is responsible for updating the resources shared by execution cores **120**. The other execution core **120** may be designated as the slave. The slave is responsible for generating results from the same code sequence to be checked against those of the master. Since an error may occur in either the master or slave, embodiments of the present invention allow the master/slave designation to be changed dynamically. As discussed below, this allows the slave to take over the master designation to implement recovery if a recoverable error is detected in the execution core that is currently designated as master.

[**0028**] In multi-core mode, execution cores **120**(*a*) and **120**(*b*) may appear to the operating system as two different logical processors on a single processor die. In this mode, execution cores **120**(*a*) and **120**(*b*) process different code sequences, and each updates a machine state associated with the code sequence it processes. Portions of the machine state of a logical processor may be stored in a cache and/or register associated with the corresponding execution core. At some point(s) on the processor die, results from execution cores **120**(*a*) and **120**(*b*) are routed to shared resource(s) **170** for, e.g., storage (cache) or transmission off of the processor die (bus). For this embodiment, additional logic is provided to mediate access to shared resources **170** by execution cores **120**(*a*) and **120**(*b*). In general, multi-core mode allows the execution cores of the processor to be controlled separately.

[**0029**] **FIG. 2** is a block diagram representing an embodiment of processor **110** that is capable of operating in multiple modes, e.g. FRC mode and multi-core mode. For the disclosed embodiment, an arbitration unit **180** is provided to manage transactions to shared resources **170** by execution cores **120**(*a*) and **120**(*b*), when processor **110**

operates in multi-core mode. Arbitration unit **180** is associated with FRC unit **130**, placing the arbitration point for multi-core mode operation proximate to the FRC boundary for FRC mode operation. In multi-core mode, signals from execution cores **120**, e.g. transaction request signals, may be processed by arbitration unit **180**, which manages access to shared resource(s) **170**. In FRC mode, signals from execution cores **120** may be processed by FRC checker **130**, which compares them to detect soft errors in either execution core. Locating FRC checker **130** and arbitration unit **180** in close proximity expands the FRC boundary to encompass most, if not all, of the logic for which signals from the two execution cores remain distinct. It also reduces the wiring necessary to support processor **110** in FRC and multi-core modes.

[**0030**] Expansion of the FRC boundary in this manner naturally increases the time necessary to propagate signals to FRC checker **130**. This increased "flight time" provides more time for a parity or ECC error to reach detector **140**, which increases the opportunity for error recovery. As noted above, the recovery routine triggered by error detector **140** provides greater system availability than the reset routine triggered by FRC checker **130**. Expanding the FRC boundary thus increases both the amount of logic that is duplicated for execution cores **120** and the flight time, during which detectable errors may be identified. The former increases FRC protection, albeit through a reset mechanism. The latter increases the likelihood that errors identifiable through parity, ECC or similar core-specific protections are handled through recovery rather than reset.

[**0031**] **FIG. 3A** is a block diagram representing one embodiment of a computing system **300** in accordance with the present invention. The disclosed embodiment of system **300** includes a processor **310**, chipset **370**, main memory **380**, non-volatile memory **390** and peripheral device(s) **398**. For the disclosed embodiment of system **300**, processor **310** may be operated in FRC mode or in multi-core mode. The mode may be selected, e.g., when computing system **300** is booted or reset. Chipset **370** manages communications among processor **310**, main memory **380**, non-volatile memory **390** and peripheral devices **398**.

[**0032**] Processor **310** includes first and second execution cores **320**(*a*) and **320**(*b*), respectively (generically, execution cores **320**). Each execution core includes execution resources **324** and a bus cluster **328**. Execution resources **324** may include, for example, one or more integer, floating point, load/store, and branch execution units, as well as register files and cache(s) to supply them with data (e.g. instructions, operands, addresses). Bus cluster **328** represents logic for managing transactions to a cache **340** that is shared by execution cores **320**(*a*) and **320**(*b*), as well as to a front side bus **360**, for those transactions that may miss in shared cache **340**. Resources corresponding to the error pipeline of **FIGS. 1 and 2** may be associated with execution resources **324** and/or bus cluster **328**.

[**0033**] Interface units (IFU) **330**(*a*), **330**(*b*) (generically, IFU **330**) represent a boundary between execution cores **320** and shared resources, cache **340** and FSB **360**. The disclosed embodiment of IFU **330** includes an FRC unit **332** and an arbitration unit **334**. As noted above, FRC unit **332** and arbitration unit **334** receive signals from execution cores **320**, and locating them proximate to each other results in significant savings of wiring on the processor die. Also shown in **FIG. 3A** are error units **336**(*a*) and **336**(*b*), which include components to monitor for detectable errors in execution cores **320**(*a*) and **320**(*b*).

[**0034**] For FRC mode, FRC unit **332** compares signals from execution cores **320** for transactions to shared resources like cache **340** and FSB **360**. FRC unit **332** thus forms part of the FRC boundary of processor **310**. For multi-core mode, arbitration unit **334** monitors signals from execution cores **320** and grants access to its associated shared resource according to an arbitration algorithm. The arbitration algorithm implemented by arbitration unit **334** may be, for example, a round robin scheme, a priority-based scheme or similar arbitration algorithms. For both FRC and multi-core mode, error unit **336** may monitor signals from execution cores **320** for recoverable errors.

[**0035**] Portions of recovery module **150** and reset module **160** (**FIG. 2**) may be located on processor **310** or elsewhere in system **300**. For one embodiment, a recovery routine **392** and a reset routine **394** may be stored in non-volatile memory **390** and images of these routines may be loaded in main memory **380** for execution. For this embodiment, recovery module **150** and reset module **160** may include pointers to recovery routine **392** and reset-routine **394**, respectively (or their images in main memory **380**).

[**0036**] The disclosed embodiment of system **300** also includes interrupt controllers **370**(*a*) and **370**(*b*) (generically, interrupt controller(s) **370**) to process interrupts for execution cores **320**(*a*) and **320**(*b*), respectively. Each interrupt controller **370** is shown having first and second components **374** and **378**, respectively, to accommodate the different clock domains in which interrupt controller **370** may operate. For example, FSB **360** typically operates at a different frequency than processor **310**. Consequently, components of processor **310** that interact directly with FSB **360** typically operate in its clock domain, which is indicated as area **364** on processor **310**.

[**0037**] The disclosed embodiment of interrupt controller **370** also includes an FRC-boundary-like component in the form of XOR **372**. XOR **372** signals an FRC error if it detects a mismatch between outgoing signals, e.g. interrupt responses, of components **374**(*a*) and **374**(*b*) from execution cores **320**(*a*) and **320**(*b*). Errors attributable to interrupt controllers **370** may still arise, however, from soft errors in components **378**(*a*), **378**(*b*) in FSB clock domain **364**. These error may be detected by the discrepancies they introduce between the subsequent operations of execution cores **320**(*a*) and **320**(*b*).

[**0038**] For the disclosed embodiment of system **300**, a common snoop block **362** processes snoop transactions to and from execution cores **320**(*a*) and **320**(*b*). XOR **366** provides FRC-checking on snoop responses from execution cores **320**(*a*), **320**(*b*) and signals an error if a mismatch is detected. XORs **372** and **366** may be disabled if processor **310** is operating in multi-core mode.

[**0039**] **FIG. 3B** is a block diagram representing one embodiment of an apparatus **344** for broadcasting-recoverable error conditions to components of computing system **300**. For example, error units **336**(*a*) and **336**(*b*) may represent ECC or parity error detection logic for various arrays (register, caches, buffers, etc) of execution cores **320**(*a*) and **320**(*b*), respectively, and/or exception logic to handle these

errors. An OR gate **338** monitors error signals from execution cores **320** and asserts a signal to disable FRC unit **332** if either error signal is asserted. The error signal may be a high level interrupt, such as the machine check abort (MCA) defined for Itanium® processors. The output of OR gate **338** is also fed back to execution cores **320** to indicate to the error-free execution core that a recovery mechanism is to be initiated. A second OR gate **339** is provided to transfer error signals to execution cores **320** from the shared resources.

[**0040**] If the error signal does not disable FRC unit **332** the corrupted data triggers an FRC error, and an otherwise recoverable error is treated as a non-recoverable, e.g. FRC, error. That is, the system goes through a reset operation rather than a shorter recovery operation. Depending on the particular implementation of the system, there may be a number of cases in which the race to FRC unit **332** between the error signal and the mismatched data signals from the execution cores (created by the recoverable error) is close. For this reason, apparatus **344** may include a mechanism to accelerate error signal propagation, at least in FRC mode.

[**0041**] For one embodiment, apparatus **334** supports a high level interrupt such as an MCA that operates in both FRC and high performance modes. In high performance mode, the error signal is subject to pipeline stalls, e.g., in the front end of the execution core or in the L2 cache. This ensures that no unnecessary MCAs are taken, since the event that triggered the stall may make the error signal moot. In FRC mode, the error signal bypasses these stalls. Bypassing stalls in FRC may result in the processing of some unnecessary error signals, but it also reduces the probability that an FRC error is triggered before the (non-FRC) error signal disables FRC unit **332**. As discussed in conjunction with **FIG. 7**, embodiments of processor **110** may also include a hardware mechanism to mitigate the race between the error signal and the core signals that reflect the corrupted data.

[**0042**] **FIG. 4** is a block diagram representing data paths for one embodiment of computing system **310**, including FRC components to support processor **310** in FRC mode. For the disclosed embodiment, cache **340**, FSB **360** and execution cores **320** are coupled through a series of buffers. For example, a write-out buffer (WOB) **410** stages data evicted from cache **340** to main memory **380**, and a snoop data buffer (SDB) **420** provides snoop data from execution cores **320** or cache **340** to FSB **360**, responsive to a snoop hit in these structures (Execution cores **320** may each have one or more levels of cache in addition to shared cache **340**).

[**0043**] A pair of write-line buffers (WLB) **430**(*a*), **430**(*b*) stage data from execution cores **320**(*a*), **320**(*b*), respectively, to cache **340** or FSB **360**, and a pair of read-line buffers **440**(*a*), **440**(*b*) stage data from FSB **360** to cache **340** or execution cores **320**. Coalescing buffers (CB) **450**(*a*), **450**(*b*) collect data to be written to memory **380** and forward it to FSB **360**, periodically. For example, multiple data writes to the same line of memory may be collected in CBs **450** before triggering a write transaction on FSB **360**.

[**0044**] For the disclosed embodiment, logic associated with these buffers provides the FRC-checking and data routing functions when processor **310** is operated in FRC mode. For example, logic block **454** represents MUX and XOR functions for data in CBs **450**(*a*), **450**(*b*). The XOR function provides FRC checking if processor **310** is operating in FRC mode. The MUX function provides data routing if processor is operating in multi-core mode. Logic blocks **434** and **444** provide similar functions for data in WLBs **430**(*a*), **430**(*b*) and RLBs **440**(*a*), **440**(*b*), respectively. MUXs **460**, **470** and **480** route data from different sources to cache **340**, FSB **360** and execution cores **320**.

[**0045**] As noted above, the recovery mechanism for errors detected within the FRC boundary may be handled by various combinations of hardware, software and firmware modules. One embodiment of the recovery mechanism employs code that is closely associated with the processor. For example, the Itanium® Processor Family of Intel® Corporation, employs a layer of firmware called the processor abstraction layer (PAL), which provides an abstraction of the processor to the rest of the computing system. Implementing recovery in the PAL hides the recovery process from system level code, such as the system abstraction layer (SAL), e.g. BIOS, and the operating system. PAL-based implementations of the recovery mechanism should be able to complete quickly enough to avoid triggering a time-out period enforced by the operating system. Recovery mechanisms may also be implemented using system level code, e.g. SAL/BIOS, or operating system code. The latter implementations may not be subject to the same time constraints as the PAL-based implementation. Unless otherwise noted, the recovery mechanisms discussed below may be implemented using code associated with any of the foregoing sources.

[**0046**] **FIG. 5** is a flowchart representing a mechanism for recovering from an error detected in one of the execution cores before it triggers an FRC reset. Responsive to a parity, ECC or other error detected in one of the execution cores, a signal is broadcast **510** to indicate the start of a recovery routine. As long as the error is detected before it triggers an FRC reset, the corrupted data can be localized to one of the execution cores, leaving the machine state data of the other execution core available for recovery. Accordingly, the machine state of the good core is saved **520**. To prepare the processor for recovery, both cores are initialized **530** to a specified condition, and the saved machine state is restored **540** to the initialized cores. FRC mode is then restored **550** and the processor returns **560** to the interrupted code.

[**0047**] For one embodiment of the invention, one of execution cores **120** may be designated as the master core and the other as the slave core when processor **10** is operating in FRC mode. For this embodiment, signals generated by the master and slave cores are compared at the FRC boundary to determine if reset is necessary. If no FRC reset is warranted, signals generated by the master core are transferred to shared resource(s) **170**, and signals generated by the slave core are dropped. For this embodiment, a bit in a status register of each execution core **120** may be used to indicate its status as master or slave. The bit may be set, for example, when the system is booted or reset. As discussed below in greater detail, the master/slave status of an execution may also be changed dynamically to allow recovery for an error in either core. For errors detected within the FRC boundary, e.g. recoverable errors, the actions of the master and slave core may differ, depending on which core generated the error.

[**0048**] **FIG. 6** is a flowchart representing one embodiment of a mechanism **600** for recovering from an error detected in an execution core designated as the slave execution core.

Operations of the slave execution core are shown on the left, and operations of the master execution core are shown on the right.

[0049] Routine **600** is initiated if the slave execution core detects **610** an error (parity, ECC, etc.). The slave core triggers **620** an interrupt to signal the error condition to other components of the computing system. For embodiments of routine **600** implemented by PAL or comparable processor-level code, broadcast of the interrupt signal may be limited to components within the processor chip, such as the master execution core. In addition to signaling the error, the slave execution core disables **630** the FRC unit and suspends its activities. Disabling the FRC unit prevents the error from triggering an FRC reset when it reaches the FRC boundary, and suspending activities in the slave core prevents it from disrupting the recovery process.

[0050] In response **624** to the interrupt, the master execution core determines **640** whether its state data contains any errors. For example, each execution core may include a status bit that is set if an error is detected. The master execution core may check this bit to determine if it has also generated an error. Except for the very rare case in which soft errors occur in both execution cores almost simultaneously, the master core is likely to be clean. If it is not clean **640**, there is no uncorrupted processor state with which to implement a recovery. In this case, the master core signals **642** a reset condition to the slave core and the computing system executes **644** a full, e.g. FRC-level, reset.

[0051] If the state data for the master core is not corrupted, the master core saves **660** its machine state and flushes **664** queues and buffers in its pipeline. For example, the master core may save the contents of its data and control registers and low-level cache(s) to a protected area of memory. The master core also signals **668** a limited reset to the slave core and sets **676** it resources to a specified state, e.g. initializes its pipeline. The slave core detects **670** the limited reset and initializes **674** its pipeline, synchronizing the states of the cores.

[0052] With the cores thus synchronized, FRC mode is reactivated **680**. This may be accomplished, for example, by having each core execute a handler routine, which sets appropriate state bits in its status/control registers. The saved state is restored **684** to both execution cores, and control is returned **690** to the interrupted code sequence.

[0053] Method **600** represents an embodiment of the recovery mechanism for the case in which the error is detected in the execution core that is currently designated as the slave core. For one embodiment, the slave core is the execution core that does not "control" the shared resource(s). For example, in FRC mode, signals from the slave execution core are dropped following comparison with those from the master execution core at the FRC boundary. If no FRC error is detected, signals from the master core are used to control the shared resources outside the FRC boundary.

[0054] If the error originates in the master core rather than the slave core, recovery may be handled by changing the master/slave designations of the execution cores. For example, master/slave designation may be indicated by the state of a bit in a status register associated with each execution core. The execution core for which this status bit is in the master state controls the shared resources, which are used to implement the state-saving operations of recovery routine **600**, e.g. operation **660**.

[0055] For one embodiment of the recovery routine, the execution core in which the error originates may check its master/slave status bit. If the status bit indicates it is the slave, method **600** may be implemented as described. If the status bit indicates it is the master, it may signal the slave to change its status to master, change its own status to slave, and suspend activities.

[0056] **FIG. 7** is a block diagram illustrating an embodiment of an FRC-checker **730** that mitigates race conditions between recoverable and non-recoverable error handling. The disclosed embodiment of FRC checker **730** includes a compare unit **734**, queue **736**, and timer unit **738**. Queue **736** receives data from execution core (a), and compare unit **734** compares the data from cores A and B, and sets a status flag to indicate if the comparison yields a match. If the data matches, the status flag is set to indicate the match.

[0057] If the data does not match, the status flag is set to indicate the mismatch and timer unit **738** is triggered to begin a countdown interval. If error detector **140** receives an error flag before the timeout interval expires, it disables FRC checker **730** and triggers recovery unit **150** to implement the recovery routine.

[0058] There has thus been disclosed a mechanism for handling recoverable and non-recoverable errors in multi-core processor. The multiple cores may be operated in FRC mode, in which case, one or more checker units compare signals from the cores to detect non-recoverable errors. In addition, each core includes an error unit to detect recoverable errors. If a recoverable error is detected, the checker units are disabled and a recovery routine is implemented. A multi-core mode embodiment of the multi-core processor may include an arbitration unit, proximate to the checker, to control access to the shared resource(s). Proximity of the FRC boundary to the shared resources increases the logic protected by the FRC boundary and reduces wiring necessary for the multi-core mode implementation.

[0059] Embodiments of the present invention detect virtually all errors that are undetected in non-FRC-enabled systems, and they support recovery of virtually all detectable errors, including those that are typically handled through reset in other FRC-enabled processors.

[0060] The disclosed embodiments have been provided to illustrate various features of the present invention. Persons skilled in the art of processor design, having the benefit of this disclosure, will recognize variations and modifications of the disclosed embodiments, which none the less fall within the spirit and scope of the appended claims.

We claim:

1. A processor comprising:

first and second execution cores to operate in an FRC mode;

a resource to process a transaction from at least one of the first and second execution cores; and

an interface control unit to regulate access to the resource by the first and second execution cores, the interface control unit including an FRC check unit to compare

transaction signals from the first and second execution cores and to signal an error if the comparison indicates a mismatch.

2. The processor of claim 1, further comprising an error detector to detect errors in the first and second execution cores and to disable the FRC checker responsive to detecting an error.

3. The processor of claim 2, wherein the error detector comprises first and second error detectors to detect errors in the first and second execution cores, respectively.

4. The processor of claim 3, wherein the first error detector triggers an error signal, responsive to an error in the first execution core, to disable the FRC check unit and to initiate a recovery procedure using the second execution core.

5. The processor of claim 4, wherein the second execution core is designated as an FRC slave and is re-designated as an FRC master, responsive to the error signal.

6. The processor of claim 5, wherein the second execution core saves its machine state data to a memory location, and executes a reset sequence.

7. The processor of claim 2, wherein the first and second execution cores may also operate in a multi-core mode and the interface control unit further comprises an arbitration unit to regulate access to the shared resource by the execution cores if they operate in multi-core mode.

8. The processor of claim 7, wherein the shared resource comprises a cache that may process transactions from both the first and second cores in multi-core mode and that may process transactions from only one of the first and second core in FRC mode.

9. The processor of claim 7, wherein, responsive to detecting an error the error detector triggers an interrupt if the processor is in multi-core mode and an accelerated interrupt if the processor is in FRC mode.

10. The processor of claim 9, wherein the accelerated interrupt bypasses portions of an execution core traversed by the interrupt in multi-core mode.

11. A system comprising:

a first memory location to store a recovery routine;

a second memory location to store a reset routine;

first and second execution cores capable of operating in an FRC mode;

an error unit to initiate the recovery routine responsive to detecting an error in one of the first and second execution cores; and

an FRC checker to initiate the reset routine responsive to detecting a mismatch between signals from the first and second execution cores.

12. The system of claim 11, wherein the error unit disables the FRC checker, responsive to detecting the error in one of the first and second execution cores.

13. The system of claim 12, wherein the reset routine includes instructions executable by the first and second execution cores to initialize the first and second execution cores in either a multi-core mode or the FRC mode.

14. The system of claim 13, further comprising a cache to be shared by the first and second execution cores if the first and second execution cores are initialized in multi-core mode.

15. The system of claim 14, further comprising an arbitration unit to manage access to the cache by the first and second execution cores in multi-core mode.

16. The system of claim 15, wherein the FRC checker monitors transaction signals to the arbitration unit from the first and second execution cores in FRC mode and initiates the reset routine responsive to a mismatch in the transaction signals.

17. The system of claim 11, wherein the first and second execution cores operate as master and slave, respectively, in an FRC mode.

18. The system of claim 17, wherein the first execution core is disabled and the second execution core operates as the master, responsive to an error in the first execution core.

19. The system of claim 11, wherein the first and second execution cores may be initialized in a multi-core mode or an FRC mode.

20. The system of claim 19, wherein the error unit triggers an interrupt to the first and second execution cores, responsive to an error in one of the execution cores.

21. The system of claim 20, wherein the interrupt is an accelerated interrupt if the execution cores are in FRC mode.

22. The system of claim 21, wherein the accelerated interrupt bypasses a portion of the execution core.

23. A method comprising:

operating first and second execution cores in an FRC mode;

monitoring data of the first and second execution cores for errors;

comparing signals generated by the first and second execution cores;

executing a recovery routine, responsive to an error in the first or second execution core; and

executing a reset routine, responsive to a mismatch between signals generated by the first and second execution cores.

24. The method of claim 23, further comprising suspending signal comparison responsive to the error in the first or second execution core.

25. The method of claim 24, wherein executing a reset routine responsive to the mismatch further comprises:

triggering a delay interval responsive to the mismatch;

executing the reset routine if no error is monitored in the execution cores before the delay interval expires.

26. The method of claim 25, further comprising executing the recovery routine if an error is detected in one of the execution cores before the delay interval expires.

27. The method of claim 23, wherein operating the first and second execution cores in an FRC mode comprises operating the first and second cores in the FRC mode or in a multi-core mode, responsive to a reset signal.

28. The method of claim 27, wherein executing the recovery routine comprises:

executing the recovery routine responsive to an error signaled by an interrupt if the cores are operating in multi-core mode; and

executing the recovery routine responsive to an error signaled by an accelerated interrupt if the cores are operating in FRC mode.

29. The method of claim 23, wherein operating the first and second execution cores in an FRC mode further comprises designating the first and second execution cores as master and slave execution cores, respectively.

30. The method of claim 23, wherein executing the recovery routine further comprises disabling the first execution core and designating the second execution core as master, responsive to an error in the first execution core.

* * * * *

US005450576A

United States Patent [19]

Kennedy

[11] Patent Number: 5,450,576

[45] Date of Patent: Sep. 12, 1995

[54] **DISTRIBUTED MULTI-PROCESSOR BOOT SYSTEM FOR BOOTING EACH PROCESSOR IN SEQUENCE INCLUDING WATCHDOG TIMER FOR RESETTING EACH CPU IF IT FAILS TO BOOT**

[75] Inventor: **Barry Kennedy**, Santa Ana, Calif.

[73] Assignee: **Ast Research, Inc.**, Irvine, Calif.

[21] Appl. No.: **52,818**

[22] Filed: **Apr. 26, 1993**

Related U.S. Application Data

[63] Continuation of Ser. No. 721,692, Jun. 26, 1991, abandoned.

[51] **Int. Cl.**[6] **G06F 11/20**

[52] **U.S. Cl.** **395/650**; 371/72; 364/DIG. 1; 364/230.6; 364/267.9; 364/268.3; 395/200.01; 395/182.11; 395/183.12

[58] **Field of Search** 395/575, 800, 700; 371/9.1, 11.1, 11.3, 12, 16.3

[56] **References Cited**

U.S. PATENT DOCUMENTS

3,786,433	1/1974	Notley et al.	395/650
4,117,974	10/1978	Ciaramella	371/51.1
4,127,768	11/1978	Negi et al.	395/425
4,578,773	3/1986	Desai et al.	395/725
4,593,349	6/1986	Chase et al.	395/750
4,679,166	7/1987	Berger et al.	395/650
4,803,682	2/1989	Hara et al.	371/12
4,882,669	11/1989	Miura et al.	371/12
4,890,285	12/1989	Dichiara	371/16.3
4,943,911	7/1990	Kopp et al.	395/375
4,975,831	12/1990	Nilsson et al.	395/800
5,038,320	8/1991	Heath et al.	371/11.1
5,070,450	12/1991	Holman, Jr. et al.	395/750
5,109,521	4/1992	Culley	395/800
5,155,833	10/1992	Cullison et al.	395/700
5,168,555	12/1992	Byers et al.	395/725

FOREIGN PATENT DOCUMENTS

486304A2	11/1991	European Pat. Off. .
63-083856	4/1988	Japan .
63-104167	5/1988	Japan .

OTHER PUBLICATIONS

Taetow, W. "DSP mit universeller Struktur", *Elektronik*, pp. 96–102, vol. 37, No. 19., Sep. 1988 with translation.

Matthes, Wolfgang, "Multimikrorechnersysteme 9", *Radio Fernsehen Elektronik*, pp. 773–776, vol. 33, No. 12, Dec. 1984 with translation.

"Start-Up Master Processor Selection Method for Multi-Processor Systems", IBM Technical Disclosure Bulletin, vol. 33, No. 4, Sep. 1990, pp. 375–376.

Bursky, Dave, "More, Not Fewer, Chips Give PC's Flexible Options," *Electronic Design*, vol. 38, No. 10, pp. 95–98, May 24, 1990.

Thorson, Mark, "S3 Introduces Flexible PC Chip Set Family," *Microprocessor Report*, vol. 4, N. 9, pp. 1–2, 9–11, May 18, 1990.

Thorson, Mark, "ACI Bus Supports 486 Multiprocessing," *Microprocessor Report, vol. 4, No. 11, pp. 9–11, Jun. 20, 1990.*

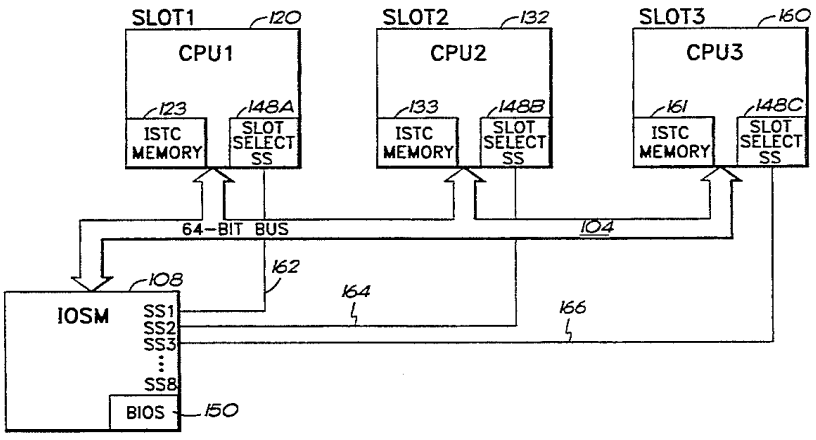
S3, Advanced Cache Controller Preliminary Information Sheet (Apr. 1990).

Primary Examiner—Parshotam S. Lall
Assistant Examiner—Richard L. Ellis
Attorney, Agent, or Firm—Knobbe, Martens, Olson & Bear

[57] **ABSTRACT**

A system for controlling initialization and self test operations in a multiprocessor system facilitates the use of central processing units based around differing microprocessor types. More specifically, the present invention involves storing configuration information, initialization self-test code, and boot code specific to each processor, memory module, or I/O circuit board in non-executable form in a non-volatile memory, on the respective circuit board, and storing the executable portion of the boot code needed by the initial boot processor in a centrally accessible non-volatile memory. Moreover, the present invention provides a system to automatically transfer default system initialization operations from a default processor to a first alternative processor if the default processor fails, and to automatically transfer initialization operations to a second alternative processor if the first alternative processor fails, and so forth depending upon how many alternative processors are installed in the multiprocessor system.

8 Claims, 4 Drawing Sheets



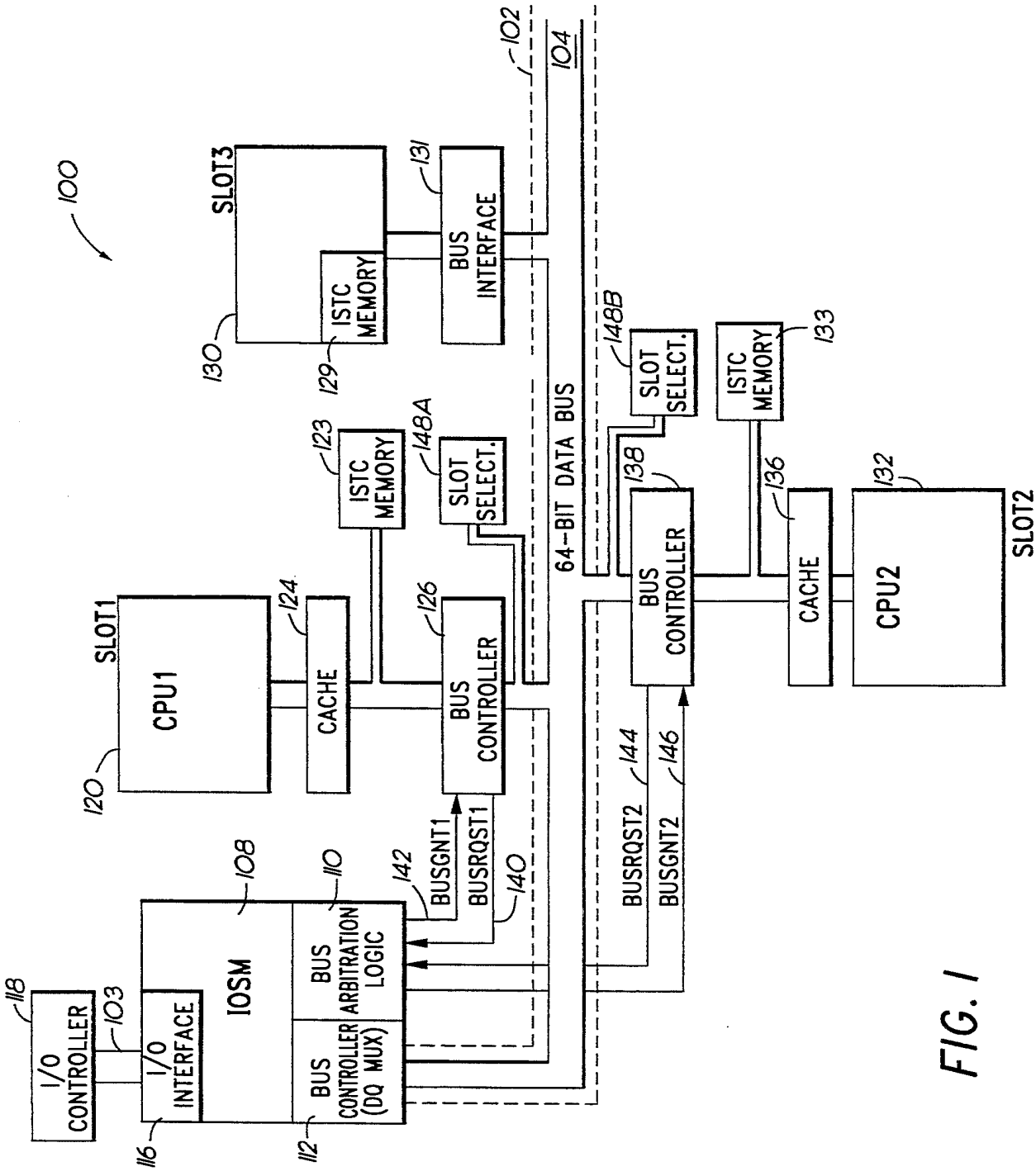

100
SLOT3
ISTC MEMORY
BUS INTERFACE
130
129
131
102
104
64-BIT DATA BUS
148B
SLOT SELECT.
ISTC MEMORY
133
123
ISTC MEMORY
148A
SLOT SELECT.
138
136
132
SLOT1
CPU1
120
124
CACHE
126
BUS CONTROLLER
BUS CONTROLLER
CACHE
CPU2
SLOT2
BUSGNT1
BUSRQST1
140
142
144
146
BUSRQST2
BUSGNT2
108
110
IOSM
BUS ARBITRATION LOGIC
BUS CONTROLLER (DQ MUX)
I/O INTERFACE
I/O CONTROLLER
103
118
116
112

FIG. 1

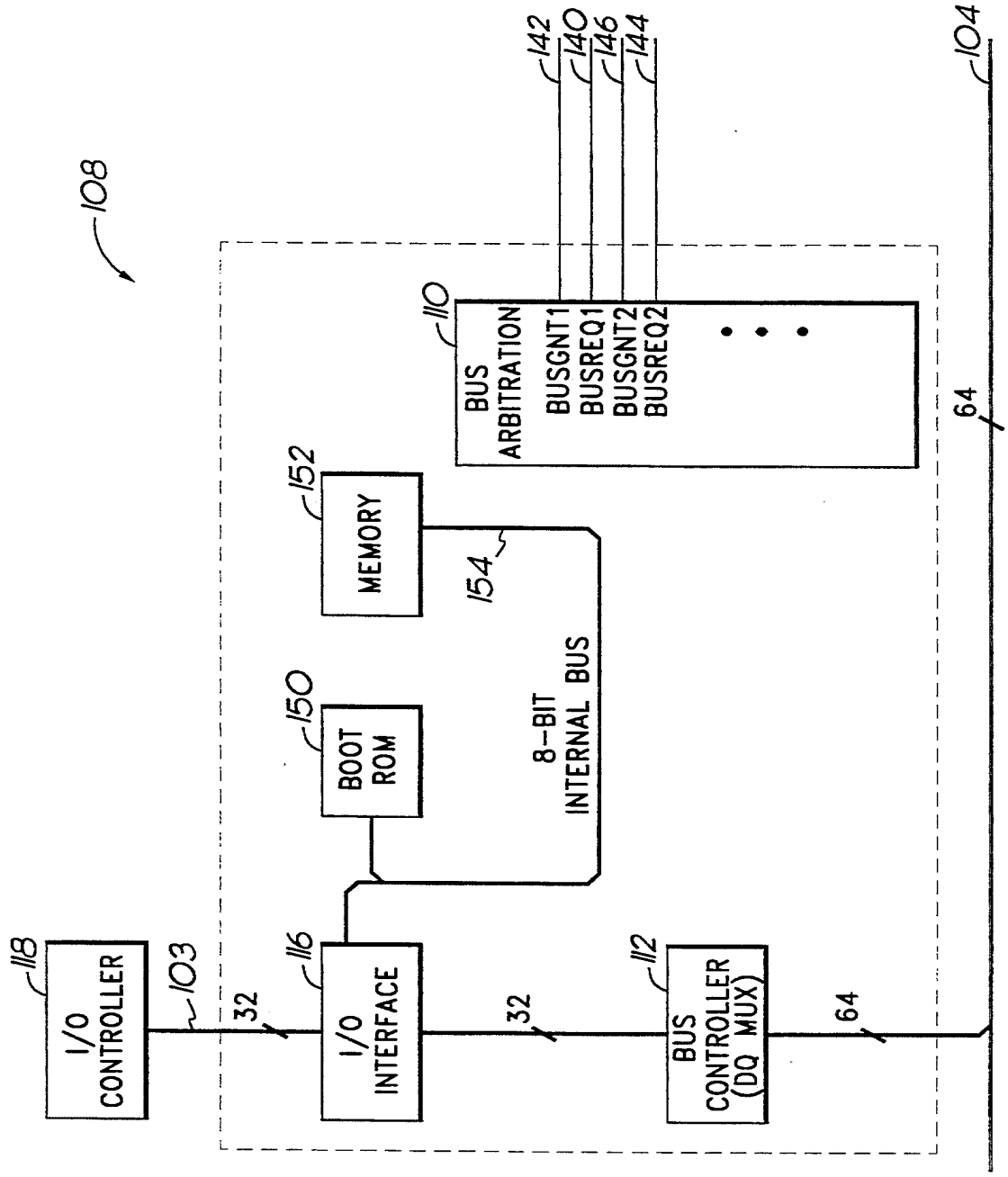

108
142
140
146
144
104
110
BUS ARBITRATION
BUSGNT1
BUSREQ1
BUSGNT2
BUSREQ2
64
152
MEMORY
154
8-BIT INTERNAL BUS
150
BOOT ROM
118
I/O CONTROLLER
103
32
116
I/O INTERFACE
32
112
BUS CONTROLLER (DQ MUX)
64

FIG. 2

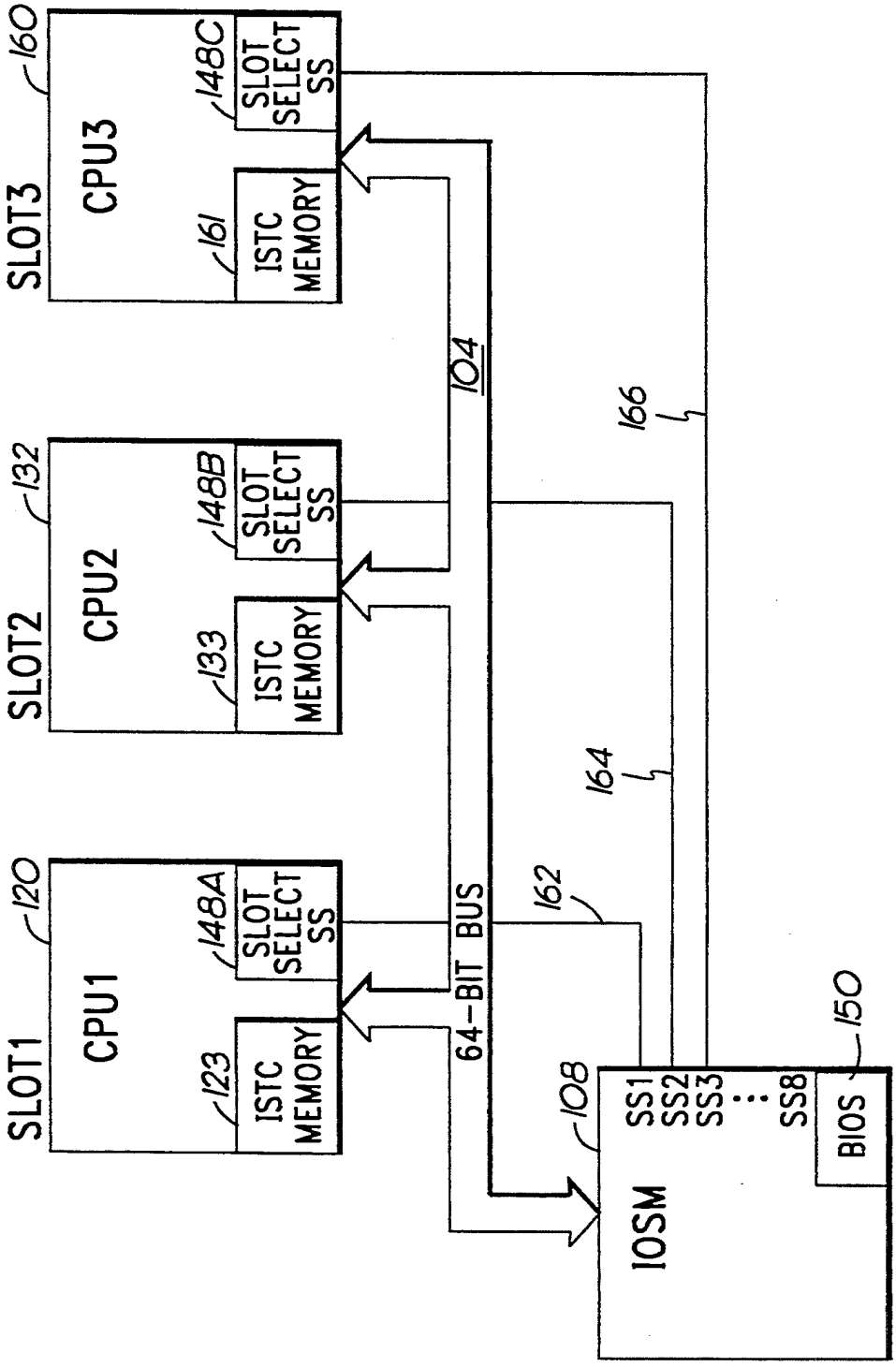

SLOT3
160
CPU3
148C
SLOT SELECT SS
161
ISTC MEMORY
SLOT2
132
CPU2
148B
SLOT SELECT SS
133
ISTC MEMORY
SLOT1
120
CPU1
148A
SLOT SELECT SS
123
ISTC MEMORY
104
166
164
162
64-BIT BUS
108
IOSM
SS1
SS2
SS3
SS8
150
BIOS

FIG. 3

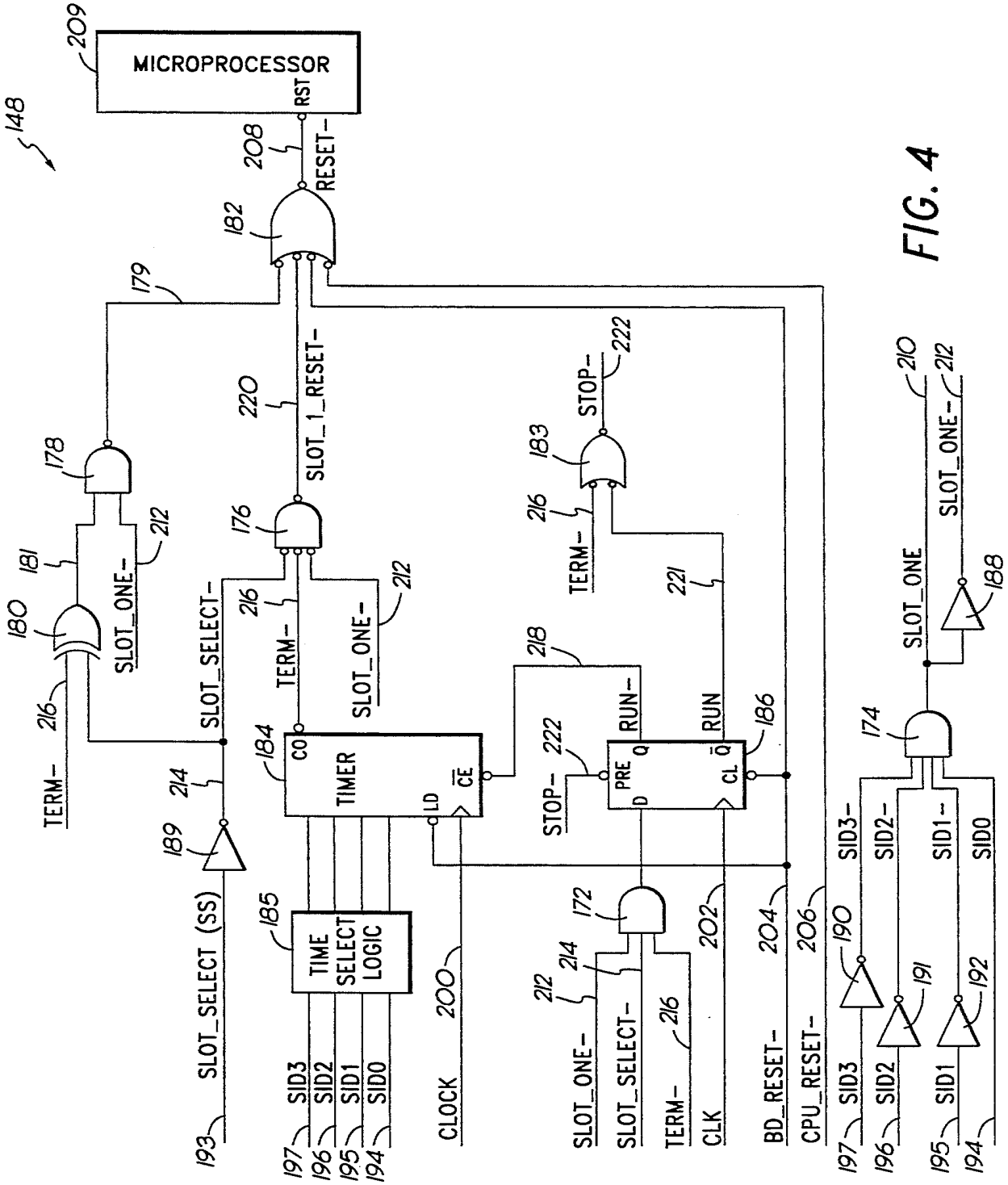

148
MICROPROCESSOR
209
RST
208
RESET-
182
179
178
180
181
SLOT_ONE-
212
216
TERM-
SLOT_SELECT-
214
189
SLOT_SELECT (SS)
193
220
SLOT_1_RESET-
176
216
TERM-
212
SLOT_ONE-
184
CO
TIMER
LD
CE
185
TIME SELECT LOGIC
SID3
SID2
SID1
SID0
197
196
195
194
CLOCK
200
183
STOP-
222
216
TERM-
218
221
RUN-
RUN
Q
PRE
D
CL
186
STOP-
222
172
212
214
SLOT_ONE-
SLOT_SELECT-
TERM-
216
CLK
202
204
BD_RESET-
206
CPU_RESET-
210
SLOT_ONE
188
174
SLOT_ONE-
212
190
191
192
SID3-
SID2-
SID1-
SID0
SID3
SID2
SID1
197
196
195
194

FIG. 4

DISTRIBUTED MULTI-PROCESSOR BOOT SYSTEM FOR BOOTING EACH PROCESSOR IN SEQUENCE INCLUDING WATCHDOG TIMER FOR RESETTING EACH CPU IF IT FAILS TO BOOT

This application is a continuation of application Ser. No. 07/721,692, filed Jun. 26, 1991 now abandoned.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to management of initialization operations in multiprocessor computer systems. Specifically, the invention relates to a method and apparatus to automatically control the determination of which processor in the system will boot the system. If the boot processor fails, control automatically shifts to an alternate processor. Additionally, the present invention involves a system wherein each central processing unit includes a corresponding copy of its own boot code. The boot code is transferred to memory for execution by the corresponding processor to carry out initialization operations.

2. Description of the Related Art

The processing requirements of multi-user computer systems for personal and business needs continuously become more demanding. For instance, more complex application programs for use with local area networks are continuously being developed. Moreover, many multi-user systems provide access to more than one operating environment, such as UNIX and DOS, on the same system.

In general, the computers servicing these needs are single processor systems conforming to conventional architectures using a standard input/output I/O bus such as the Extended Industry Standard Architecture (EISA). New and more powerful systems constantly emerge. However, upgrading an old system generally requires investing in substantial hardware modifications or buying a new system.

One solution to the constantly changing power of microprocessors controlling a system is the CUPID architecture designed by AST Research Inc. In the CUPID architecture, the microprocessor based central processing unit (CPU) is not permanently attached to the backplane bus, but is a removable circuit board running at its own speed, asynchronous with the backplane bus operations. Thus, when more power from the microprocessor is desired, a faster CPU can replace the existing CPU.

However, as processing power demands increase, application software and operating systems performance would benefit from an architecture similar to the CUPID architecture, but which has multiprocessor capabilities to provide parallel processing, and to service high numbers of simultaneous users while still retaining high batch performance.

A number of problems hamper the development of such a multiprocessor architecture, such as, determining which processor boots the system, and, if one processor fails to boot the system, enabling another alternate processor to take over boot operations. Moreover, if after one processor boots the system (by executing instructions out of a read-only-memory (ROM)), but another processor utilizes a different instruction set than the boot processor installed in the system, then this other processor will require access to a different boot ROM.

SUMMARY OF THE INVENTION

Advantageously, a system to provide multiprocessor capabilities would overcome these problems and provide expandability capabilities for adding additional memory, I/O controllers, and CPUs. Such a system would provide high performance batch processing and/or high-availability, multi-user services. The system architecture would also be suitable for a wide range of microprocessor-based designs.

The present invention provides a multiprocessor interconnection architecture with a backplane system bus for use with multiple processors. The architecture provides a method and apparatus to interface multiple processors to the backplane system bus such that upon start-up of the computer system, one default boot CPU is designated to boot the system, and if this CPU fails, control transfers automatically to an alternate processor. This provides fault tolerance and high availability for the system. Additionally, each CPU, memory, and I/O module for the system maintains a copy of the initialization data and self test code portion of its own boot code, and potentially its entire boot code, which can be transferred to memory for execution for the purpose of initialization and self-testing of the associated module.

One aspect of the invention involves a multiprocessor distributed initialize and self test system for use with a multiprocessor interconnect having a backplane bus with connector slots capable of receiving multiple central processing units, memory modules, and other input/output modules. The system has a centrally accessible memory, a default boot central processing unit installed in a first slot on the backplane bus, and may also have one or more alternate boot central processing units installed in a second slot on the backplane bus. The system also has centrally accessible boot code executable by the default central processing unit and the alternate central processing unit. A first slot select circuit associated with the default central processing unit which allows the default central processing unit to execute boot operations, is assigned a pre-determined time-out period, selected by a respective slot identification code. If the time-out period elapses before the default central processing unit successfully boots, the first slot select circuit disables the default central processing unit by placing it in a reset state.

A second slot select circuit associated with the alternate central processing unit holds the alternate processor in a reset state until either the default central processing unit successfully boots the system, or a predetermined period of time, selected by a respective slot identification code, elapses indicating that the default central processing unit failed to boot the system.

In one embodiment the default boot central processing unit comprises an INTEL 80486 compatible central processing unit, and the alternate central processing unit comprises an INTEL 80486 compatible central processing unit.

Another aspect of the present invention provides a multiprocessor distributed initialize and self test system for use with a multiprocessor interconnect, wherein the interconnect includes a backplane bus capable of connecting multiple central processing units, memory modules, and other input/output modules. The system has a first central processing unit installed in the backplane

bus, a second central processing unit installed in the backplane bus, and a centrally accessible memory. The system also has centrally accessible boot code accessible and executable by the default central processing unit. A first non-volatile memory associated with the first central processing unit contains initialization and self test code specific to the first central processing unit. In a preferred embodiment, this is transferred to the centrally accessible memory and assembled for execution by the first central processing unit. A second non-volatile memory associated with the second central processing unit contains boot code specific to the second central processing unit. Preferably, this code may be transferred to the centrally accessible memory and assembled for execution by the second central processing unit.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram of an exemplary multiprocessor interconnection system with a backplane system bus according to the present invention.

FIG. **2** is a more detailed block diagram of the input-/output services module (IOSM) shown in FIG. **1**.

FIG. **3** is a block diagram of a configuration of a multiprocessor interconnection system with three central processing units and illustrates the slot select interconnections between the IOSM and the slot select circuits of the present invention.

FIG. **4** is a schematic diagram of the slot select logic of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

FIG. **1** is a block diagram of a multiprocessor interconnection system **100**. The system **100** of the present embodiment comprises, in general, a backplane system bus **102** with a 64-bit multiple processor data bus **104**, an input/output (I/O) bus **103**, which advantageously comprises a 32-bit Extended Industry Standard Architecture (EISA) bus in the present embodiment, an input-/output services module (IOSM) **108** with a corresponding bus controller **112**, a conventional I/O controller(s) **118**, a first central processing unit (CPU1) **120** with a corresponding cache **124** and bus controller **126**, a memory module **130** with a corresponding bus interface **131**, and a second central processing unit (CPU2) **132** with a corresponding cache **136** and bus controller **138**. Preferably, the I/O controller(s) **118** comprise(s) conventional EISA or ISA compatible controller(s) in the present embodiment. Advantageously, the I/O bus **103** has 8 I/O connectors (e.g., conventional EISA I/O connectors) for the I/O controller(s) **118**, and the backplane system bus **102** for the 64-bit system has eight system connectors along the 64-bit system bus **104**. An additional connector designated to contain the IOSM **108** is located between the backplane system bus **102** and the I/O bus **103**. The IOSM **108** interfaces the 64-bit system bus **104** with the 32-bit I/O bus **103**.

Advantageously, the bus connectors for the 64-bit system are 240 pin METRAL connectors from DuPont, and the bus connector for the IOSM **108** is a 480 pin METRAL connector. The I/O bus connectors in the present embodiment are standard connectors from Burndy Corp., as well known in the art.

The IOSM

The IOSM **108**, as shown in more detail in FIG. **2**, comprises bus arbitration logic **110**, the bus controller **112** which interfaces with the 64-bit system bus **104**, an I/O interface **116**, which interfaces with the 32-bit I/O bus **103**, a central boot read-only-memory (ROM) **150**, a memory **152**, and an internal 8-bit data bus **154** which interconnects the central boot ROM **150**, the memory **152** and the I/O interface **116**. Preferably, the internal 8-bit bus **154** also connects to a real time clock (not shown), a parallel port (not shown) a serial port (not shown), a floppy disk controller (not shown), a keyboard controller (not shown), and a system timer/counter (not shown), as well understood in the art.

The I/O interface **116** advantageously comprises a conventional EISA bus controller chip set, well known in the art, and interfaces the conventional I/O controllers **118** and the internal 8-bit bus **154** with the 64-bit multiple processor system bus **104** via the bus controller **112**. The bus controller **112** interfaces with the system bus **104** using a 32-bit to 64-bit multiplexer/demultiplexer (a double word/quad word multiplexer ('DQ MUX')). The DQ-MUX of the bus controller **112** breaks up 64-bit words into two 32-bit words, or combines pairs of 32-bit words into 64-bit words, as well known in the art. Advantageously, the bus controller **112** also includes a single level storage buffer (not shown).

In the present embodiment, the central boot ROM **150** comprises a read-only-memory with the basic input-/output system (BIOS) instruction set for an INTEL 80486 or 80386 microprocessor. Accordingly, in the present embodiment, at least one CPU connected to the 64-bit bus is, or emulates, an INTEL 80486 or 80386 microprocessor in order to boot the system. Moreover, any alternate boot processors are, or emulate an INTEL 80486 or 80386. However, other processor types may be selected to boot the system with a corresponding change in the boot ROM **150**, as well known in the art. Advantageously, the memory **152** comprises 8 Kbytes of complementary metal oxide semi-conductor (CMOS), static random access memory (SRAM).

The bus arbitration logic **110** accepts a number of individual bus requests from various devices which can become bus masters and provides a signal to grant the bus to the device requesting the bus as well understood in the art. The bus arbitration logic **110** operates on a conventional scheme consisting of two signals carried by a bus request (BUSRQSTn) signal line and a bus grant (BUSGNTn) signal line, one of each for every device which can become a bus master. The bus arbitration logic **110** communicates with bus controllers for these operations. For example, the bus controller **126** for the CPU1 **120** (FIG. **1**) requests the bus by activating a BUSRQST1 signal line **140**, and the bus arbitration logic **110** responds with an active signal on the a BUSGNT1 signal line **142** to grant the bus to the bus controller **126**. Similarly, the bus controller **138** for the CPU2 **132** requests the bus by activating the BUSRQST2 signal line **144**, and the bus arbitration logic **110** grants the bus to the bus controller **138** by activating the BUSGNT2 signal line **146**. The I/O interface **116** may also obtain control of the bus, on behalf of an I/O controller **118** requesting to be a bus master, with corresponding BUSRQST0 and BUSGNT0 signal lines (not shown).

Devices installed on the system bus **102** advantageously accept a 64-bit transfer even though the actual device may not utilize a full 64-bit data bus. For instance, if the CPU1 **120** is based upon an INTEL 80486 which uses a 32-bit data bus, the bus controller **126** accepts a 64-bit transfer from the system bus **102**, places

this data into the cache **124** which provides a 32-bit interface to the CPU1 **120**.

The CPU Modules

The CPU1 module **120** could be any microprocessor chip set running at any speed. In the present embodiment, at least one CPU is based upon an INTEL 80486 or compatible microprocessor. Accordingly, throughout the remainder of this description, references to the CPU1 **120** assume an INTEL 80486-based CPU with supporting resources and on-board crystal oscillators for independent timing. Other CPUs in the system need not be 080486-based as explained in more detail herein.

CPUs installed in the bus **104** may have independent asynchronous timing with respect to the bus **104**.

The CPU1 **120** has a corresponding non-volatile memory **123** which contains configuration information and CPU specific initialize and self test code (ISTC) for the CPU1 **120**, and further comprises a slot select circuit **148**A which is explained in detail below. In one alternative embodiment of the invention, the non-volatile memory **123** comprises a programmable read-only-memory (PROM), as well known in the art. In the present embodiment, the cache **124** for the CPU1 **120** is a 256-Kbyte, two-way, set-associative, write-back cache with a 32-byte line size (4 bus cycles×8 bytes). The cache **124** interfaces the asynchronous CPU1 **120** with the synchronous 64-bit bus **104** via the bus controller **126** which responds to signals on the BUSRQST1 signal line **140** and the BUSGNT1 signal line **142** as explained. The cache **124** supports write-back and the conventional Modified, Exclusive, Shared, Invalid (MESI) protocol to maintain cache coherency for the multiprocessor system **100**. The cache **124** has a corresponding 64-bit interface (not shown) for the 64-bit bus **104** and a 32-bit interface (not shown) with the 80486 processor. When the cache **124**, or any other cache, generates a write-back cycle, it asserts an active low signal on the write-back start (WBSTRT-) control line (not shown) to indicate the beginning of a write-back cycle from the cache, as well understood in the art.

The CPU2 **132** is similar to the CPU1 **120** except that the CPU2 **132** need not be an 80486 or 80386 based CPU. The CPU2 **132** includes an non-volatile ISTC memory **133** which contains configuration information and ISTC for the microprocessor of the CPU2 **132**. The CPU2 **132** also comprises a slot select circuit **148**B, and has a corresponding bus controller **138** and a cache **136** similar to those associated with CPU1 **120**. Further CPUs may also be added to the system and need not comprise INTEL 80486 or 80386 based CPUs.

The Memory Modules

In the present embodiment, the memory module **130** accepts 64-bit transfers. However, memory modules need not be capable of accepting full 64-bit transfers. Advantageously, the memory **130** comprises 40-bit single-in-line memory modules (SIMMs) which could be constructed from 1-Meg-by-4 or 4-Meg-by-4 dynamic random access memories (DRAMs). Toshiba's THM401020SG-80 is an exemplary 10-chip SIMM appropriate for use in the present system. The memory **130** supports 64 megabytes (Mbytes) (per module) of RAM with 1-Meg-by-4 DRAM based SIMMs, or 256 megabytes (per module) with 4-Meg-by-4 DRAM based SIMMs. The present embodiment allows up to four 256 Mbyte memory modules to be installed in the system. The memory module **130** also includes error correction code (ECC) capability for reliability. However, to eliminate the read-modify-write cycle caused by 32-bit operations on a 64-bit ECC memory, ECC is performed on a 32-bit basis and not on a 64-bit basis.

The memory module **130** also comprises a DRAM controller (not shown) which provides conventional row address select (RAS), column address select (CAS), hidden refresh, address multiplexing, page mode and burst mode support. Accordingly, the DRAM controller comprises DRAM RAS/CAS address multiplexers, RAS/CAS generation logic, refresh counters, page-mode/page-hit logic and DRAM address drivers, all well understood in the art. The DRAM controller derives memory timing from an external clock signal carried on the system bus **104**, and therefore, runs synchronously with the system bus **104**.

In the present embodiment, the memory **130** also includes a corresponding bus interface **131** to the 64-bit bus **104**. Advantageously, the bus interface **131** comprises a 8-level deep by 64-bit wide register. Parity generation and checking for the bus interface **131** is performed on a byte-by-byte basis on the 64-bit bus side of the register.

Additionally, the memory has an associated non-volatile ISTC memory **129** which contains the ISTC for the memory **130**.

FIG. **3** illustrates an embodiment with three CPUs: the CPU1 **120** installed in slot **1**, the CPU2 **132** installed in slot **2**, and a CPU3 **160** installed in slot **3** with a corresponding ISTC memory **161** and a slot select circuit **148**C. FIG. **3** further illustrates the connections from the slot select circuits **148**A, **148**B, and **148**C to the IOSM **108**. As seen in FIG. **3**, the slot select circuit **148**A for the CPU1 **120** connects to the IOSM **108** with a slot select signal line (SS1) **162**. Similarly the slot select circuits **148**B and **148**C connect to the IOSM **108** with slot select signal lines (SS2 and SS3) **164** and **166** respectively. Any other slot select circuits on CPUs installed in the system have a corresponding slot select signal line from the IOSM **108**. The SS1, SS2 and SS3 signal lines **162**, **164**, and **166** respectively, connect to a respective SLOT_SELECT signal line **193** (FIG. **4**) for each slot select circuit **148**A, **148**B and **148**C, respectively. The signals on the SSn signal lines are controlled by a memory or I/O mapped slot select register (not shown) in the IOSM **108**. Thus, the CPUs in the system can access or change the signals on these lines by executing a read or write to the memory or I/O mapped address for the register. These SSn signal lines are used during boot operations as explained herein.

The Slot Select Circuits

FIG. **4** illustrates the logic **148** corresponding to the slot select circuits **148**A and **148**B and slot select circuits for any other CPU installed in the system. The slot select logic **148** comprises AND gates **172**,**174**, **176**, a NAND gate **178**, an EXOR gate **180**, OR gates **182**,**183**, a timer **184**, time select logic **185**, a latch **186**, and inverters **188**, **189**, **190**, **191**, and **192**. The timing circuit responds to signals on a SLOT_SELECT signal line **193**, SLOT_ID (SID0, SID1, SID2, SID3) signal lines **194**–**197**, a CLOCK signal line **200**, a CLK signal line **202**, a BD_RESET- signal line **204**, and a CPU_RESET- signal line **206**. The slot select circuit **148** provides a signal on a RESET- signal line **208** connected to the reset pin of the microprocessor **209** on the CPU board. The slot select circuit **148** further responds to a number of internal signal lines, such as a SLOT_ONE signal line **210**, a SLOT_ONE- signal line **212**, a SLOT_SELECT- signal line **214**, a TERM- signal line **216**, a RUN- signal line **218**, a SLOT_1_RESET- signal line

220, a RUN signal line 221, and a STOP- signal line 222. The functions of these signal lines are further explained herein.

Distributed Initialize Function

In the multiprocessor system 100 which utilizes a system bus 102, a number of problems can arise if more than one processor type is installed on the 64-bit bus 104. For instance, provision needs to be made to determine which processor will boot the system, and if this "default" boot processor fails to boot the system, an alternative processor, if one is installed, should take over boot operations to enhance the availability of the system.

Another potential difficulty with the multiprocessor system 100 is that a central system boot ROM 150 should be accessible by all CPUs for initialization operations. However, the boot code for each CPU in the system may differ. Thus, changes to the system boot code are often needed when the processor type changes, when problems are found, or when the memory, cache and I/O features are enhanced. Even if boot code for numerous processors were included in the central boot ROM 150, as new developments emerge, the boot ROM would need updating.

The present invention solves this problem by providing memories (e.g., ISTC memory 123, ISTC memory 133, and ISTC memory 129, etc.) for each CPU, I/O board, and memory module. These ISTC memories contain configuration, and initialization and self-test code (ISTC) (commonly referred to as the power-on self test (POST) and INITIALIZE portion of the boot code in the art) specific to the respective CPU, I/O board or memory module. The CPU ISTC memories hold configuration information and the ISTC specific to the associated CPU, and the I/O board ISTC memories hold the peripheral configuration information and the ISTC for the associated I/O board. For instance the memory 123 contains configuration information and ISTC for the CPU1 120 and the memory 133 contains configuration information and ISTC for CPU2 132, and the memory 129 contains configuration information and ISTC for testing and initializing the memory module 130.

Advantageously, the PROMs 123, 129, 133, and any other ISTC memories are memory mapped, and preferably addressable in upper memory space (e.g., above the 2 gigabyte boundary). The precise address location mapped to each ISTC memory on a circuit board depends upon the slot in which the board is installed. For instance, advantageously, the memory 123 on CPU1 120 is addressable beginning at address location D100,0000 hexadecimal (hex), if CPU1 120 is installed in slot 1, and the memory 133 on CPU2 132 is addressable beginning at address location D200,0000 hex, if CPU2 132 is installed in slot 2 on the bus.

However, if each circuit board ISTC memory contained boot code in executable form (e.g., 32-bit words for INTEL 80486 based CPUs) and was directly accessible for execution of the code in the ISTC memory, then each board would need data shift logic to interface the ISTC memory with the 64-bit system bus 104. This logic would add to the complexity, and therefore increase the cost, of the system. Therefore, in the present embodiment, the ISTC memories are 8 bits (1 byte) wide and are not directly accessible for execution of the code (i.e. the ISTC memories are execution inaccessible). In the present embodiment, the board specific code held in the ISTC memories on the circuit boards is accessible on a byte-by-byte, non-executable basis. Accordingly, in the present embodiment, the ISTC memories interface with the least significant byte of the system bus 104.

When a circuit board is initializing, the board specific ISTC is transferred from the associated ISTC memory to the memory 130 by the boot processor and assembled into executable code. The non-executable code is assembled by transferring the code, byte-by-byte, from the byte-wide memory into 32-bit words in the memory 130.

The assembly of the information in the ISTC memories into system memory 130 preferably involves moving the information, byte-by-byte in the present embodiment, into a register until 4 bytes are assembled in the register (or 8 bytes if the system memory is 8 bytes wide). Then the contents of the register are preferably moved to the system memory 130 in one move operation. Alternatively, the ISTC memories could be assembled directly on a byte-by-byte basis into the system memory, but this would require more time because of the increase in system memory accesses.

As well understood in the art, conventional addressing allows byte addressing of the bytes on the system bus 104. For instance, the INTEL 80386 and 80486 provide address lines starting at address line A2 (the third significant bit in the address) and provide four byte enable lines for selecting which bytes are active during a bus transaction. As explained above, in the present embodiment, the ISTC memories are one byte wide and interface to the least significant byte of the system bus 104. Therefore, the byte enable lines select the least significant byte as active during transfers from the ISTC memories to the memory 130 and any data on the remaining seven bytes of the system bus 104 are ignored.

However, in an alternative embodiment, the ISTC memories may be wider that one byte. Preferably, the width of the ISTC memory interface to the system bus 104 is contained along with the configuration information in the ISTC memory. Thus, the system can determine how to transfer the information from an ISTC memory to the memory 130.

In order to properly access the ISTC memories, the address lines of these ISTC memories connect to system address lines A3 and above (where A0 is the least significant bit). In this manner, incrementing the address on the system address lines by eight, is detected by the ISTC memories as an increment of one on address line A3, and the byte enable line are used to select which bytes are active on the system bus 104.

For instance, if the ISTC memory interface to the system bus 104 is two bytes wide, then the software which transfers the information from an ISTC memory to the system memory 130 for execution preferably transfers two bytes at a time instead of a single byte per transfer. The byte enable lines are used to select the two least significant bytes as active on the system bus 104 with each transfer.

Once the ISTC is assembled in memory, it is executed from memory by the corresponding CPU, or executed by the boot CPU if the ISTC corresponds to a memory module or I/O circuit board.

However, until a processor is controlling the system, the non-executable code from the ISTC memories is inaccessible by a CPU attempting to boot the system. Therefore, a central boot ROM is still needed that is accessible to any CPU which can boot the system. In

the present embodiment, a central system boot ROM **150** accessible by all processors contains INTEL 80×86 executable boot code in addition to conventional DOS BIOS CALL code which may be used by the operating system, as well understood in the art. Accordingly, in the present embodiment, the boot CPU is based around an INTEL 80×86 compatible processor (e.g., an INTEL 80486). Because the only board unique to the entire system is the IOSM **108**, the boot ROM **150** (FIG. 2) provides the central system boot code and contains the conventional DOS BIOS code.

Upon power-up reset, the default boot CPU (assume this is CPU1 **120** for present discussion) executes code from the system boot ROM **150** (this code is for an INTEL 80×86 which is why the boot CPU is an 80×86 based CPU in the present embodiment). As previously explained, the ISTC memory associated with each CPU contains the ISTC code as well as configuration information about the CPU. Thus, the boot CPU determines how many CPUs are present on the bus **104** by reading the memory location assigned to the ISTC memory for each slot (e.g. D100,0000 hex for slot 1, D200,0000 hex for slot 2, etc). If a slot contains a CPU, then the boot CPU will receive information about the type of CPU installed in the slot.

The CPU1 **120** then tests the first megabyte of memory in the system and sets a 'check-in-word' in the CMOS portion of the memory **152**. Once the integrity of the first megabyte of memory is established, the CPU1 **120** copies and assembles its own ISTC byte-by-byte from its ISTC memory **123** into memory in 32-bit word executable format and begins executing its own ISTC. Once the CPU1 **120** is initialized, it checks itself 'in' by writing to the check-in-word in the CMOS portion of the memory **152**.

After the CPU1 **120** is checked 'in', it copies the ISTC from the ISTC memory **129** to the memory **130**, executes the ISTC, and creates a memory configured/-memory good table in the CMOS portion of the memory **152**.

Once the memory module **130** is configured (and any other memory modules installed in the system are configured), the CPU1 **120** transfers and executes the ISTC from appropriate ISTC memories on the I/O controllers that are installed on the system bus **104**. Then, the CPU1 **120** transfers the next CPU's (assume CPU2 **132** for present discussion) ISTC from the next CPUs's associated ISTC memory (memory **133** for the CPU2 **132**) to the memory **130** and allows the CPU2 **132** to exit a reset state and execute its own ISTC. After successful completion of the initialization and self test, the CPU2 **132** checks 'in.'

Once each CPU in the system has executed its ISTC and has checked 'in,' the CPU1 **120** controls the system again and begins the operating system boot process.

If one or more CPUs in the system are not INTEL 80×86 based CPUs, then the ISTC memories for these CPUs would hold the entire boot code for these CPUs, not just the configuration information and ISTC.

A time limit is provided for each stage of the boot process so that if the time allotted for a device to initialize and check-in is exceeded, that device is assumed to be non-functional and is disabled. The remaining hardware may then attempt to complete the boot process. When multiple CPUs are installed in the system, if the default boot processor exceeds the time allotted for it to boot (it fails), an alternative boot processor can take over boot operations as explained below.

Alternative Boot Processor Functions

In a multiprocessor environment, one critical determination is which processor will boot the system. Moreover, if one processor fails, provision for disabling the non-functional processor and allowing another alternative processor to boot the system enhances the availability of the system. With multiple CPUs connected to the same bus, the problem is automatic boot control of the system. The reason for the difficulty is that until at least one CPU is operating the system, no microprocessor control is available to supervise the other processors.

Accordingly, in the present invention, each CPU on the bus includes slot select logic **148** (FIG. 4). In general, at power-up or system reset the slot select logic **148** for each CPU, is loaded with a time-out count dependant upon the slot in which the board is installed. Each slot connector has four hard connections which provide signals for a 4-bit identification code (slot ID) on signal lines **194–197** for a board installed in the slot. The signals on the SLOT_ID signal lines **194–197** select a pre-determined time-out period from the time select logic **185** corresponding to the time-out period assigned to the slot in which the slot select logic **148** is installed. The time-out period for the slot select logic **148**A is a lower value than the time-out period for the slot select logic **148**B and so forth. Advantageously, the time select logic **185** may comprise a programmable logic array or a read-only-memory with time-out values programmed corresponding to each slot. In general, the slot select logic **148** on the default boot CPU allows this CPU to boot. The remainder of the slot select logic circuits **148**, installed in other slots, hold the associated CPUs in reset until the predetermined time-out period has expired for the associated CPU, or until the boot CPU successfully boots the system and then activates the CPUs in other slots.

If the default boot CPU does not successfully boot and disable its own slot select logic **148** before it times-out, it enters a reset state. The next CPU (based upon a slot identification) then times-out and exits the reset state and attempts to boot the system as an alternate boot processor. The alternate boot processors are the same type processor as the default boot processor so that they can execute the boot code in the central boot ROM **150** (e.g., in a preferred embodiment, the alternate boot processor(s) is an 80486 compatible). As part of the alternate processor boot operations, it disables CPUs installed in slots with a lower number than its own that have not checked 'in.' If the alternate boot CPU fails to boot, then another CPU (of the same processor type) has a chance to boot the system and so forth.

If after all the CPUs installed in the system have booted, but one or more have failed to check-in, the boot CPU disables any CPUs which failed to check-in.

The slot select logic **148** is described in more detail in reference to the CPU1 **120** installed in slot **1** as the default boot processor and the CPU2 **132** installed in slot 2 as the alternate boot processor. The slot select circuit **148**A for CPU1 **120** is identical to the slot select circuit **148**B for CPU2 **132**, however, the circuits **148**A, **148**B provide slightly different functions depending on in which slot each circuit is installed.

Installed in Slot 1

In the present embodiment, slot **1** contains the default boot CPU. Thus, the CPU1 **120** in slot **1** is first allowed to boot the system. The general function of the slot select logic **148**A installed in slot **1** is to allow the CPU1 **120** to perform system boot operations. However, if the

CPU1 120 fails to boot properly, the slot select logic 148A disables CPU1 120 and control automatically shifts to the next CPU in the system—in the depicted embodiment, the CPU2 132 in slot 2—as described in more detail below.

The SLOT_ONE signal line 210, the SLOT_SELECT signal line 193 and a TERM- signal line 216 carry the main control signals for the slot select logic 148A and the resulting signal from the logic 148A is a reset signal on the RESET-signal line 208 (FIG. 4). The values of the reset signal for varying combinations of signal levels on the SLOT_SELECT signal line 193, the TERM- signal line 216 and the SLOT_ONE signal line 210 for circuit boards installed in slot 1 are shown in Table 1 below. Corresponding values for circuit boards installed in slots 2–8 are shown in Table 2 below.

TABLE 1

	Slot 1			
	SLOT_ONE	SLOT_SELECT	TERM—	RESET—
BOARD RESET	1	1	1	0
FAIL	1	1	0	0
RUN	1	0	0	1
SHUT DOWN	1	0	0	0

As seen in row 1 of Table 1, at board reset, the SLOT_SELECT line 193 and the TERM- signal line 216 from the timer 184 are both at a high level, regardless of the slot in which a circuit board is installed. For a circuit board installed in slot 1, the SLOT_ONE signal line 210 is also high because the AND gate 174 in conjunction with the inverters 190, 191, and 192 decodes the slot ID on signal lines 194–197 to produce an active signal on the SLOT_ONE signal line 210. This signal is inverted by the inverter 188 to produce a low signal on the SLOT_ONE- signal line 212. The high signal on the SLOT_SELECT signal line 193 is inverted by the inverter 190 to produce a low signal on a SLOT_SELECT- signal line 214.

The function of the SLOT_SELECT line 193 changes for the slot select circuit 148A in slot 1 depending on the output of the timer. While the timer is running, the signal on the SLOT_SELECT signal line 193 has no effect on the slot select circuit 148A. When the timer stops running, then a low signal on the SLOT_SELECT signal line 193 indicates that the slot is enabled, while a high signal disables the CPU1 120 in slot 1.

As long as the BD_RESET- signal line 204 is active, all circuit boards in the system are held in reset because the active signal on the BD_RESET- signal line 204 activates the OR gate 182 connected to the RESET-signal line 208 which provides a reset signal to the microprocessor 209 (FIG. 4).

The BD_RESET- signal line 204 becoming active also loads the time-out value from the time select logic 185 into the timer 184, and clears the latch 186 which produces a low signal on the RUN- signal line 218 and a high signal on the RUN signal line 221. The low signal on the RUN- signal line 218 enables the timer 184 to start counting down from the value provided by the time select logic 185 once the BD_RESET- signal line 204 becomes inactive. The timer 184 continues to run until the latch 186 produces a high signal on the RUN-signal line 218. This occurs if the AND gate 172 is activated or the STOP- signal line 222, which is connected to the preset input of the latch 186, becomes active, as further explained herein.

Until the timer, which is counting down, reaches the terminal count of zero, the signal on the TERM- signal line 216 remains high. With the signal on the TERM-signal line 216 at a high level, the AND gate 176 also remains inactive. However, because the SLOT_SELECT- signal line 214 and the SLOT_ONE- signal line carry low signals at this point, if the timer 184 reaches zero before the signal on the SLOT_SELECT-signal line 214 becomes high (the CPU1 120 failed to boot), the signal on the TERM- signal line 216, which becomes active when the timer reaches zero, activates the AND gate 176 to provide a low signal on the SLOT_1_RESET- signal line 220.

The low signal on the SLOT_1_RESET- signal line 220 activates the OR gate 182 to produce a low signal on the RESET- signal line 208 and hold the microprocessor 209 in reset (i.e. to disable the circuit board).

The active signal on the TERM- signal line 216 activates the OR gate 183 to produce an active signal on the STOP-signal line 222. The active signal on the STOP-signal line 222 stops the timer 184 by activating the preset connection to the latch 186. Thus, the latch 186 drives the RUN- signal line 218 high to stop the timer 184 from operating and also drives the RUN signal line 221 low which also activates the OR gate 183 to produce the active signal on the STOP- signal line 222. Advantageously, this prevents the latch 186 from activating the RUN- signal line 218 until another board reset is signalled on the BD_RESET- signal line 204. The signals stabilize as indicated in the FAIL row in Table 1.

Because the signal on the SLOT_ONE- signal line 212 is low, the NAND gate 178 remains inactive with a high signal on a signal line 179. Thus, the signal from the NAND gate 178 does not effect operation of the OR gate 182 if the circuit board is installed in slot 1. Accordingly, as long as the BD_RESET- signal line 204 and the CPU_RESET- signal line 206 also remain inactive, the signal from AND gate 176 on the SLOT_1_RESET- signal line 220 controls the OR gate 182.

If the CPU1 120 boots properly, it changes the level of SS1 signal line 162 from the IOSM 108 by writing to the memory or I/O mapped slot select register in the IOSM 108 as previously explained. The SS1 signal line 162 connects to the SLOT_SELECT (SS) signal line 193 for slot select circuit 148A. The change in the level on the SS1 signal line results in a low level on the SLOT_SELECT signal line 193 and a high level on the SLOT_SELECT- signal line 214. The high level on the SLOT_SELECT- signal line 214 prevents the AND gate 176 from becoming active when the TERM- signal line 216 becomes active upon the timer 184 reaching zero. Thus, when the CPU1 120 boots properly, the SLOT_1_RESET- signal line 220, the OR gate 182, and the RESET- signal line 208 remain inactive even if the TERM- signal line 216 becomes active.

Even after the CPU1 120 in slot 1 has booted, the timer continues to time-out to activate the TERM-signal line 216. The timer stops at zero because the

active signal on the TERM- signal line **216** activates the OR gate **183** to produce an active signal on the STOP- signal line **222**. The active signal line the STOP- signal line **222** presets the latch **186** to produce a high signal on the RUN- signal line **218** which stops the timer **184** from operating, and leaves the timer **184** with the TERM- signal line **216** active. Accordingly, when the CPU1 in slot **1** properly boots, the slot select logic becomes stable with AND gate **176** and the OR gate **182** inactive so that signal line **208** is inactive and the processor is not held in reset. The stable signals are indicated by the 'RUN' row in Table 1.

From the state shown in the RUN row of Table 1, the IOSM **108** can reset the CPU1 **120** by activating the corresponding SS1 signal line **162** which connects to the SLOT_SELECT signal line **193** of the slot select circuit **148A**. An active signal on the SLOT_SELECT signal line **193** resets the CPU1 **120** because the corresponding active signal on the SLOT_SELECT- signal line **214** activates the AND gate **174** (the TERM- signal line **216** and the SLOT_ONE- signal line **212** are also active) to activate the SLOT_1_RESET- signal line **220** which activates the OR gate **182** as previously explained. The signals stabilize as indicated in the 'Shut Down' row of Table 1.

With the CPU1 **120** installed in slot **1**, the AND gate **172** remains inactive and does not effect operation of the slot select logic **148A** because the SLOT_ONE- signal line **212** is low which prevents the AND gate **172** from being activated.

Installed in Slot 2–8

The general function of the slot select logic **148** installed in any of the slots **2–8** is to hold the corresponding CPU in reset until CPUs in previous slots have had sufficient (pre-determined) time to boot. In other words, for the CPU2 **132** installed in slot **2**, the slot select logic **148B** holds the CPU **132** in a reset state until sufficient time has elapsed for the CPU1 **120** to boot. If the CPU1 **120** does not complete boot operations within the time allotted, then the CPU2 **132** takes over default boot operations as an alternative boot processor instead of CPU1 **120**. If the CPU1 **120** does boot properly, then the CPU2 **132** checks in as an additional processor rather than taking over as a boot processor.

Table 2 below indicates the value of the signals on the RESET- signal line **208** for various combinations of signals on the TERM- signal line **216**, the SLOT_SELECT signal line **193** and the SLOT_ONE signal line **210**. The function of the slot select logic **148** for slots **2–8** is explained with reference to CPU2 **132** in slot **2** of the bus **104**.

TABLE 2

	Slots 2–8			
	SLOT_ONE	SLOT_SELECT	TERM—	RESET—
BD_RESET	0	1	1	0
ALT BOOT	0	1	0	1
RUN	0	0	1	1
FAIL	0	0	0	0

Upon a board reset, the active pulse on the BD_RESET-signal line **204** loads the timer **184** with the value from the time select logic **185** corresponding to the slot **2** four-bit slot ID on lines **194–197** and the timer **184** begins to operate (as explained for slot **1**). The slot ID value loaded in the timer **184** is greater than the value for slot **1** and is pre-selected to allow enough time for the CPU1 **120** in slot **1** to execute initialization operations.

At board reset, the SLOT_SELECT signal line **193** is high, as seen in Table 2, which is inverted by the inverter **189** to a low level on the SLOT_SELECT- signal line **214**. The TERM-signal line **216** is high because the timer **184** is active. The high signal on the TERM- signal line **216** and the low level on the SLOT_SELECT- signal line **214** activate the EXOR gate **180** which produces a high level on a signal line **181** connected to the NAND gate **178**. The high level signals on the signal line **181** and the SLOT_ONE- signal line **212** combine to activate the NAND gate **178** which produces an active signal on the signal line **179**. The active signal on signal line **179** activates the OR gate **182** to produce an active signal on the RESET- signal line **208** to hold the CPU2 **132** in reset.

For CPU2 **132** in slot **2**, the SLOT_ONE signal line is inactive because the AND gate **174** in conjunction with the inverters **190**, **191**, and **192** does not detect the correct slot ID on signal lines **194–197** to indicate that the CPU2 **132** is installed in slot **1**. Thus, the SLOT_ONE- signal line **212** is high because the inactive signal on the SLOT_ONE signal line **210** is inverted by the inverter **188** to produce a high signal on the SLOT_ONE- signal line **212**.

If the CPU1 **120** in slot **1** boots properly before the timer **184** for the CPU2 **132** in slot **2** times out, the timer **184** for the CPU2 **132** is prevented from counting and the RESET-signal line **208** becomes inactive, allowing the CPU2 **132** to boot. The signals stabilize as seen in the 'RUN' row of Table 2.

The timer stops because when the CPU1 **120** boots properly, the CPU1 **120** changes the level of the SS2 signal line **164** in the IOSM **108** which is connected to the SLOT_SELECT line **193** for the slot select circuit **148B**. This results in a high signal on the SLOT_SELECT- signal line **214**. The high signal on the SLOT_SELECT- signal line **214**, the high signal on the SLOT_ONE- signal line **212**, and the high signal on the TERM- signal line **216** combine to activate the AND gate **172** to produce a high signal for the input to the latch **186**. A pulse on the CLK signal line **202** causes the high signal to propagate through the latch **186** to the RUN- signal line **218** to stop the timer **184**. This results in a low signal on the RUN signal line **221**, which causes the OR gate **183** to activate the STOP- signal line **222**. The active signal on the STOP-signal line **221** presets the latch **186** to hold the RUN- signal line **218** high and the RUN signal line **221** low. Accordingly, the signal on the TERM- signal line **216** remains high because the timer did not time-out.

The RESET- signal line **208** becomes inactive because the low signal on the SLOT_SELECT signal line **193** is inverted by the inverter **189** to a high level on the SLOT_SELECT- signal line **214**. The high signals on the TERM- signal line **216** and the SLOT_SELECT- signal line **214** combine to cause the EXOR gate **180** to become inactive and produce an inactive low signal on

the signal line **181**. The inactive signal on the signal line **181** from the EXOR gate **180** inactivates the NAND gate **178** which produces an inactive signal on the signal line **179** to inactivate the NOR gate **182**. The AND gate **182** produces an inactive signal on the RESET- signal line **208** which allows the microprocessor of the CPU2 **132** to exit the reset state and initialize itself as explained above.

If the CPU1 **120** in slot **1** does not boot (proper boot is indicated by a change in the level of the SLOT_SELECT signal line **193** from high to low) before the timer **184** in slot **2** times-out, then the CPU2 **132** exits the reset state and attempts to boot the system as an alternate boot CPU.

The TERM- signal line **216** becomes active when the timer **184** times-out. The SLOT_SELECT signal line **193** remains high because the CPU1 **132** (which failed to boot) did not change the level of this signal. The active (low) signal on the TERM- signal line **216** and low signal on the SLOT_SELECT-signal line **214** deactivate the EXOR gate **180** which produces an inactive signal on the signal line **181** to deactivate the NAND gate **178**. The inactive NAND gate **178** provides an inactive signal on the signal line **179** which allows the 0R gate **182** to become inactive and produce an inactive signal on the RESET- signal line **208**. Accordingly, the CPU2 **132** exits from the reset state and takes over boot operations in place of the CPU1 **120**. By checking the level of the SLOT_SELECT signal line **193**, the CPU2 **132** determines that the CPU1 **132** failed to boot and that it (the CPU2 **132**) is now the boot CPU. If the CPU2 **132** boots properly, then the signals stabilize as indicated in the 'ALT BOOT' row of Table 2.

The functional level of the signal on the SLOT_SELECT signal line **193** changes depending upon the output of the timer **184** for slot select logic **148B** installed in slot **2** (as well as for slot select logic **148** installed in all other slots **3–8**). While the timer is running, a low signal on the SLOT_SELECT signal **193** freezes the timer and indicates that slot **2** is enabled and a high on the SLOT_SELECT signal line **193** disables the slot. When the timer **184** stops running, a low signal on the SLOT_SELECT signal line **193** indicates that the slot **2** is disabled, and a high signal indicates that the slot is enabled. The change in level on the SLOT_SELECT signal line **193** from high to low if the CPU2 **132** also failed to boot occurs because the timer **184** for CPU3 **160** in slot **3** will time-out and begin to boot the system. When the CPU3 **160** begins boot operations, it disables the CPU2 **132** by changing the level of the SS2 signal line **164** in the IOSM **108** to a low level.

Thus, regardless of which CPU boots the system, the boot CPU can let other CPUs time out, at which time they will begin to initialize. Otherwise, the boot CPU can specifically enable them before they time-out by use of the SLOT_SELECT signal line **193**, or the boot CPU can disable the timer on any or all CPUs and hold the corresponding CPUs in reset by toggling the corresponding SLOT_SELECT line **193**.

Accordingly, after the CPU1 **120** in slot **1** has failed to boot properly, and the CPU2 **132** in slot **2** takes over boot operation, the timer **184** of slot select circuit **148B** has stopped operating and the signal on the TERM- signal line **216** is active. The level on the SLOT_SELECT line **193** remains high as explained above which indicates at this point that the CPU2 **132** is enabled. However, if the level on the SLOT_SELECT signal line **193** now changes to low (driven by the IOSM **108** with the SS2 signal line **164**), indicating that the CPU2 **132** failed to boot the system within the time allotted for the CPU2 **132** to complete boot operations, then the CPU2 **132** enters a reset state because the active signal on the TERM- signal line **216** combines with the now high signal on the SLOT_SELECT- signal line **214** in the EXOR gate **180** to produce an active signal on the signal line **181**. The active signal on the signal line **181** combines with the high signal on the SLOT_ONE- signal line **212** to activate the NAND gate **178** which in turn provides an active signal on the signal line **179** to activate the OR gate **182** to hold the microprocessor of the CPU2 **132** in reset. Accordingly, the signals become stable as indicated in the 'FAIL' row of Table 2.

Although preferred embodiments of the present invention has been described and illustrated above, those skilled in the art will appreciate that various changes and modifications can be made to the present invention without departing from its spirit. Accordingly, the scope of the present invention is deemed to be limited only by the scope of the following appended claims.

What is claimed is:

1. A system for coordinating the initialization of multiprocessor systems utilizing a multiprocessor backplane bus, wherein said backplane bus has connector slots for connecting multiple central processing units, memory modules, and other input/output modules, said system for coordinating comprising:

a default boot central processing unit coupled to said backplane bus in a first slot;

at least one alternate boot central processing unit coupled to said backplane bus in a second slot, said alternate boot central processing unit substantially compatible with said default boot central processing unit;

a first slot select circuit coupled to said default boot central processing unit, said first slot select circuit configured to release said default boot central processing unit from a reset state upon system start-up and allow said default central processing unit to execute said boot code, and further configured to place said default boot central processing unit in a reset state if a first predetermined time-out period, selected by a first respective slot identification code, elapses before said default central processing unit successfully boots; and

a second slot select circuit coupled to said alternate boot central processing unit, wherein said second slot select circuit is configured to hold said alternate processing unit in a reset state until either said default boot central processing unit successfully boots, or a second predetermined time-out period, selected by a second respective slot identification code, elapses, indicating that said default boot central processing unit failed to boot the system, and wherein said second slot select circuit is further configured to release said alternate boot central processing unit from said reset state after said default central processing unit successfully boots or said second predetermined time-out period elapses.

2. The system for coordinating the initialization of multiprocessor systems of claim **1**, wherein said second predetermined time-out period is longer than said first predetermined time-out period.

3. The system for coordinating the initialization of multiprocessor systems of claim **1**, wherein said second predetermined time-out period is equal to said first predetermined time-out period.

4. The system for coordinating the initialization of multiprocessor systems of claim 1, wherein said default boot central processing unit comprises an INTEL 80486 based compatible processor and supporting resources, and wherein said alternate boot central processing unit comprises and INTEL 80486 based compatible processor and supporting resources.

5. A system for coordinating the initialization and self-test of multiprocessor systems, said multiprocessor system having a backplane bus with slots for connecting multiple central processing units, memory modules, and other input/output modules, said system comprising:

a first central processing unit coupled to said backplane bus in a first slot, said first central processing unit assigned to boot first upon system start-up;

a second central processing unit coupled to said backplane bus in a second slot, said second central processing unit substantially compatible with said first central processing unit;

a first non-volatile memory having memory locations for storing central processing unit boot instructions in executable form, said first non-volatile memory directly accessible by said first and said second central processing units;

first means coupled to said first central processing unit for releasing said first central processing unit from a reset state during system start-up to allow said first central processing unit to execute said central processing unit boot instructions, and for placing said first central processing unit in a reset state if a first predetermined time-out period, selected by a first respective slot identification code, elapses before said first central processing unit successfully boots; and

second means coupled to said second central processing unit for holding said second central processing unit in the reset state until either said first central processing unit successfully boots, or a second predetermined time-out period, selected by a second respective slot identification code, elapses, and for releasing said second central processing unit from said reset state after said first central processing unit successfully boots or said second predetermined time-out period elapses.

6. The system for coordinating the initialization and self-test of multiprocessor systems of claim 5, further comprising first slot identification code circuitry coupled to said first slot and a second slot identification code circuitry coupled to said second slot, said first and said second slot identification code circuitry configured to provide mutually exclusive slot identification codes to a central processing unit, memory module, or input-/output module coupled to the backplane bus in the respective slot.

7. The system for coordinating the initialization of multiprocessor systems of claim 5, wherein said second predetermined time-out period is equal to said first predetermined time-out period.

8. The system for coordinating the initialization of multiprocessor systems of claim 5, wherein said second predetermined time-out period is longer than said first pre-determined time-out period.

* * * * *

PTO/SB/57 (01-18)
Approved for use through 11/30/2024. OMB 0651-0064
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

(Also referred to as FORM PTO-1465)

REQUEST FOR *EX PARTE* REEXAMINATION TRANSMITTAL FORM

Address to:
Mail Stop *Ex Parte* Reexam
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Attorney Docket No.: 185945.012600

Date: 9/1/23

1. ☒ This is a request for *ex parte* reexamination pursuant to 37 CFR 1.510 of patent number 7,624,302 issued November 24, 2009. The request is made by:
   ☐ patent owner. ☒ third party requester.

2. ☒ The name and address of the person requesting reexamination is:
   Unified Patents, LLC
   4445 Willard Ave., Suite 600
   Chevy Chase, MD 20815

3. Requester ☐ asserts small entity status (37 CFR 1.27) or ☐ certifies micro entity status (37 CFR 1.29). Only a patent owner requester can certify micro entity status. Form PTO/SB/15A or B must be attached to certify micro entity status.

4. This request is accompanied by payment of the reexamination fee as set forth in:
   ☒ 37 CFR 1.20(c)(2); **or**
   ☐ 37 CFR 1.20(c)(1). **In checking this box for payment of the fee set forth in 37 CFR 1.20(c)(1), requester asserts that this request has forty (40) or fewer pages and complies with all other requirements of 37 CFR 1.20(c)(1).**

   Payment of the reexamination fee is made by the method set forth below.
   a. ☐ A check in the amount of $_____ is enclosed to cover the reexamination fee;
   b. ☒ The Director is hereby authorized to charge the reexamination fee to Deposit Account No. 50-6990;
   c. ☐ Payment by credit card. Form PTO-2038 is attached; **or**
   d. ☐ Payment made via EFS-Web.

   ☒ In addition, the Director is hereby authorized to charge any fee deficiencies to Deposit Account No. 50-6990.

5. ☒ Any refund should be made by ☐ check or ☐ credit to Deposit Account No. 50-6990. 37 CFR 1.26(c). If payment is made by credit card, refund must be to credit card account.

6. ☒ A copy of the patent to be reexamined having a double column format on one side of a separate paper is enclosed. 37 CFR 1.510(b)(4).

7. ☐ CD-ROM or CD-R in duplicate, Computer Program (Appendix) or large table
   ☐ Landscape Table on CD

A Federal agency may not conduct or sponsor, and a person is not required to respond to, nor shall a person be subject to a penalty for failure to comply with an information collection subject to the requirements of the Paperwork Reduction Act of 1995, unless the information collection has a currently valid OMB Control Number. The OMB Control Number for this information collection is 0651-0064. Public burden for this form is estimated to average 18 minutes per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the information collection. Send comments regarding this burden estimate or any other aspect of this information collection, including suggestions for reducing this burden to the Chief Administrative Officer, United States Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450 or email InformationCollection@uspto.gov. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **If filing this completed form by mail, send to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

8. ☐ Nucleotide and/or Amino Acid Sequence Submission
If applicable, items a. – c. are required.
   a. ☐ Computer Readable Form (CRF)
   b. Specification Sequence Listing on:
      i. ☐ CD-ROM (2 copies) or CD-R (2 copies) **or**
      ii. ☐ paper
   c. ☐ Statements verifying identity of above copies.

9. ☐ A copy of any disclaimer, certificate of correction or reexamination certificate issued in the patent is included.

10. ■ Reexamination of claim(s) 1-27 is requested.

11. ■ A copy of every patent or printed publication relied upon is submitted herewith including a listing thereof on Form PTO/SB/08, PTO-1449, or equivalent.

12. ■ An English language translation of all necessary and pertinent non-English language patents and/or printed publications is attached.

13. ■ The attached detailed request includes at least the following items:
   a. A statement identifying each substantial new question of patentability based on prior patents and printed publications. 37 CFR 1.510(b)(1).
   b. An identification of every claim for which reexamination is requested, and a detailed explanation of the pertinency and manner of applying the cited art to every claim for which reexamination is requested. 37 CFR1.510(b)(2).

14. ☐ A proposed amendment is included (only where the patent owner is the requester). 37 CFR 1.510(e).

15. ■ It is certified that the statutory estoppel provisions of 35 U.S.C. 315(e)(1) or 35 U.S.C. 325(e)(1) do not prohibit requester from filing this *ex parte* reexamination request. 37 CFR 1.510(b)(6).

16. Service
   a. ■ It is certified that a copy of this request (if filed by other than the patent owner) has been served in its entirety on the patent owner as provided in 37 CFR 1.33(c).

      The name and address of the party served are:

      Foras Technologies Ltd. c/o 56436 - Hewlett Packard Enterprise
      3404 E. Harmony Road, Mail Stop 79
      Fort Collins, CO 80528

      Date of Service: 9/1/23

   OR

   b. ☐ A duplicate copy is enclosed since service on patent owner was not possible. An explanation of the efforts made to serve patent owner **is attached**. See MPEP 2220.

17. Correspondence Address: Direct all communication about the reexamination to:

☒ The address associated with Customer Number: 67395

OR

☐ Firm or Individual Name _____
(at the address identified below)

Address

City	State	Zip

Country

Telephone	Email

18. ☐ The patent is currently the subject of the following concurrent proceeding(s):

a. ☐ Copending reissue Application No. _____

b. ☐ Copending reexamination Control No. _____

c. ☐ Copending Interference No. _____

d. ☐ Copending litigation styled:

_____

_____

WARNING : Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

/Greg Lunt/ — Authorized Signature

9/1/23 — Date

Greg Lunt — Typed/Printed Name

57354 — Registration No.

☐ For Patent Owner Requester ☒ For Third Party Requester

Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. The United States Patent and Trademark Office (USPTO) collects the information in this record under authority of 35 U.S.C. 2. The USPTO's system of records is used to manage all applicant and owner information including name, citizenship, residence, post office address, and other information with respect to inventors and their legal representatives pertaining to the applicant's/owner's activities in connection with the invention for which a patent is sought or has been granted. The applicable Privacy Act System of Records Notice for the information collected in this form is COMMERCE/PAT-TM-7 Patent Application Files, available in the Federal Register at 78 FR 19243 (March 29, 2013). https://www.govinfo.gov/content/pkg/FR-2013-03-29/pdf/2013-07341.pdf

Routine uses of the information in this record may include disclosure to: 1) law enforcement, in the event that the system of records indicates a violation or potential violation of law; 2) a Federal, state, local, or international agency, in response to its request; 3) a contractor of the USPTO having need for the information in order to perform a contract; 4) the Department of Justice for determination of whether the Freedom of Information Act (FOIA) requires disclosure of the record; 5) a Member of Congress submitting a request involving an individual to whom the record pertains, when the individual has requested the Member's assistance with respect to the subject matter of the record; 6) a court, magistrate, or administrative tribunal, in the course of presenting evidence, including disclosures to opposing counsel in the course of settlement negotiations; 7) the Administrator, General Services Administration (GSA), or their designee, during an inspection of records conducted by GSA under authority of 44 U.S.C. 2904 and 2906, in accordance with the GSA regulations and any other relevant (i.e., GSA or Commerce) directive, where such disclosure shall not be used to make determinations about individuals; 8) another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)); 9) the Office of Personnel Management (OPM) for personnel research purposes; and 9) the Office of Management and Budget (OMB) for legislative coordination and clearance.

If you do not furnish the information requested on this form, the USPTO may not be able to process and/or examine your submission, which may result in termination of proceedings, abandonment of the application, and/or expiration of the patent.

Additional Uses

Additional USPTO uses of the information in this record may include disclosure to: 1) the International Bureau of the World Intellectual Property Organization, if the record is related to an international application filed under the Patent Cooperation Treaty; 2) the public i) after publication of the application pursuant to 35 U.S.C. 122(b), ii) after issuance of a patent pursuant to 35 U.S.C. 151, iii) if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspections, or an issued patent, or iv) without publication of the application or patent under the specific circumstances provided for by 37 CFR 1.14(a)(1)(v)-(vii); and/or 3) the National Archives and Records Administration, for inspection of records.

US007085959B2

(12) **United States Patent**
Safford

(10) **Patent No.: US 7,085,959 B2**
(45) **Date of Patent: Aug. 1, 2006**

(54) **METHOD AND APPARATUS FOR RECOVERY FROM LOSS OF LOCK STEP**

(75) Inventor: **Kevin David Safford**, Fort Collins, CO (US)

(73) Assignee: **Hewlett-Packard Development Company, L.P.**, Houston, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 651 days.

(21) Appl. No.: **10/187,833**

(22) Filed: **Jul. 3, 2002**

(65) **Prior Publication Data**

US 2004/0006722 A1 Jan. 8, 2004

(51) **Int. Cl.**
G06F 11/00 (2006.01)

(52) **U.S. Cl.** **714/11**

(58) **Field of Classification Search** 714/10, 714/11, 12, 13

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,589,066	A	5/1986	Lam et al.	
5,732,209	A *	3/1998	Vigil et al.	714/30
6,065,135	A	5/2000	Marshall et al.	
6,148,348	A	11/2000	Garnett et al.	
6,263,452	B1	7/2001	Jewett et al.	
6,378,021	B1 *	4/2002	Okazawa et al.	710/317
6,615,366	B1 *	9/2003	Grochowski et al.	714/10
6,751,749	B1 *	6/2004	Hofstee et al.	714/11
6,772,368	B1 *	8/2004	Dhong et al.	714/11
2002/0073357	A1 *	6/2002	Dhong et al.	714/19
2002/0144177	A1 *	10/2002	Kondo et al.	714/13
2003/0126498	A1 *	7/2003	Bigbee et al.	714/10
2004/0019771	A1 *	1/2004	Quach	712/229
2004/0153731	A1 *	8/2004	Aino et al.	714/6
2004/0153857	A1 *	8/2004	Yamazaki et al.	714/43

OTHER PUBLICATIONS

Tamir, Yuval, "Self-Checking Self-Repairing Computer Nodes Using the Mirror Processor", IEEE Journal Of Solid-State Circuits, vol. 27. No. 1, Jan. 1992.

Search Report issued on Dec. 30, 2003 in counterpart foreign application in GB under application No. 315295.6.

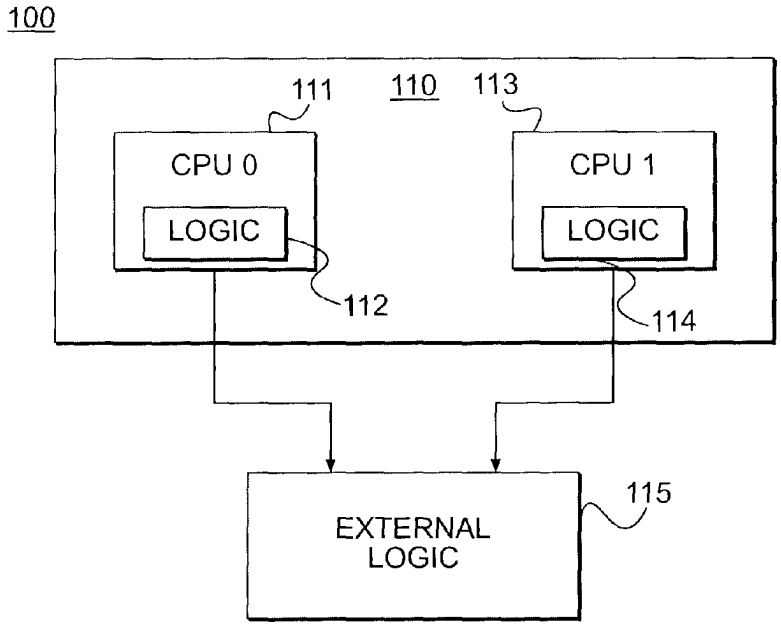
* cited by examiner

Primary Examiner—Bryce P. Bonzo

(57) **ABSTRACT**

An apparatus, operating on an advanced multi-core processor architecture, and a corresponding method, are used to enhance recovery from loss of lock step in a multi-processor computer system. The apparatus for recovery from loss of lock step includes multiple processor units operating in the computer system, each of the processor units having at least two processor units operating in lock step, and at least one idle processor unit operating in lock step; and a controller coupled to the two processor units operating in lock step and the idle processor unit. The controller includes mechanisms for copying an architected state of each of the two lock step processor units to the idle processor unit.

14 Claims, 4 Drawing Sheets



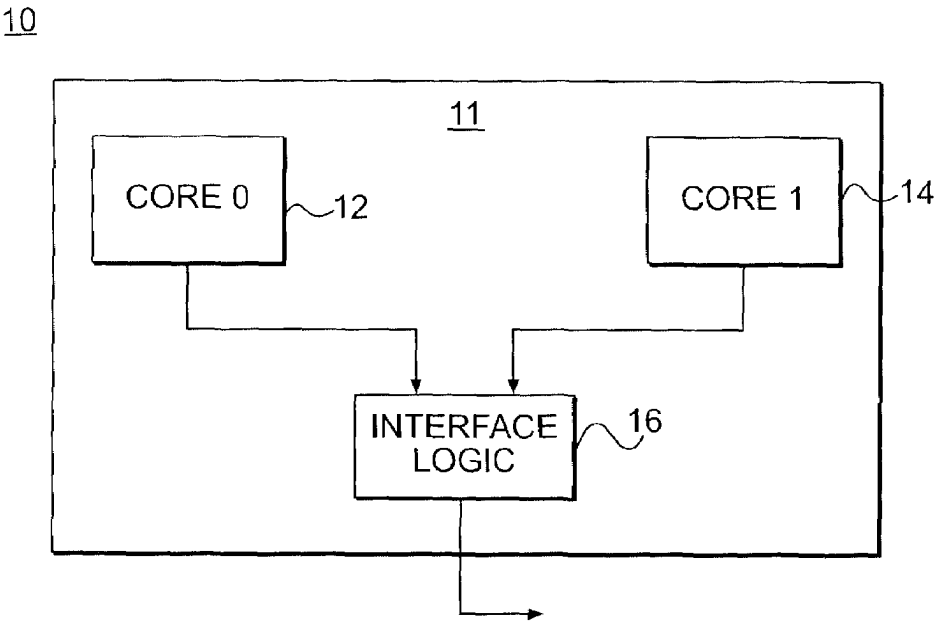


FIG. 1A
PRIOR ART

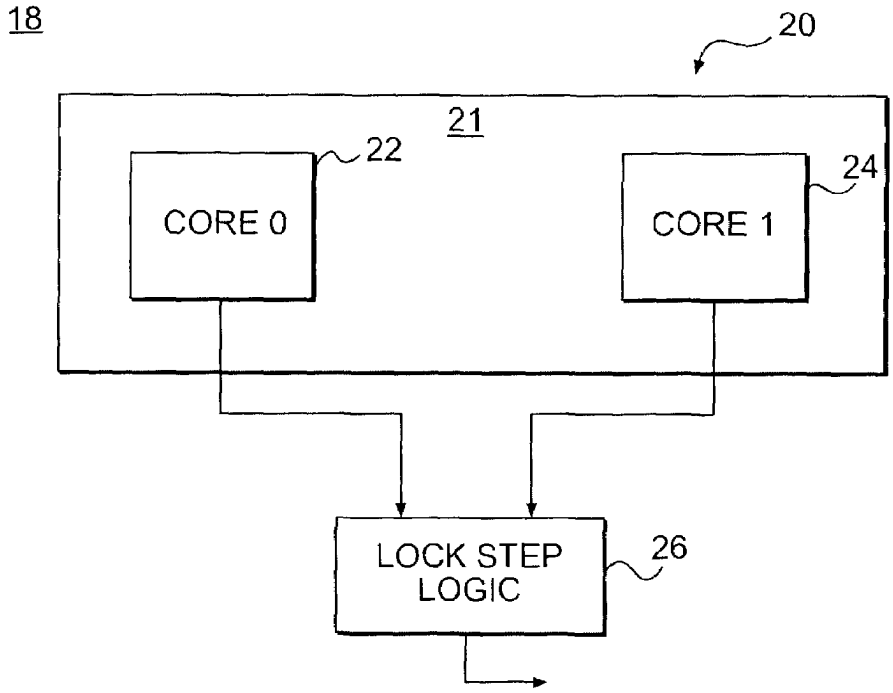


FIG. 1B
PRIOR ART

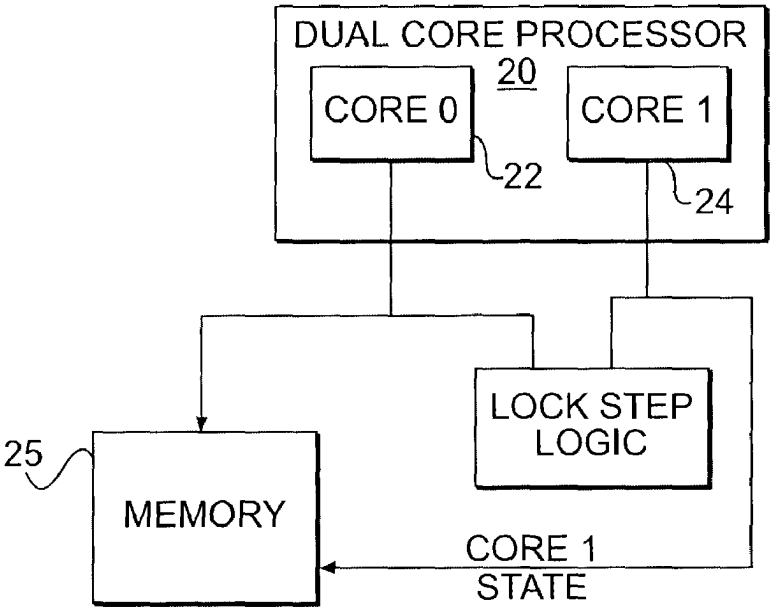


FIG. 1C
PRIOR ART

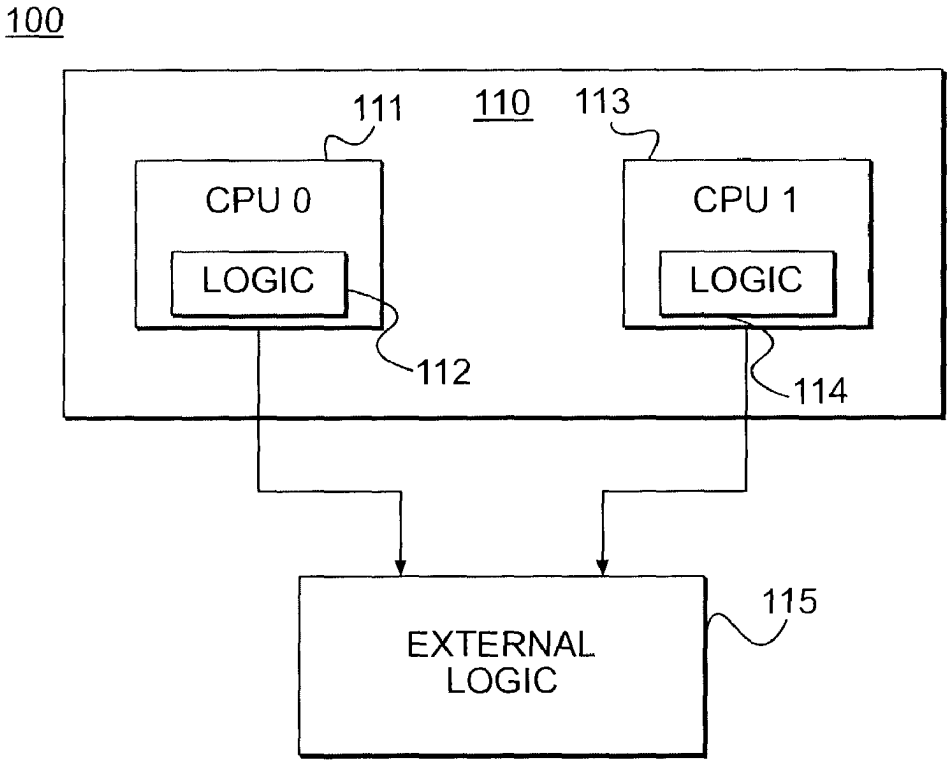


FIG. 2

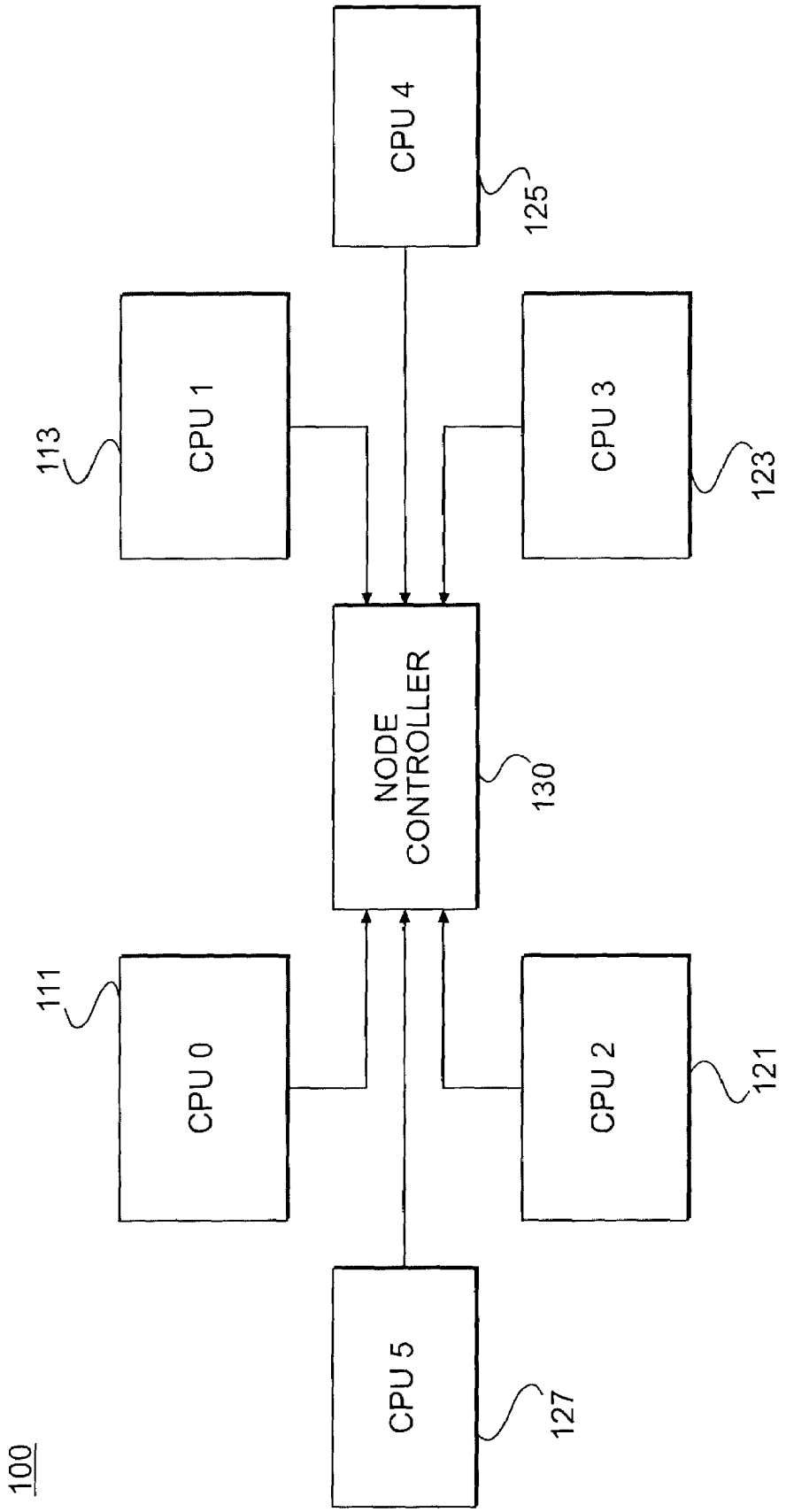

CPU 4
125
CPU 1
113
CPU 3
123
NODE CONTROLLER
130
111
CPU 0
CPU 2
121
CPU 5
127
100

FIG. 3

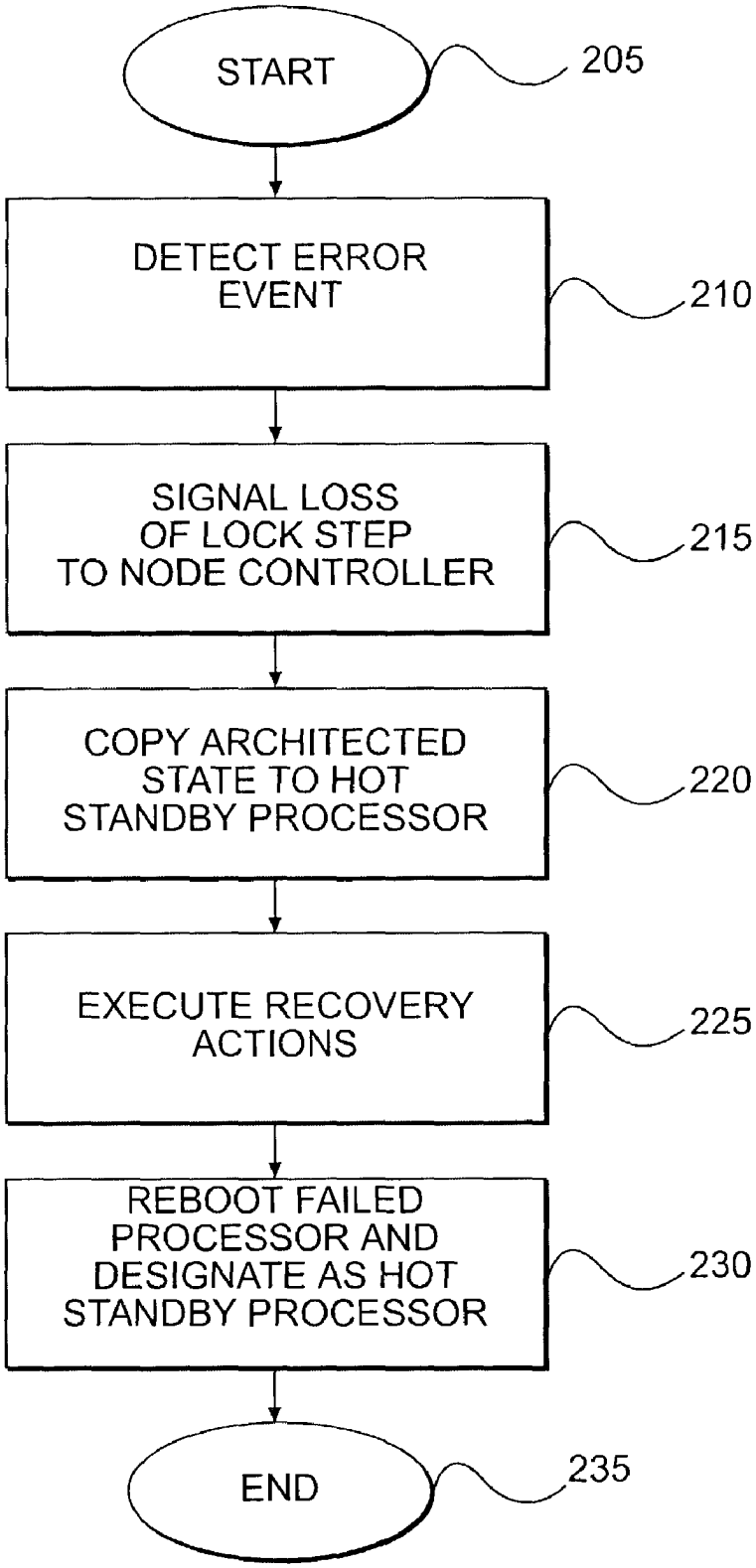

START
205
DETECT ERROR EVENT
210
SIGNAL LOSS OF LOCK STEP TO NODE CONTROLLER
215
COPY ARCHITECTED STATE TO HOT STANDBY PROCESSOR
220
EXECUTE RECOVERY ACTIONS
225
REBOOT FAILED PROCESSOR AND DESIGNATE AS HOT STANDBY PROCESSOR
230
END
235

FIG. 4

METHOD AND APPARATUS FOR RECOVERY FROM LOSS OF LOCK STEP

TECHNICAL FIELD

The technical field is computer systems employing lock stepped microprocessors.

BACKGROUND

Advanced computer architectures may employ multiple microprocessors. Some advanced computer architectures may employ multiple microprocessors on one silicon chip. In a typical application, two microprocessors may be implemented on a single silicon chip, and the implementation may be referred to as a dual core processor. Two or more of the multiple microprocessors may operate in a lock step mode, meaning that each of the lock stepped microprocessors process the same code sequences, and should, therefore, produce identical outputs. FIG. **1**A illustrates a typical implementation of a dual core processor. A dual core processor **10** includes a silicon chip **11** having microprocessor core **12** (core **0**) and microprocessor core **14** (core **1**). The microprocessor cores **12** and **14** are coupled to an interface logic **16** that monitors external communications from the microprocessor cores **12** and **14**. In the dual core processor **10**, the microprocessor cores **12** and **14** operate as independent entities. While the dual core processor **10** has advantages in terms of size and processing speed, the reliability of the dual core processor **10** is not significantly better than that of two single core processors.

To enhance reliability, the dual core processor, or other multiple microprocessor architected computer systems, may employ lock step features. FIG. **1**B is a diagram of a prior art dual core processor that uses lock step techniques to improve overall reliability. In FIG. **1**B, a computer system **18** includes a dual core processor **20** having a single silicon chip **21**, on which are implemented microprocessor core **22** and microprocessor core **24**. To employ lock step, each of the microprocessor cores **22** and **24** process the same code streams. To ensure reliable operation of the dual core processor **20**, each of the microprocessors **22** and **24** may operate in "lock step." An event that causes a loss of lock step can occur on either or both of the microprocessor cores **22** and **24**. An example of such an event is a data cache error. A loss of lock step, if not promptly corrected, may cause the computer system **18** to "crash." That is, a failure of one microprocessor core may halt processing of the dual core processor **20**, and the computer system **18**, even if the other microprocessor core does not encounter an error.

To detect a loss of lock step, a lock step logic **26**, which may be external to the chip **21**, compares outputs from the microprocessor cores **22** and **24**. A difference in processing detected by the lock step logic **26** is by definition a loss of lock step. A drawback to the dual core processor architecture shown in FIG. **1**B is that the logic to determine loss of lock step is external to the chip. This configuration imposes delays in determining loss of lock step, and requires additional architectural features.

The dual core processor **20** also makes recovery from a loss of lock step difficult and time-consuming. FIG. **1**C illustrates a current methodology for recovering from a loss of lock step. In FIG. **1**C, the dual core processor **20** is shown coupled to memory **25**. Should the dual core processor **20** suffer a loss of lock step, recovery may be initiated by the memory **25** saving the architected state of one of the microprocessors **22** and **24** (i.e., the microprocessor that is considered "good"). Then, both microprocessors **22** and **24** are reset and reinitialized. Finally, the architected states of each of the microprocessors **22** and **24** is copied from the memory **25** into the microprocessors **22** and **24**, respectively. This prior art methodology for recovery from a loss of lock step makes the microprocessors **22** and **24** unavailable for an amount of time. If the amount of time required for recovery is too long, the computer system **18** employing the dual core processor **20** may "crash."

SUMMARY

An apparatus, operating on an advanced multi-core processor architecture, and a corresponding method, are used to enhance recovery from loss of lock step in a computer system. In an embodiment, the apparatus for recovery from loss of lock step comprises a plurality of processor units operating in the computer system, each of the processor units comprising at least two processor units operating in lock step, and at least one idle processor unit operating in lock step; and a controller coupled to at least the at least two processor units operating in lock step and the at least one idle processor unit, the controller comprising means for copying an architected state of each of the at least two processor units to the idle processor unit.

The method comprises receiving a loss of lock step signal from a processor unit; receiving a notice from the processor unit experiencing the loss of lock step to take the processor unit off line; and moving an architected state of the processor unit experiencing the loss of lock step to a spare processor unit, wherein the spare processor unit becomes an active processor unit in the computer system.

DESCRIPTION OF THE DRAWINGS

The detailed description will refer to the following figures, in which like numbers refer to like elements, and in which:

FIG. **1**A is a diagram of a prior art dual-core processor;

FIG. **1**B is a diagram of a prior art dual-core processor employing lock step;

FIG. **1**C is a diagram illustrating prior art recovery from loss of lock step;

FIG. **2** is a diagram of a computer system that uses an improved, multi-core processor employing lock step processing;

FIG. **3** illustrates additional architectural features for use in recovery from loss of lock step for the computer system of FIG. **2**; and

FIG. **4** is a flowchart illustrating a process for recovery from loss of lock step in the computer system of FIG. **3**.

DETAILED DESCRIPTION

To improve reliability of processing assets, a computer system employs lock stepped processor cores that operate in a master/checker pair. Each of two processors in the pair processes the same code sequences, and the resulting outputs of the processors are compared by a logic circuit located near external interfaces of the two processors. Any difference in the processor outputs indicates the existence of an error. The logic circuit may then initiate a sequence of steps that halt operation of the two processors. FIG. **2** shows a computer system **100** that employs processors **111** (central processor unit (CPU) **0**) and **113** (CPU **1**), which, in an embodiment, may be located on a common silicon chip or substrate **110**. Alternatively, the processors **111** and **113** may

be implemented on separate substrates. The processors **111** and **113** may operate in an independent mode, or in a lock step mode. When operating in a lock step mode, the processors **111** and **113** will appear to the computer system **100** to be a single processor core, or a logical CPU **0**. The processor **111** may include error detection and signaling logic **112**, and the processor **113** may include error detection and signaling logic **114**. The error detection and signaling logic will be described later.

External logic circuit **115** monitors outputs of the processors **111** and **113** and may be used to detect any differences in the outputs. As noted above, such differences are indicative a potential error in at least one of the processors **111** and **113**. However, which of the processors **111** and **113** is subject to an error condition may not be known. On rare occasions, both the processors **111** and **113** may be subject to an error condition. Such an error condition may lead to a halt in processing of the processors **111** and **113** until the error can be corrected. In other words, any difference in the outputs causes a loss of lock step, and a halt to processing.

To improve availability of the processors assets of the computer system **100**, additional features, such as means for detecting and signaling occurrence of errors, may be incorporated into the computer system **100**. For example, the error detection and signaling logic **112** and **114** may be included in the processors **111** and **113**, respectively, or in other parts of the computer system **100**, to signal an impending loss of lock step. Using the impending loss of lock step signal, the computer system **100** may continue operating (processing) using one of the processors **111** and **113** that did not experience an error. In particular, certain events within either of the lock stepped processors **111** and **113** may be used by the processors **111** and **113**, respectively, to indicate the impending loss of lock step. As an example, and possibly due to completely random circumstances, a data cache error for a cache associated with the processor **111** may occur. Such an error can be completely corrected (i.e., the processor **111** does not need to be replaced), but will guarantee that the processors **111** and **113** will break lock step at some future time because the data cache error causes timing differences between the processors **111** and **113**. The processor **111** may detect the data cache error, and use the detection of this data cache error to signal the logic circuit **115** that the processor **111** is experiencing an error that will cause a loss of lock step, and that the processor **111** is "bad." The logic circuit **115** may then "turn off," thereby ending lock step operations, and processing may continue using the "good" processor **113**. At some future time, recovery from the loss of lock step (and correction of the data cache error) is executed to restore lock step operation of the processors **111** and **113**.

FIG. **3** illustrates further architectural details for recovery from loss of lock step in the computer system **100** of FIG. **2**. In FIG. **3**, the computer system **100** is shown with additional processors **121**, **123**, **125**, and **127**, as well as the processors **111** and **113**. The processors **111**, **113**, **121**, **123**, **125**, and **127** are coupled to node controller **130**. The processors operate as pairs when in lock step (i.e., the processors **111** and **113** are a first pair; the processors **121** and **123** are a second pair; and the processors **125** and **127** are a third pair). From the node controller's perspective, each pair of processors appears as a single (logical) processor. The processor pairs, or processor units, are coupled to a lockstep logic, such as the lockstep logic **115** shown in FIG. **2**, and the lockstep logic is then connected to the node controller **130**. The node controller **130** provides means for copying the architected state of a processor to another processor. In an embodiment, the node controller **130** has available at all time a current architected state of the processors to which the node controller **130** is coupled. In another embodiment, the node controller **130** simply provides means for communication among the processors **111**, **113**, **121**, **123**, **125**, and **127**. For example, the node controller **130** may store the architected state of the processors **111**, **113**, **121**, **123**, **125**, and **127**, either internally in the node controller **130**, or in another component of the computer system **100**. Alternatively, the node controller **130** may allow one processor (e.g., the processor **111**) to copy the architected state of the processor **111** to another processor (e.g., the processor **125**). In yet another alternative embodiment, the node controller **130** may allow a processor that has broken lock step to copy, as part of the process for recovering from loss of lock step, the architected state of the processor to the node controller **130**, which will in turn copy the architected state to a "hot standby" processor.

The six processors **111**, **113**, **121**, **123**, **125**, and **127** operate in lock step (i.e., are processing code sequences). For example, the processor **111** operates in lock step with the processor **113**, and the processor **121** operates in lock step with the processor **123** and the processor **125** operates in lock step with the processor **127**.

The processor **125** may be designated as a "hot standby," and is sitting idle in lock step mode with the processor **127**. Should one of the processors **111**, **113**, **121**, and **123** suffer an error, the hot standby processors **125**, **127** may be used to speed recovery from the resulting loss of lock step.

FIG. **4** is a flow chart illustrating a process **200** for recovery from a loss of lock step using the computer system **100** shown in FIG. **3**. The process **200** will be shown with an error condition in the first processor pair **111/113**. The operation **200** begins in block **205** with the system **100** operating in a normal lock step fashion. In block **210**, the processor **111** detects an error event that indicates an impending loss of lock step. In block **215**, the processor **111** signals the node controller **130** that the first processor pair **111/113** has broken lock step and that the first processor pair **111/113** should be taken "off-line." In block **220**, the node controller **130** copies the architected state of the first processor pair **111/113** to the hot standby processor pair **125/127**. In an embodiment, the architected state of the first processor pair **111/113** may be stored in the node controller **130**, and to facilitate recovery, the node controller **130** copies the stored state to the third processor pair **125/127**. Alternatively, the node controller **130** may copy the state of the first processor pair **111/113** directly from the processors **111** and **113** to the processors **125** and **127** without any intermediate storage of the architected state in the node controller **130**, or other component of the computer system **100**. The processor pair **125/127** then becomes the logical CPU **0** in the computer system **100**, and the computer system **100** operates without a hot standby processor pair. In block **225**, recovery actions are executed on the first processor pair **111/113** (e.g., all caches are flushed on the processors **111** and **113**). In block **230**, the node controller **130** "reboots" the processors **111** and **113**, and the processors **111** and **113** become the new "hot standby" processor pair on the system **100**. In block **235**, the operation **200** ends, with the computer system **100** operating the processors **121**, **123**, **125**, and **127** in lock step, and with the processors **111/113** idle and in hot standby.

The terms and descriptions used herein are set forth by way of illustration only and are not meant as limitations. Those skilled in the art will recognize that many variations are possible within the spirit and scope of the invention as

defined in the following claims, and there equivalents, in which all terms are to be understood in their broadest possible sense unless otherwise indicated.

The invention claimed is:

1. A method for recovery from loss of lock step in a computer system, comprising:

receiving a loss of lock step signal from a processor unit;

receiving a notice from the loss of lock step processor unit to take the loss of lock step processor unit off line; and

moving an architected state of the loss of lock step processor unit to a spare processor unit, wherein the spare processor unit becomes an active processor unit in the computer system.

2. The method of claim **1**, further comprising correcting an error condition in the loss of lock step processor unit.

3. The method of claim **2**, further comprising rebooting the loss of lock step processor unit, whereby the rebooted processor unit becomes a new spare processor unit.

4. The method of claim **1**, wherein moving the architected state comprises copying the architected state from a node controller that couples the loss of lock step processor unit and the spare processor unit.

5. The method of claim **1**, wherein moving the architected state comprises copying the architected state directly from the loss of lock step processor unit to the spare processor unit.

6. The method of claim **1**, wherein moving the architected state comprises:

copying the architected state from the loss of lock step processor unit to a node controller; and

copying the architected state from the node controller to the idle processor unit.

7. An apparatus to recover from a loss of lock step in a multiprocessor computer system, wherein two or more processor units operate in lock step, comprising:

in each of the two or more processor units operating in lock step:

means for detecting a loss of lock step initiating event, and

means for signaling an impending loss of lock step; and

means for moving an architected state of a processor unit having the loss of lock step initiating event to a spare processor unit, the spare processor unit operating idle in lock step.

8. The apparatus of claim **7**, further comprising means for taking the loss of lock step processor unit off line.

9. The apparatus of claim **8**, further comprising means for rebooting the loss of lock step processor unit, whereby the rebooted processor unit is designated as a new spare processor unit.

10. The apparatus of claim **7**, wherein the moving means comprises:

means for copying the architected state; and

means for storing the architected state.

11. The apparatus of claim **10**, wherein the means for storing the architected state comprises a node controller coupled to the two or more processor units and the spare processor unit.

12. The apparatus of claim **7**, wherein the moving means comprises means for copying the architected state directly from the loss of lock step processor unit to the spare processor unit.

13. A method for recovering from a loss of lock step operation in a multi-processor computer system, comprising:

detecting a loss of lock step initiating event in a processor;

signaling an impending loss of lock step;

copying an architected state of the loss of lock step processor to a spare processor; and

idling the loss of lock step processor.

14. The method of claim **13**, further comprising:

correcting the loss of lock step event in the loss of lock step processor; and rebooting the loss of lock step processor, whereby the rebooted processor is a new spare processor.

* * * * *

Appendix A

This claim chart compares claims 1, 5-8, 13, 17, 21, 22, and 25-26 to U.S. Patent No. 7,328,371 to Kalyanasundharam et al. ("*Kalyanasundharam*") (Ex. 1005), in view of U.S. Patent No. 5,450,576 to Kennedy ("*Kennedy*") (Ex. 1006). Kalyanasundharam qualifies as prior art under pre-AIA § U.S.C. 102(e) because it was filed in the United States on October 15, 2004, before the '302 patent was filed. Kennedy qualifies as prior under pre-AIA § U.S.C. 102(b) because it was filed in the United States on April 26, 1993, and was issued as a patent on September 12, 1995, over a year before the '302 patent was filed.

The table below reflects SNQ1 presented in the corresponding Request for Reexamination, which includes the pertinent motivations to combine the relevant art. Further, corroborated evidence reflecting the perspective knowledge of a person having ordinary skill in the art at the time of the invention of the '302 Patent (a "POSITA") is provided through the declaration of Andrew Wolfe ("Wolfe Decl."), (Ex. 1003).

References
Kalyanasundharam (Ex. 1005)
Kennedy (Ex. 1006)

Claim Language	Relevant Prior Art Disclosures and Argument
[1P] A method comprising:	Kalyanasundharam and Kennedy teach a method of processor failover by disclosing: "FIG. 6 is a flowchart illustrating operation of one embodiment of the node controller including the redundancy check circuit shown in FIG. 5 in response to receiving a communication from a processor core." *Kalyanasundharam* (Ex. 1005) at 2:7-10.
[1A] detecting loss of lockstep (LOL) for a processor in a multi-processor system;	Kalyanasundharam and Kennedy teach detecting loss of lockstep (LOL) for a processor in a multi-processor system by disclosing an error detecting mechanism that checks for errors within processors 18A-18B that execute in lockstep. "In embodiments in which **the processor cores 18A-18B execute in lockstep**, the redundancy check circuit 26A may check that corresponding communications are concurrently received from the processor cores 18A-18B (e.g. received on the same clock cycle). For example, if a valid communication is received from one of the processor cores 18A-18B and a valid communication is not concurrently received from the other processor core 18A-18B, **an execution error may be detected**. *Id.* at 4:27-35).

[1B] determining that the processor for which said LOL is detected is assigned a role of boot processor;	Kalyanasundharam and Kennedy teach determining whether a processor module is assigned a role of system boot processor by disclosing "provision needs to be made to determine which processor will boot the system, and if this "default" boot processor fails to boot the system, an alternative processor, if one is installed, should take over boot operations to enhance the availability of the system." *Kennedy* (Ex. 1006) at 7:5-13. Kalyanasundharam describes use of multiple pairs of processors that operate in lockstep: The processor cores may be configured as N/2 pairs of redundant processor cores. Each pair may redundantly execute different software code from other pairs. In some embodiments, the processor cores may be preassigned into pairs for redundant execution purposes. *Kalyanasundharam* (Ex. 1005) at 13:60-64. Kalyanasundharam further notes: "If the masked comparison results indicate that the communication from the primary processor core mismatches with all auxiliary processor cores (that are enabled) (decision block 100, "yes" leg), the control circuit 84 may **disable the primary processor core and select one of the auxiliary processor cores to be the primary processor core** (block 102)." *Kalyanasundharam* (Ex. 1005) at 13:16-22. While Kalyanasundharam determines, upon detecting a loss of lockstep, whether a processor is a primary processor core, Kalyanasundharam fails to specifically indicate that the processor is a "boot processor." In analogous art, Kennedy shows a multiprocessor system that identifies default boot processors and alternative boot processors should the default boot processor fail to boot. *Kennedy* (Ex. 1006) at 7:5-13). This is directly analogous to being assigned the role of boot processor in the '302 Patent's system. "In the multiprocessor system 100 which utilizes a system bus 102, a number of problems can arise if more than one processor type is installed on the 64-bit bus 104. For instance, provision needs to be made to **determine which processor will boot the system**, and if this **"default" boot processor** fails to boot the system, **an alternative processor**, if one is installed, **should take over boot operations** to enhance the availability of the system." *Id.* It would have been obvious to combine Kalyanasundharam with Kennedy to expand Kalyanasundharam's ability to fail over to many different processors including, specifically, boot processors, as noted by Kennedy. Because Kalyanasundharam and Kennedy are both high-reliability computing systems, a POSITA would reasonably expect to succeed when combining the boot processor of Kennedy to the primary and secondary processor cores of Kalyanasundharam. *Wolfe Decl.* (Ex. 1003) ¶ 75.
[1Ci] changing a system identifier, used by the system's operating system	Kalyanasundharam and Kennedy teach changing a system identifier, used by the system's operating system to recognize the boot processor, to a different value by disclosing

to recognize the boot processor, of said boot processor for which said LOL is detected, to a different value	a control circuit that can "update the PC field of the configuration register 42 to reflect the newly selected primary processor core." *Kalyanasundharam* (Ex. 1005) at 13:22-29. Updating the PC field of a configuration register in Kalyanasundharam is equivalent (or at least would have been obvious) to changing the system identifier of claim 1 in the '302 patent because Kalyanasundharam's system uses the PC field to recognize which processor is the primary core and which is the secondary core and, as such, changes to the PC field will dictate how the system determines the identity of available processor cores. *Wolfe Decl.* (Ex. 1003) ¶ 76. Kalyanasundharam states: "As with the discussion above for disabling the auxiliary processor core, the control circuit 84 may simply **disable the primary processor core** from comparison **by clearing the corresponding enable bit in the configuration register 42**, or may disable the primary processor core from executing at all. The control circuit 84 may also **update the PC field of the configuration register 42** to reflect the newly selected primary processor core." *Kalyanasundharam* (Ex. 1005) at 13:22-29. To the extent that changing a primary processor core to a secondary processor core (or vice versa) by changing an enable bit in a configuration register is not interpreted as changing a system identifier of a boot processor, Kennedy states, "If the CPU1 120 in slot 1 does not boot (proper boot is indicated by a change in the level of the SLOT SELECT signal line 193 from high to low) before the timer 184 in slot 2 times-out, then the CPU2 132 exits the reset state and attempts to boot the system as an alternate boot CPU." *Kennedy* (Ex. 1006) at 15:9-14. "If the CPU1 120 does not complete boot operations within the time allotted, then the CPU2 132 **takes over default boot operations as an alternative boot processor instead of CPU1** 120. If the CPU1 120 does boot properly, then the CPU2 132 checks in as an additional processor rather than taking over as a boot processor." *Kennedy* (Ex. 1006) at 13:38-44).

[1Cii] which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor	Kalyanasundharam and Kennedy teach causing the system's operating system to not recognize the processor for which said LOL is detected as the boot processor by disclosing that if one CPU fails to complete boot operations, another CPU will be used by the operating system as boot processor. Furthermore, to the extent that Kalyanasundharam is not interpreted to include a boot processor, Kalyanasundharam as modified by Kennedy, would cause the system's operating system to not recognize the processor for which said LOL is detected as the boot processor by indicating that slot select signals used by the operating system for communication would be changed in the event of a failure to boot: "If the CPU1 120 does not complete boot operations within the time allotted, then the CPU2 132 **takes over default boot operations as an alternative boot processor instead of CPU1** 120. If the CPU1 120 does boot properly, then the CPU2 132 checks in as an additional processor rather than taking over as a boot processor." *Kennedy* (Ex. 1006) at 13:38-44.
[1Di] changing a system identifier of the spare processor to that of the boot processor,	Kalyanasundharam and Kennedy teach changing a system identifier of the spare processor to that of the boot processor by disclosing a control circuit that can "update the PC field of the configuration register 42 to reflect the newly selected primary processor core." *Kalyanasundharam* (Ex. 1005) at 13:22-29. Kalyanasundharam states: "As with the discussion above for disabling the auxiliary processor core, the control circuit 84 may simply **disable the primary processor core** from comparison **by clearing the corresponding enable bit in the configuration register 42**, or may disable the primary processor core from executing at all. The control circuit 84 may also **update the PC field of the configuration register 42** to reflect the newly selected primary processor core." *Id.* Moreover, to the extent that changing a primary processor core to a secondary processor core (or vice versa) by changing an enable bit in a configuration register is not interpreted as changing a system identifier of a boot processor, Kennedy states, "provision needs to be made to **determine which processor will boot the system, and if this 'default' boot processor fails** to boot the system**, an alternative processor,** if one is installed, **should take over boot operations** to enhance the availability of the system." *Kennedy* (Ex. 1006) at 7:5-13. Thus, as modified by Kennedy, Kalyanasundharam's system would have used bits in the configuration register to indicate that the new selected processor core would be the boot processor.

[1Dii] thereby switching the role of boot processor to a spare processor without shutting down the system's operating system.	Kalyanasundharam and Kennedy teach switching the role of boot processor to a spare processor without shutting down the system's operating system by disclosing a control circuit that can "update the PC field of the configuration register 42 to reflect the newly selected primary processor core." *Kalyanasundharam* (Ex. 1005) at 13:22-29. Since the configuration register is part of a redundancy checking circuit, changes to the configuration register would not require an operating system shutdown to implement. *Wolfe Decl.* (Ex. 1003) ¶ 80. To the extent that Kalyanasundharam is not interpreted to include a boot processor, Kalyanasundharam as modified by Kennedy, would cause the system's operating system to switch the role of boot processor to a spare processor: "If the CPU1 120 does not complete boot operations within the time allotted, then the CPU2 132 **takes over default boot operations as an alternative boot processor instead of CPU1** 120. If the CPU1 120 does boot properly, then the CPU2 132 checks in as an additional processor rather than taking over as a boot processor." *Kennedy* (Ex. 1006) at 13:38-44). Thus, as modified by Kennedy, Kalyanasundharam's system would have used bits in the configuration register to indicate that the new selected processor core would be the boot processor.
[5] wherein the spare processor is a processor module that includes a master and slave processors that operate in lockstep.	Kalyanasundharam teaches that the spare processor is a processor module that includes a master and slave processors that operate in lockstep by disclosing multiple processor core pairs, where one pair of processor cores is a primary core and another pair of processor cores is an auxiliary processor core that, together, are equivalent to master and slave processors. > The processor cores may be configured as N/2 pairs of redundant processor cores. Each pair may redundantly execute different software code from other pairs. In some embodiments, the processor cores may be preassigned into pairs for redundant execution purposes. *Kalyanasundharam* (Ex. 1005) at 13:60-64. Generally, one of the processor cores 18A-18B may be designated as a primary processor core and the other processor core 18A-18B is referred to as the auxiliary processor core. The primary processor core may be the processor core whose execution results will be used as the result of executing the redundantly-executed software code. The auxiliary processor core is used to verify correct execution by the primary processor core." *Kalyanasundharam* (Ex. 1005) at 4:58-65.

[6] wherein said determining that the processor for which said LOL is detected is assigned the role of boot processor comprises accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor.	To the extent that Kalyanasundharam is not interpreted to disclose accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor, Kalyanasundharam as modified by Kennedy teaches accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor. Kennedy states: "The CPU1 120 has a corresponding **non-volatile memory 123** which contains **configuration information and CPU specific initialize and self test code** (ISTC) for the CPU1 120..." *Kennedy* (Ex. 1006) at 5:15-18). It would have been obvious to a POSITA to combine Kalyanasundharam with Kennedy to increase the reliability of the system by storing and accessing CPU-specific information from non-volatile memory, as taught by Kennedy. *Id.*, *see also Wolfe Decl.* (Ex. 1003) ¶ 82.
[7] further comprising: determining the processor that is assigned the role of spare processor.	Kalyanasundharam as modified by Kennedy teaches determining the processor that is assigned the role of spare processor by disclosing: "...provision needs to be made to **determine which processor will boot the system**, and if this **"default" boot processor fails** to boot the system, **an alternative processor**, if one is installed, **should take over boot operations** to enhance the availability of the system." *Kennedy* (Ex. 1006) at 7:5-13). It would have been obvious to a POSITA to combine Kalyanasundharam with Kennedy to increase the reliability of the system by identifying which processors are assigned certain roles, as taught by Kennedy. *Id.*, *see also Wolfe Decl.* (Ex. 1003) ¶ 83.
[8] wherein said determining the processor that is assigned the role of spare processor comprises: accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of spare processor.	Kalyanasundharam as modified by Kennedy teaches accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of spare processor by disclosing an alternative processor that takes over if the default boot processor fails. Kennedy states: "...provision needs to be made to **determine which processor will boot the system**, and if this **"default" boot processor fails** to boot the system, **an alternative processor**, if one is installed, **should take over boot operations** to enhance the availability of the system." *Id.* at 7:5-13). It would have been obvious to a POSITA to combine Kalyanasundharam with Kennedy to avoid system errors by identifying which processors are assigned certain roles, as taught by Kennedy. *Id.*, *see also Wolfe Decl.* (Ex. 1003) ¶ 84.
[13P] A method comprising:	*See* Limitation [1P].
[13A] system firmware determining that loss of lockstep (LOL) is detected for a lockstep pair of processors that	*See* Limitation [1A].

are assigned the role of boot processor in a system;	
[13B] determining one of said lockstep pair of processors that is not the cause of said LOL;	*See* Limitation [1B].
[13C] copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to a spare processor;	Kalyanasundharam and Kennedy teach copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to a spare processor by disclosing that state information for processor communications is stored. Kalyanasundharam states that "[a]dditionally, the SRO control unit 32 may be configured to update state in the SRO 30 for a given communication in response to completion of various internal activities. For example, if a probe is in the SRO 30 and the corresponding responses from the processors 18A-18B are received for the probe, the SRO control unit 32 may update the state in the entry storing the probe to record the receipt of the response and to record the coherency State reported in the response." *Kalyanasundharam* (Ex. 1005) at 7:34-43.
[13Di] changing a system identifier, used by the system's operating system to recognize the boot processor, of said lockstep pair of processors that are assigned the role of boot processor, to a different value,	*See* Limitation [1Ci].
[13Dii] which causes the s system's operating system to not recognize the lock step pair of processors for which said LOL is detected as the processor; and	*See* Limitation [1Cii].
[13E] changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to said spare processor.	*See* Limitations [1Di] and [1Dii].
[17] wherein said switching causes the system's operating system to	*See* Limitation [1Dii].

recognize said spare processor as said boot processor without shutting down said operating system.	
[21P] A system compristing:	Kalyanasundharam & Kennedy disclose a system 10 of FIG. 1 that includes nodes 12A-12B, memories 14A-14B, and peripheral devices 16A-16B" *Kalyanasundharam* at 2:26-29.
[21A] a plurality of processor modules that each include a master processor and a slave processor that operate in lockstep;	Kalyanasundharam and Kennedy teach a plurality of processor modules that each include a master processor and a slave processor that operate in lockstep by disclosing primary and auxiliary processors that operate in lockstep. "In one embodiment, **the processor cores 18A-18B may execute in lockstep** (that is, may execute the same instructions on the same clock cycle)." *Kalyanasundharam* (Ex. 1005) at 4:15-18. "Generally, one of the processor cores 18A-18B may be designated as a primary processor core and the other processor core 18A-18B is referred to as the auxiliary processor core. The primary processor core may be the processor core whose execution results will be used as the result of executing the redundantly-executed software code. **The auxiliary processor core is used to verify correct execution by the primary processor core**." *Id.* at 4:58-65).
[21B] an operating system;	"Generally, the processor cores 18A-18B may use the 15 interface(s) to the node controller 20A to communicate with other components of the computer system 10 (e.g., peripheral devices 16A-16B, processor cores 18C-18D, the memory controllers 22A-22B, etc.)." *Id.* at 3:15-18.
[21C] error detection logic operable to detect loss of lockstep (LOL) for at least one of said processor modules; and	Kalyanasundharam and Kennedy teach error detection logic operable to detect loss of lockstep (LOL) by disclosing an error detecting mechanism that checks for errors within processors 18A-18B that execute in lockstep. "In embodiments in which **the processor cores 18A-18B execute in lockstep**, the redundancy check circuit 26A may check that corresponding communications are concurrently received from the processor cores 18A-18B (e.g., received on the same clock cycle)." For example, if a valid communication is received from one of the processor cores 18A-18B and a valid communication is not concurrently received from the other processor core 18A-18B, **an execution error may be detected**. *Kalyanasundharam* (Ex. 1005) at 4:27-35.

[21Di] system firmware operable, responsive to the detection of LOL for a first of said processor modules, to determine whether said first processor module is assigned a role of system boot processor;	Kalyanasundharam and Kennedy teach system firmware operable, responsive to the detection of LOL, to determine whether a processor module is assigned a role of system boot processor by disclosing "provision needs to be made to determine which processor will boot the system, and if this "default" boot processor fails to boot the system, an alternative processor, if one is installed, should take over boot operations to enhance the availability of the system." *Kennedy* (Ex. 1006) at 7:5-13. Kalyanasundharam notes: "If the masked comparison results indicate that the communication from the primary processor core mismatches with all auxiliary processor cores (that are enabled) (decision block 100, "yes" leg), the control circuit 84 may **disable the primary processor core and select one of the auxiliary processor cores to be the primary processor core** (block 102)." *Kalyanasundharam* (Ex. 1005) at 13:16-22. While Kalyanasundharam determines, upon detecting a loss of lockstep, whether a processor is a primary processor core, Kalyanasundharam fails to specifically indicate that the processor is a "boot processor." In analogous art, Kennedy shows a multiprocessor system that identifies default boot processors and alternative boot processors should the default boot processor fail to boot. This is directly analogous to being assigned the role of boot processor in the '302 Patent's system. "In the multiprocessor system 100 which utilizes a system bus 102, a number of problems can arise if more than one processor type is installed on the 64-bit bus 104. For instance, provision needs to be made to **determine which processor will boot the system**, and if this **"default" boot processor** fails to boot the system, **an alternative processor**, if one is installed, **should take over boot operations** to enhance the availability of the system." *Kennedy* (Ex. 1006) at 7:5-13. It would have been to combine Kalyanasundharam with Kennedy to expand Kalyanasundharam's ability to fail over to many different processors including, specifically, boot processors, as noted by Kennedy. Because Kalyanasundharam and Kennedy are both high-reliability computing systems, a POSITA would reasonably expect to succeed when combining the boot processor of Kennedy to the primary and secondary processor cores of Kalyanasundharam. *Wolfe Decl.* (Ex. 1003) ¶ 94.

[21Dii] wherein if determined that said first processor module is assigned the role of system boot processor, said system firmware is further operable to cause said operating system to recognize another processor module as system boot processor without shutting down said operating system.	Kalyanasundharam and Kennedy teach system firmware that causes the operating system to recognize another processor module as system boot processor without shutting down the operating system by disclosing a control circuit that can "update the PC field of the configuration register 42 to reflect the newly selected primary processor core." *Kalyanasundharam* (Ex. 1005) at 13:22-29. Kalyanasundharam states: "As with the discussion above for disabling the auxiliary processor core, the control circuit 84 may simply **disable the primary processor core** from comparison **by clearing the corresponding enable bit in the configuration register 42**, or may disable the primary processor core from executing at all. The control circuit 84 may also **update the PC field of the configuration register 42** to reflect the newly selected primary processor core." *Id.*
[22] wherein the another processor module is a processor module assigned a role of spare processor.	Kalyanasundharam and Kennedy teach that the other processor module is a processor module assigned a role of spare processor by disclosing an alternative processor that takes over if the default boot processor fails. Kennedy states: "…provision needs to be made to **determine which processor will boot the system**, and if this **"default" boot processor fails** to boot the system, **an alternative processor**, if one is installed, **should take over boot operations** to enhance the availability of the system." *Id.* at 7:5-13). It would have been obvious to a POSITA to combine Kalyanasundharam with Kennedy to increase reduce system errors by assigning processor roles, as taught by Kennedy. *Id.*, *see also Wolfe Decl.* (Ex. 1003) ¶ 96.
[25] further comprising: non-volatile data storage storing information that assigns the role of system boot processor to one of said plurality of processor modules.	*See* Limitation [8].
[26P] A system comprising:	*See* Limitation [21P].
[26A] system firmware means for detecting loss of lockstep (LOL) for a processor in a multi-processor system;	*See* Limitation [21C].
[26B] means for determining whether the processor for which said LOL is detected is assigned the role of boot processor; and	*See* Limitation [21Di].

[**26C**] system firmware means for switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which said LOL is detected is assigned the role of boot processor.	*See* Limitation [21Dii].

PTO/SB/57 (01-18)
Approved for use through 11/30/2024. OMB 0651-0064
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

(Also referred to as FORM PTO-1465)

REQUEST FOR *EX PARTE* REEXAMINATION TRANSMITTAL FORM

Address to:
Mail Stop *Ex Parte* Reexam
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Attorney Docket No.: 185945.012600

Date: 9/1/23

1. ☒ This is a request for *ex parte* reexamination pursuant to 37 CFR 1.510 of patent number 7,624,302 issued November 24, 2009. The request is made by:
   ☐ patent owner. ☒ third party requester.

2. ☒ The name and address of the person requesting reexamination is:
   Unified Patents, LLC
   4445 Willard Ave., Suite 600
   Chevy Chase, MD 20815

3. Requester ☐ asserts small entity status (37 CFR 1.27) or ☐ certifies micro entity status (37 CFR 1.29). Only a patent owner requester can certify micro entity status. Form PTO/SB/15A or B must be attached to certify micro entity status.

4. This request is accompanied by payment of the reexamination fee as set forth in:
   ☒ 37 CFR 1.20(c)(2); **or**
   ☐ 37 CFR 1.20(c)(1). **In checking this box for payment of the fee set forth in 37 CFR 1.20(c)(1), requester asserts that this request has forty (40) or fewer pages and complies with all other requirements of 37 CFR 1.20(c)(1).**

   Payment of the reexamination fee is made by the method set forth below.
   a. ☐ A check in the amount of $_____ is enclosed to cover the reexamination fee;
   b. ☒ The Director is hereby authorized to charge the reexamination fee to Deposit Account No. 50-6990;
   c. ☐ Payment by credit card. Form PTO-2038 is attached; **or**
   d. ☐ Payment made via EFS-Web.

   ☒ In addition, the Director is hereby authorized to charge any fee deficiencies to Deposit Account No. 50-6990.

5. ☒ Any refund should be made by ☐ check or ☐ credit to Deposit Account No. 50-6990. 37 CFR 1.26(c). If payment is made by credit card, refund must be to credit card account.

6. ☒ A copy of the patent to be reexamined having a double column format on one side of a separate paper is enclosed. 37 CFR 1.510(b)(4).

7. ☐ CD-ROM or CD-R in duplicate, Computer Program (Appendix) or large table
   ☐ Landscape Table on CD

A Federal agency may not conduct or sponsor, and a person is not required to respond to, nor shall a person be subject to a penalty for failure to comply with an information collection subject to the requirements of the Paperwork Reduction Act of 1995, unless the information collection has a currently valid OMB Control Number. The OMB Control Number for this information collection is 0651-0064. Public burden for this form is estimated to average 18 minutes per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the information collection. Send comments regarding this burden estimate or any other aspect of this information collection, including suggestions for reducing this burden to the Chief Administrative Officer, United States Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450 or email InformationCollection@uspto.gov. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **If filing this completed form by mail, send to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

8. ☐ Nucleotide and/or Amino Acid Sequence Submission
If applicable, items a. – c. are required.
   a. ☐ Computer Readable Form (CRF)
   b. Specification Sequence Listing on:
      i. ☐ CD-ROM (2 copies) or CD-R (2 copies) **or**
      ii. ☐ paper
   c. ☐ Statements verifying identity of above copies.

9. ☐ A copy of any disclaimer, certificate of correction or reexamination certificate issued in the patent is included.

10. ■ Reexamination of claim(s) 1-27 is requested.

11. ■ A copy of every patent or printed publication relied upon is submitted herewith including a listing thereof on Form PTO/SB/08, PTO-1449, or equivalent.

12. ■ An English language translation of all necessary and pertinent non-English language patents and/or printed publications is attached.

13. ■ The attached detailed request includes at least the following items:
   a. A statement identifying each substantial new question of patentability based on prior patents and printed publications. 37 CFR 1.510(b)(1).
   b. An identification of every claim for which reexamination is requested, and a detailed explanation of the pertinency and manner of applying the cited art to every claim for which reexamination is requested. 37 CFR1.510(b)(2).

14. ☐ A proposed amendment is included (only where the patent owner is the requester). 37 CFR 1.510(e).

15. ■ It is certified that the statutory estoppel provisions of 35 U.S.C. 315(e)(1) or 35 U.S.C. 325(e)(1) do not prohibit requester from filing this *ex parte* reexamination request. 37 CFR 1.510(b)(6).

16. Service
   a. ■ It is certified that a copy of this request (if filed by other than the patent owner) has been served in its entirety on the patent owner as provided in 37 CFR 1.33(c).

      The name and address of the party served are:

      Foras Technologies Ltd. c/o 56436 - Hewlett Packard Enterprise
      3404 E. Harmony Road, Mail Stop 79
      Fort Collins, CO 80528

      Date of Service: 9/1/23

   OR

   b. ☐ A duplicate copy is enclosed since service on patent owner was not possible. An explanation of the efforts made to serve patent owner **is attached**. See MPEP 2220.

17. Correspondence Address: Direct all communication about the reexamination to:

☒ The address associated with Customer Number: 67395

OR

☐ Firm or Individual Name _____
(at the address identified below)

Address

City	State	Zip

Country

Telephone	Email

18. ☒ The patent is currently the subject of the following concurrent proceeding(s):

a. ☐ Copending reissue Application No. _____

b. ☐ Copending reexamination Control No. _____

c. ☐ Copending Interference No. _____

d. ☒ Copending litigation styled:

Case No. 1:23-cv-00640 (W.D. Tex., June 6, 2023)

Case No. 2:23-cv-00150 (E.D. Tex., Apr. 5, 2023)

WARNING : Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

/Greg Lunt/ — Authorized Signature

9/1/23 — Date

Greg Lunt — Typed/Printed Name

57354 — Registration No.

☐ For Patent Owner Requester ☒ For Third Party Requester

Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. The United States Patent and Trademark Office (USPTO) collects the information in this record under authority of 35 U.S.C. 2. The USPTO's system of records is used to manage all applicant and owner information including name, citizenship, residence, post office address, and other information with respect to inventors and their legal representatives pertaining to the applicant's/owner's activities in connection with the invention for which a patent is sought or has been granted. The applicable Privacy Act System of Records Notice for the information collected in this form is COMMERCE/PAT-TM-7 Patent Application Files, available in the Federal Register at 78 FR 19243 (March 29, 2013). https://www.govinfo.gov/content/pkg/FR-2013-03-29/pdf/2013-07341.pdf

Routine uses of the information in this record may include disclosure to: 1) law enforcement, in the event that the system of records indicates a violation or potential violation of law; 2) a Federal, state, local, or international agency, in response to its request; 3) a contractor of the USPTO having need for the information in order to perform a contract; 4) the Department of Justice for determination of whether the Freedom of Information Act (FOIA) requires disclosure of the record; 5) a Member of Congress submitting a request involving an individual to whom the record pertains, when the individual has requested the Member's assistance with respect to the subject matter of the record; 6) a court, magistrate, or administrative tribunal, in the course of presenting evidence, including disclosures to opposing counsel in the course of settlement negotiations; 7) the Administrator, General Services Administration (GSA), or their designee, during an inspection of records conducted by GSA under authority of 44 U.S.C. 2904 and 2906, in accordance with the GSA regulations and any other relevant (i.e., GSA or Commerce) directive, where such disclosure shall not be used to make determinations about individuals; 8) another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)); 9) the Office of Personnel Management (OPM) for personnel research purposes; and 9) the Office of Management and Budget (OMB) for legislative coordination and clearance.

If you do not furnish the information requested on this form, the USPTO may not be able to process and/or examine your submission, which may result in termination of proceedings, abandonment of the application, and/or expiration of the patent.

Additional Uses

Additional USPTO uses of the information in this record may include disclosure to: 1) the International Bureau of the World Intellectual Property Organization, if the record is related to an international application filed under the Patent Cooperation Treaty; 2) the public i) after publication of the application pursuant to 35 U.S.C. 122(b), ii) after issuance of a patent pursuant to 35 U.S.C. 151, iii) if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspections, or an issued patent, or iv) without publication of the application or patent under the specific circumstances provided for by 37 CFR 1.14(a)(1)(v)-(vii); and/or 3) the National Archives and Records Administration, for inspection of records.

Commercial Fault Tolerance: A Tale of Two Systems

Wendy Bartlett, *Member, IEEE Computer Society*, and Lisa Spainhower, *Member, IEEE*

Abstract—This paper compares and contrasts the design philosophies and implementations of two computer system families: the IBM S/360 and its evolution to the current zSeries line, and the Tandem (now HP) NonStop® Server. Both systems have a long history; the initial IBM S/360 machines were shipped in 1964, and the Tandem NonStop System was first shipped in 1976. They were aimed at similar markets, what would today be called enterprise-class applications. The requirement for the original S/360 line was for very high availability; the requirement for the NonStop platform was for single fault tolerance against unplanned outages. Since their initial shipments, availability expectations for both platforms have continued to rise and the system designers and developers have been challenged to keep up. There were and still are many similarities in the design philosophies of the two lines, including the use of redundant components and extensive error checking. The primary difference is that the S/360-zSeries focus has been on localized retry and restore to keep processors functioning as long as possible, while the NonStop developers have based systems on a loosely coupled multiprocessor design that supports a "fail-fast" philosophy implemented through a combination of hardware and software, with workload being actively taken over by another resource when one fails.

Index Terms—Computer systems implementation, fault tolerance, high availability.

1 Introduction

Since the 1960s, there have been a number of commercial systems introduced with a specific focus on high availability or fault tolerance. This paper describes the design points and evolution of two of them—IBM's S/360 and its evolution to the zSeries, and Tandem's (now HP's) NonStop systems. Other well-known systems in this space include Stratus and Marathon.

For enterprise customers, definitions of what constitutes "availability" vary and may factor in not only the ability to provide a service but also acceptable response time and whether the results are accurate and consistent. Application availability requirements also vary tremendously, including different applications for the same customer. The need for availability may be "7 x 24 x forever" (e.g., ATM/POS applications and manufacturing floors for three-shift operations), or it may be for 100 percent availability within given timeframes (e.g., stock markets), or it may allow for infrequent, short disruptions (many applications). A framework for describing levels of availability is shown in Fig. 1.

Both system families described in this paper were originally targeted toward the High Availability/Fault Tolerant quadrant and since then have evolved toward Continuous Availability. While the goal has been the same in both cases, their paths toward achieving it have been different. The details appear in later sections, but Table 1 gives a high-level comparison of their approaches.

- *W. Bartlett is with Hewlett Packard, 19333 Vallco Parkway MS 4421 Cupertino, CA 95014. E-mail: wendy.bartlett@hp.com.*
- *L. Spainhower is with IBM, 2455 South Road, Mail Station P314, Poughkeepsie, NY 12601. E-mail: lisa@us.ibm.com.*

Manuscript received 10 June 2004; accepted 20 Aug. 2004.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSC-0084-0604.

1545-5971/04/$20.00 © 2004 IEEE Published by the IEEE Computer Society

2 Initial Target Audiences/Applications for the Systems

Both systems serve commercial business audiences for applications that require very high to continuous availability. From the beginning, IBM mainframes were designed with general-purpose business processing in mind. In the mid-1960s, most real-time business processes were manual and System/360 mainframes were used primarily for background jobs and overnight batch processing. The primary availability goals were to detect failures quickly enough to preserve data integrity and to locate and repair those failures quickly enough to not interfere with getting the tasks done by their deadlines. Over time, mainframes were used for more and more of the real-time work, although it tended to be local with well-defined hours of usage during the day and days during the week. This increased the requirements to lessen the frequency and impact of failures. As businesses became more global and moved to 24 x 7 operations, the demand for continuous operation became common. Limiting planned downtime required for change and upgrade became an increasingly important goal. IBM mainframes responded to each of these evolutionary needs for business computing.

The original Tandem NonStop systems were targeted at a few specific applications that had very high availability requirements, including back-end support for Automated Teller Machines (ATMs) or Point Of Sale (POS) devices and reliable message switching for applications such as credit card authorization. The expectation at the time for ATMs was that the infrastructure might be taken down for service at designated times (typically, 2:00-6:00 a.m. on Sundays) but otherwise was expected to supply 100 percent overall availability, so the emphasis was on fault tolerance rather than continuous operations. Individual machines might fail, but whole sites should not be

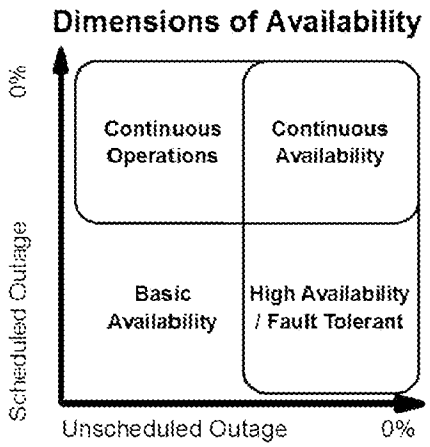


Fig. 1. Availability dimensions.

unavailable. Typically, a single site such as a bank branch would have two ATMs, connected via two different networks to two different systems, or at least to two different processors in the same system. NonStop servers soon moved into other financial markets such as stock exchanges and then into the telecommunications arena—not as switch replacements but for high-availability off-switch functions such as collection of call data records and billing. Use in other markets, including manufacturing shop floors, emergency 911 services, retail, and health care, followed shortly thereafter.

3 Initial Fault Tolerance Philosophies

3.1 zSeries

There was partial, but not complete, overlap in the original design principles for the two systems. Both focused on providing fault tolerance through duplicate components and paths. They also emphasized rapid error detection, as data integrity was and is a key requirement in their markets.

Three of the System/360 founding concepts are specific to dependability:

1. "It must be possible and straightforward to assemble systems with redundant I/O, storages, and CPUs so that the system can operate when modules fail.
2. As CPUs become increasingly reliable, built-in thorough checking against hardware malfunction is imperative for all systems, regardless of application.
3. Since the largest servicing problem is diagnosis of malfunction, built-in hardware fault-locating aids are essential to reduce downtimes. Furthermore, identification of individual malfunctions and of individual invalidities in program syntax would have to be provided" [1].

These concepts have come to be embodied as three prioritized principles on how to react to faults:

1. Ensure system integrity.
2. Provide uninterrupted applications.
3. Repair without disruption.

TABLE 1
System Comparison

	IBM zSeries	HP (Tandem) NonStop
Scope	General purpose server (Supports multiple OSs: zOS , VM, TPF, Linux) Integrated zOS Cluster (Parallel Sysplex)	Integrated system (Server, disk, comm, OS, DB, Txn monitor) Specific classes of target applications Loosely-coupled cluster of systems (ServerNet Clusters)
Fault Model	Two types: transient, permanent	Two types: recoverable, nonrecoverable
FT Operation	Explicit redundancy in support subsystems; implicit redundancy in computational subsystems Parallel Sysplex: no SPOF, distributed applications, data sharing	Multiple hardware components and paths Process pairs, mainly at lower levels of the system Persistent processes plus transactions at application level ServerNet Clusters: Distributed applications
Recovery Techniques	Begin with local retry Recover from transient Spare cache, memory, CPUs Spare or degrade from permanent Parallel Sysplex: Transaction-level restart	Takeover by backup processes in case of processor failure caused by either hardware or software Restart of persistent processes Transaction backout Switch paths
Data Integrity	Redundant CPU logic, Inline checking in I/O subsystem, ECC	Lockstepped microprocessors, ECC, end-to-end disk checksums, CRC

The zSeries, like its System/360-390 standalone mainframe predecessors, is designed to maximize hardware resource utilization during normal, fault-free operation, retry and recover locally when failures are detected, and continue operation without interrupting the ongoing applications. System packaging facilitates online maintenance of hardware components. Firmware and system software are designed to allow updates concurrent with normal operation in most cases. For well-tested software, clusters of zSeries mainframes, Parallel Sysplex, can be configured for no single point of failure in hardware or software. Maintenance, for repair or upgrade, can be performed online as far as the application is concerned.

Ensuring system integrity is paramount, even if application disruption is required in order to preserve it. These design principles apply equally well to the NonStop line, but its implementation philosophy is somewhat different.

3.2 NonStop Server

NonStop systems always have been designed to keep properly designed applications running after a single failure of any kind, to support online repair and reintegration of hardware components, and to preserve data integrity by preventing silent data corruption. For hardware, these goals have been met with a loosely coupled design based on multiple modules with multiple interconnections among them. The modularity helps to delineate fault containment zones and facilitate online repair. Individual modules are designed for ease of maintenance, in order to minimize repair time and reduce opportunities for service-induced failures. Fault detection is built-in to provide "fail-fast" capability to detect errors and isolate the offending hardware from the rest of the system. In contrast to most architectures, the software is an equal partner to the hardware in implementing fault tolerance and preservation of data integrity. The operating system provides its own layer of fault detection and fail-fast response, and supplies the infrastructure that allows applications to survive single processor failures. For a more detailed overview of the NonStop fault tolerance philosophy, see [2].

4 Design Principles

Although they use different system recovery designs, both HP and IBM mainframes rely on near-instantaneous error detection. Whether the aim is to fail fast and rapidly switchover or to transparently retry and recover locally, it is imperative that the scope of any faulty operation be strictly contained.

4.1 zSeries

There has been a steady advancement in the fault tolerant characteristics of IBM mainframes in the 40 years since System/360 was introduced [3]. As previously mentioned, the goal was and is to meet the requirements of general purpose business processing. System/360 was designed with robust error checking and was the first commercial computer to include Error Correction Code (ECC) in memory. Early system requirements were to reduce downtime by improving diagnosis. In the 1960s, computer downtime could be easily masked from customers and clients. Much of the computer workload was overnight batch and businesses depended on manual fall-back methods to continue processing bills, invoices, and service requests during downtime.

In the 1970s, designers improved mainframe fault tolerance with recovery from transient failures and increased the ability to mask hard failures with graceful degradation. For a specialized class of applications, S/370 Extended Recovery Facility was developed to permit one mainframe to monitor another and to assume its workload upon detection of a failure.

In the 1980s and early 1990s, as more and more business applications, particularly online transaction processing (OLTP), started to run 24 x 7, continuous availability grew as a mainframe requirement. To meet the demand, one major change was that online repair capabilities, formerly found only on multiprocessors, were greatly enhanced for all IBM mainframes. Initially, support subsystems—power, cooling, service processor—were either duplicated or designed to be noncritical to the ongoing processing of the computer. Later, those functional elements replicated for higher performance, like CPUs and I/O channels, were designed so that an operational element could take over in a transparent fashion for one that had failed. Increasingly, redundant elements were packaged so that they could be removed, replaced, and restarted without interrupting processing. Another type of online repair, electronic sparing, was also introduced. Mean time to application crash increased to 10 years with these enhancements.

The mid-1990s System/390 semiconductor technology transition from Emitter Coupled Logic (ECL) to Complementary Metal Oxide Semiconductor (CMOS) began a new evolution of fault tolerant design [4]. Fault tolerant enhancements were added with each new generation of CMOS, and today's zSeries has the most extensive detection and recovery in the mainframe history, as well as mean time to hardware-caused application crash in excess of 30 years.

Today, the zSeries memory hierarchy uses store-through (write-through) cache design, spare elements, and Error Correction Codes (ECC) for data redundancy [5]. Pending instruction, results are kept in both a store-through microprocessor cache (L1) and an ECC-protected store buffer and completed results are immediately stored into L2. L1 data is thus always replicated, allowing byte parity to be sufficient within the cache. A transient L1 failure is recovered by CPU instruction retry. For a permanent failure, depending on the scope, a cache line or quarter-cache delete is performed dynamically. A deleted line may be restored with a spare line at the next power on. Shared Level 2 (L2) cache is protected by ECC. Permanent faults in L2 that might result in an uncorrectable data error can be avoided by using a cache delete capability. Faulty locations either in the data array or in the address directory can be dynamically marked as invalid and a spare line can be substituted for the failed one. Each main memory (L3) array chip contributes one bit to an ECC word, so a (72,64) SEC/DED code can protect the system from catastrophic failures due to any single array chip failure. Automatic online repair of faulty DRAMs is done using built-in spare

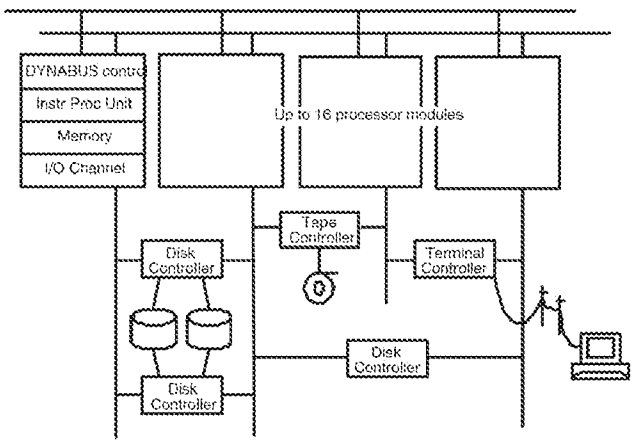


Fig. 2. Original tandem system architecture (1976).

chips. Counts of correctable errors are maintained on a per chip basis. When a threshold is exceeded, the data from the overthreshold chip is dynamically transferred to an error-free spare chip.

The I/O subsystem exploits the redundant paths between all devices and main memory to minimize the scope of any failures [6]. All paths are normally active and are used to enhance performance as well as to provide backup for one another if one should fail. The I/O channel adapters perform direct memory access with robust memory protection on behalf of I/O devices such as disk and tape storage, network communications, and server-to-server cluster (Parallel Sysplex) services. They prevent I/O devices from unauthorized memory read operations and from memory write operations into arbitrary memory locations. I/O channel adapters also provide error isolation by preventing the propagation of interface errors into the system. The bulk of the I/O subsystem hardware is packaged into various circuit cards that provide many different I/O interfaces and most of these cards can be concurrently replaced. Each I/O card position has an interface that controls the soft switch on the card. Before an I/O card is removed, the soft switch is deactivated causing the power on the card to drop. After a card is inserted, the soft switch is activated and the power to the card is turned on. To reduce human errors during concurrent card replacement, a group of LED indicators, one for each card position, is used to positively identify the card to be replaced.

To meet mainframe availability requirements, the power and cooling subsystem is designed with no single points of failure and concurrent repair capability of virtually all components.

4.2 NonStop Server

The Tandem NonStop system shown in Fig. 2, introduced in 1976, consisted of two to 16 independent processors connected by a pair of interprocessor buses collectively referred to as the Dynabus. Each processor had its own memory and ran its own copy of the operating system. Each processor also had an I/O bus. Each dual-ported I/O controller was connected to two processors' I/O buses, and had internal logic that selected which port was currently the primary path. If a processor or its I/O bus were to fail, the controllers whose primary paths were currently configured to use that I/O bus would switch ownership to their backup paths. Controller configuration was flexible enough that the workload of a failing processor could be spread over multiple surviving processors rather than all being taken over by a single processor.

The general design principle was that there be at least two of everything, including power supplies and fans as well as the more obvious processors, controllers, and peripherals. Dual-ported controllers and dual-ported peripherals provided four paths to each device. For disks, the use of host-based RAID-1 mirrored pairs of drives provided eight paths to the data and offered improved data integrity.

There were several special requirements on the hardware design. The first was that the Dynabus interfaces had to protect against a single processor failing in a way that disabled both buses or induced errors on both buses when power was removed from the processor module. The second was that power be managed in a way that allowed a power supply failure to be tolerated. The third was that hot insertion of boards into powered slots be supported, which enabled both online repair and online expansion. Battery backup, which preserved memory contents and supported power-fail restart, was standard [7]. Online repair was designed in for all hardware components, including power supplies and fans.

Beyond those requirements, there was a general need for extensive error checking using appropriate techniques such as parity, parity prediction, ECC, self-checking, and checksums. Controllers had their own fail-fast requirements to not corrupt data and, if possible, make available diagnostic information on the failure. End-to-end checksums were generated at the host and written to disk along with the data, with the checksum being recalculated and compared when the data was read back. The hardware was, in a sense, fault-intolerant—it was designed to fail quickly and cleanly at the first sign of a nonrecoverable error.

The system was architected to maximize useful, active modules and minimize the existence of "redundant," inactive modules, both for cost reasons and to prevent the accumulation of undetected, latent errors. This maximization was primarily the responsibility of the operating system software. As examples, processes were run in all processors—there were no "standby" processors; the message system used both paths of the Dynabus; and the disk process issued reads against both halves of the mirror (which also improved read performance).

Both the hardware and the software were optimized to support the message-based operating system design, including microcode-level support for sending and receiving messages. The choice of design yielded a system that is both highly distributed and fault tolerant. In addition to being easy to repair, the system could be easily expanded online.

Software fault tolerance was built into the operating system from the beginning [8], [9]. The system was structured as a set of processes communicating via interprocess messages, influenced by Dijkstra's "THE" system [10] and Brinch Hansen's implementation of a message-based operating system nucleus [11]. Each processor was

preconfigured with a small set of daemon-type processes that were responsible for specific tasks within that processor such as process control and memory management, along with a supporting set of library routines, interrupt handlers, etc. The key software abstraction, process pairs, allowed two communicating processes running in different processors to collectively represent a named resource such as a disk volume, communications line, or group of terminals, with one process functioning as the primary at any point in time and checkpointing necessary state to its backup. If a primary process or its processor were to fail, the backup process would take over and the operating system would transparently redirect requests to it without the application seeing an error. Operating system infrastructure was put in place to reliably maintain the named resource table that provides the redirection transparency.

Process pairs are an effective design for recovering from transient problems ("Heisenbugs" [12]) such as race conditions, where the identical conditions are unlikely to be in effect when the backup process takes over. They are vulnerable to deterministic defects ("Bohrbugs"), but those are relatively rare in a well-tested system. See [13] for an empirical study on this topic.

The initial focus of the system was to provide tolerance against single faults. Over time, the direction has been to add more support for continuous operations in order to serve the needs of customers who have been under increasing pressure to provide true continuous availability.

5 Advanced Design

5.1 Chip Technology

The designers of both system lines have had to continuously redesign their error checking logic in order to deal with both evolving technologies and the increasing rate of transient errors that is the inevitable side effect of the decreasing silicon feature size associated with newer, faster microprocessors and other components [14].

5.1.1 zSeries

From System/360 through System/390 ECL mainframes, increasingly complete and effective inline error checking was built into the circuitry. Within the CPU, parity and ECC were used extensively for all arrays, data paths, and control paths. Control logic, such as state machines and ALUs, used a variety of mechanisms such as parity prediction or illegal state detection.

For CMOS, a totally new design approach was needed [15]. The requirement for dynamic recovery, coupled with other design constraints (e.g., power, performance, second-level packaging) pointed to a highly checked single-chip microprocessor. However, inline checking of control and arithmetic logic was prohibited by performance penalties. As shown in Fig. 3, the decision was made to duplicate I-unit and E-unit and compare outputs. With a carefully laid out floorplan, there are no cross-chip performance-limiting paths. The cycle time is the same as if the same design were implemented on a smaller chip with only one unchecked I-unit and E-unit. The compare and detect cycle is

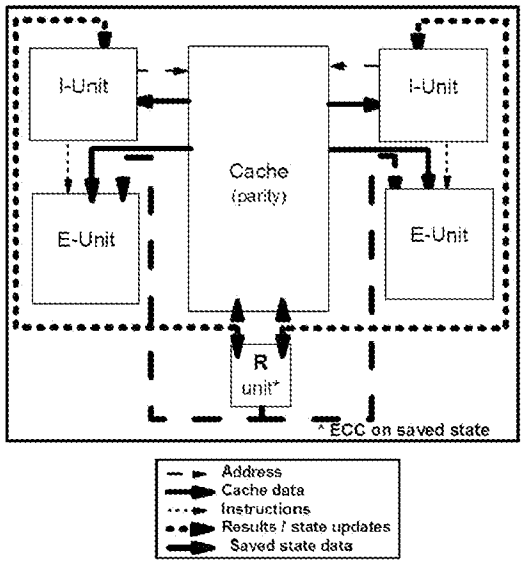


Fig. 3. Logical layout of the zSeries microprocessor chip.

completely overlapped in the instruction execution pipeline, so cycles per instruction (CPI) is not increased.

Most of the microprocessor area is devoted to the cache where data is primarily moved around unchanged. Updates to memory are maintained in an ECC-protected store buffer during instruction execution and transferred to an ECC-protected L2 cache when instruction execution completes. Thus, for the on-chip cache, low-overhead parity checking is sufficient.

A key element of the microprocessor's R-unit is an ECC-protected register file, the checkpoint array. The checkpoint array keeps track of the entire state of the CPU including register contents and instruction addresses. Completed instructions are known to be error-free and the checkpoint array and the store buffer accurately describe the state of the machine at successful instruction execution.

The R-unit permits recovery to be completely controlled by hardware and identical for all instructions. The R-unit manages instruction retry when an error is detected as follows: Except for the R-unit itself, the CPU is reset and store buffer contents are sent to L2. The state of the CPU is returned to what it was at the last instruction completion. Serialized instruction processing starts up, retrying the instruction that caused the error. If it is successful, the failure was transient and the CPU resumes pipelined instruction processing. If not successful, then the error is permanent and the CPU will be stopped. For permanent failures, the service element moves the checkpoint to another, typically spare, microprocessor and instruction execution resumes with no apparent interruption.

Total circuit overhead for cache parity, Register Unit ECC and duplicate I-unit and E-unit is about 30 percent. Design, verification, and test are greatly simplified compared to inline checking.

5.1.2 NonStop Server

The original NonStop system was designed in 1974-1975. The hardware logic was based on 7400 series TTL

integrated circuits, with the most complex being the 4-bit ALU. As gate density within an IC package increased and gate arrays became available, the designers were able to increase the self-checking logic. Because processor design was under Tandem's control, it was possible to build in as much self-checking logic as the developers deemed necessary, using techniques such as parity on buses and branch prediction. This approach continued to be used for more than 10 years. In the late 1980s, Tandem moved to ECL logic for its Cyclone processor [16]. By then, it was becoming evident that the economics of building its own processors from the ground up were no longer attractive, and the Cyclone was the last internally designed processor in the line.

In 1991, Tandem moved to building its processors around commercial microprocessors, choosing the MIPS 3000 for its initial CMOS system. Because the amount of error checking within the microprocessor was not within Tandem's control and was considered to be inadequate for its target markets, the design changed to a scheme based on tight lockstepping of a pair of MIPS chips. That approach has continued to be used up until now, and a variant is being developed for future Intel Itanium®-based platforms. The new approach enables use of a looser, but still effective, level of synchronization between microprocessors. This design allows individual microprocessors to temporarily diverge in their instruction streams, e.g., to perform transient error recovery, without their associated logical processors being declared down. It also supports a triplex configuration for customers who wish to retain processor-level hardware single fault tolerance and data protection during processor repair actions.

5.2 Other Changes

5.2.1 zSeries

The next step in the evolution of availability beyond a single operating system and mainframe, whether uniprocessor or SMP, was the tightly coupled multiprocessor. Such mainframes were comprised of two mainframes, which could be run as one operating system image or as two isolated images. When run as two, there was no common hardware, thus no single point of failure. A very common operational mode was to run a tightly coupled multiprocessor as one operating system image and, when needed for maintenance or when a failure occurred, degrade to a single mainframe. This is in keeping with the general design philosophy of delivering all available capacity during normal operation and only using capacity for availability in unusual conditions.

Logical partitioning, (LPAR), further enhanced the ability to design a single mainframe for higher availability. LPAR allows multiple operating system instances to concurrently reside on one mainframe. Each is isolated from the other, test and production can be run at the same time, and new releases run with old. With the advent of LPAR came Extended Recovery Facility (XRF), which allowed a backup partition to be created within the same or a different mainframe. The partition needed very few resources during normal operation as it monitored the primary operating partition. It did require a high priority such that, upon failure, it could take over the necessary resources to keep the primary workload running on the backup. XRF in one mainframe had no software single point of failure (SPOF) and running across two mainframes, no software or hardware SPOF.

A major new design paradigm came about with the introduction of Parallel Sysplex, a data-sharing cluster design [17]. Multiple mainframes (up to 32) can be connected together via a coupling facility, or for the most robust installations, two coupling facilities. All mainframes, or at the least, all mainframes in the Sysplex sharing data, have equal access to shared disks as well as to the coupling facilities which manage the sharing. The basic hardware structure of the Sysplex is therefore to have no SPOF in the hardware.

To get the same results in the software, it is necessary to spread identical workloads across each (or a subset) of the mainframes in the Sysplex, shown in Fig. 4. At first, the notion of putting all critical software on each of the mainframes is somewhat counter intuitive if a primary guideline has always been to isolate workloads from one another on different hardware. With Parallel Sysplex, by contrast, the ideal high availability configuration is one in which each mainframe is a clone of the others, from a software perspective. Transactions are distributed to the mainframes via a Workload Manager (WLM) that determines which mainframe is least busy for that particular type of work at the moment. In the event of a failure anywhere within the Sysplex, there is no formal switch-over—merely a cessation of assigning work to a no-longer-available application, operating system, or mainframe. The design makes it unnecessary to preassign backups and, during normal operation, all available capacity can be used by the ongoing workload.

For those customers without the stringent availability needs of Parallel Sysplex but for whom the software reliability of a standalone mainframe is insufficient, it is possible to use LPAR to construct a partition (or partitions) that will serve as coupling facilities and to build a "Sysplex in a box." Given the highly robust mainframe hardware, with a mean time to unplanned customer application loss in excess of 30 years, this middle ground is adequate for many applications that are effectively mission-critical.

5.2.2 NonStop Server

As the engineers gained experience after the initial shipment, they adjusted algorithms to reflect what was seen in the field. Data integrity was and is the key system requirement, and several refinements were made to strengthen end-to-end data checks. Changes included adding end-to-end checksums on messages and sequence numbers to individual packets (originally, messages were simply checksummed at the packet level), and including the disk unit number in sector checksums to enable detection of the right data being written to the wrong disk. A less obviously availability-related feature was the addition of instrumentation to help debug performance problems; this was important because a poorly performing application effectively is unavailable.

The primary hardware change for NonStop servers (aside from the move to CMOS technology) has been to

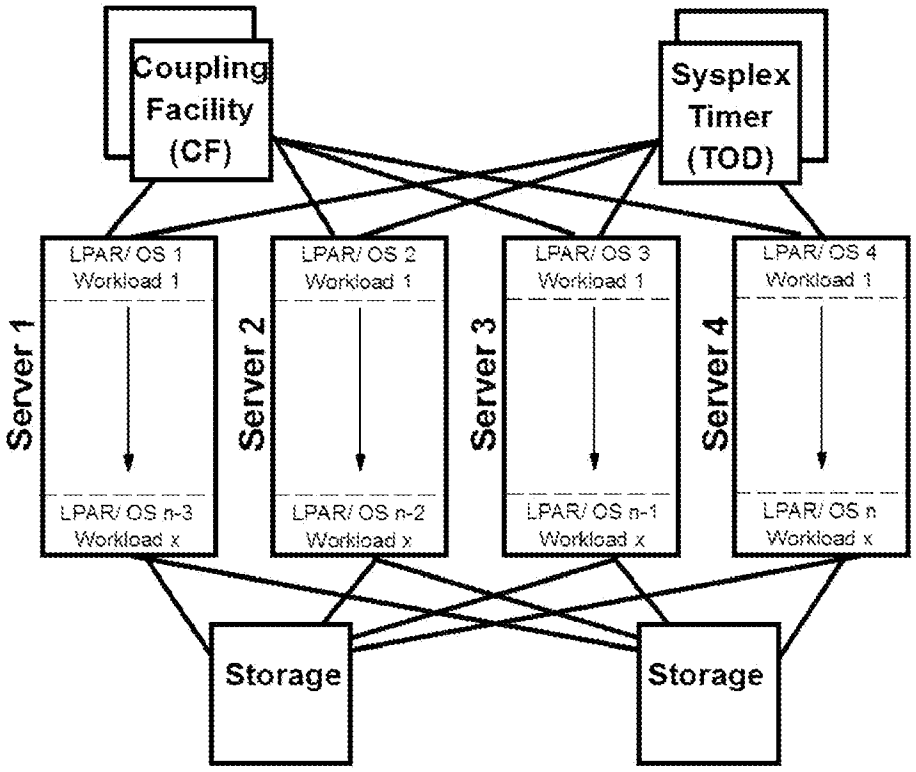


Fig. 4. zSeries parallel sysplex architecture.

rearchitect the system interconnect. As was shown earlier, the original NonStop architecture used a combination of I/O channels and interprocessor buses for its interconnect. Both were designed in-house, once again offering the opportunity to build in an appropriate level of checking. However, the Dynabus had to be greatly overdesigned in terms of capacity in order to support up to 16 processors and allow for future online upgrades to faster processors. In 1997, the NonStop S-series systems were introduced with a completely redesigned interconnect architecture. This architecture, shown in Fig. 5, takes a more network-oriented approach, which by its nature includes the capability to scale up bandwidth in all dimensions as the system grows.

The heart of the design is ServerNet, which carries both I/O and interprocessor traffic. The new interconnect had requirements to improve system performance, both through higher bandwidth and through offloading of part of the software communication stack, while meeting all of the expected fault tolerance requirements [18]. At the time, there were no commercially available interconnects that met all of the requirements, so ServerNet was designed internally. The original design was a 6-port router with 300MB/sec aggregate bandwidth. Wormhole routing was and is utilized to minimize both latency and the need for buffering at the router. DMA transfers, with appropriate protection, remove the need for host-level execution at the remote node. Single-bit command signal errors map to invalid commands. Each packet is protected with a cyclic redundancy check (CRC), and each router recalculates the CRC and appends either a "This Packet Good" or "This Packet Bad" symbol to the packet, making it easy to isolate link failures. ServerNet has subsequently been redesigned to provide even better bandwidth and diagnosability [19].

NonStop servers are inherently clustered due to their loosely coupled design. In the late 1980s, that design was

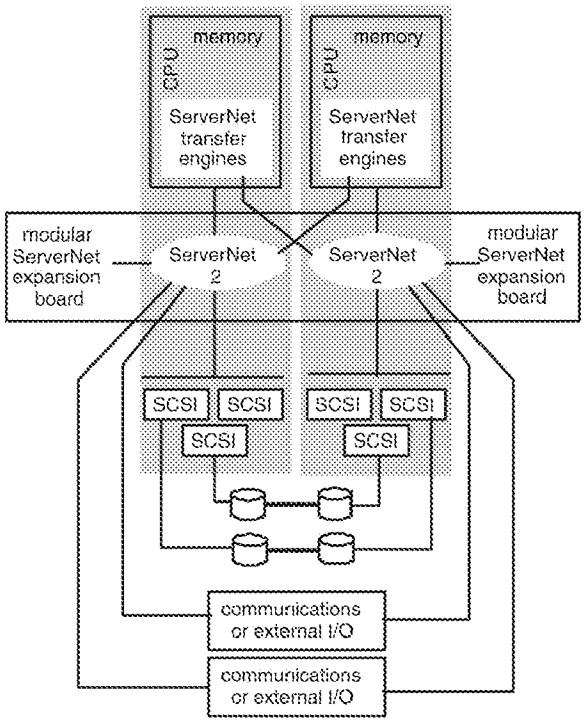


Fig. 5. Tandem NonStop Server S-series architecture (two processors shown).

extended to build the EXPAND networking product supporting homogeneous "clusters of clusters," allowing multiple interconnected systems to function in many ways as a single system image [20]. The original interconnect was via X.25 and other WAN protocols, with support added later for SNA, LAN, and IP. The process-and-message architecture provided a base that could be very straightforwardly extended to support multiple systems. Customers also began to require the ability to group together multiple systems in a data-center environment using high-speed, low-latency connections. A fiber-optic interconnect, FOX, was developed in 1983 to connect systems at subkilometer distances, and EXPAND was optimized to take advantage of the high reliability and speed of the new interconnect. For the S series, the data-center interconnect is ServerNet, which can cover distances of up to 15 KM.

The original software design assumed that programmers would work "close to the metal" and use a programming language, TAL, whose functionality was similar to C, with some customers possibly even supplying their own file systems. However, COBOL support also was available from almost the beginning. It soon became evident that most customers wanted their programmers to be able to focus on the business problem at hand without having to do any special coding to achieve fault tolerance and scalability. The PATHWAY transaction processing monitor [21] was developed to meet this need by handling the distributed and fault-tolerant aspects of the work on the programmer's behalf, with a high degree of transparency as far as the application is concerned. The programmer uses normal business logic and indicates transaction boundaries, and the system takes care of aborting incomplete transactions, restarting server instances, and redirecting requests when an application server process fails.

The other need that soon became evident was for data consistency as well as data integrity—that is to say, a database that is transactionally consistent. The Transaction Management Facility (TMF) was developed in the early 1980s to coordinate data accesses and updates in a manner that guaranteed the ACID properties of atomicity, consistency, isolation, and durability. In order to take advantage of TMF, an application programmer need only determine the appropriate locations in a program to begin, commit, and, if necessary, abort transactions. The Remote Database Facility (RDF) allows a TMF-protected database to be replicated at one or multiple additional sites at any geographic distance.

6 Design Trade Offs

Trade offs in computing system design are generally made among price, performance, and reliability. Most commercial servers are optimized for price, performance, or price/performance, and reliability enhancements which detract from the optimization point are closely scrutinized. Over time, reliability features once found only on servers optimized for reliability have become standard on virtually all servers. For instance, memory ECC and redundant power are now common. Reasons for the evolution include changing economics (e.g., when spare parts and warranty costs are included in the equation, certain reliability features may save money), technology failure rates which are unacceptably high for a broad range of servers, and increased customer expectation for better system availability at almost every price point.

For Tandem and zSeries systems, from the beginning, the target markets had extremely high data integrity requirements and the systems had to be designed accordingly. As described above, both systems were constructed using proprietary processors, which allowed for extensive self checking to be incorporated. Microprocessors have a component to fetch and decode instructions (I-unit), one or more instruction execution elements (E-unit), and a cache. Modern microprocessors usually include parity for cache data and data paths. Checking the control, arithmetic, and logical functions in the I-unit and E-unit is considered to be difficult, time-consuming, and to introduce performance penalties. As a result, other than zSeries, today's microprocessors leave these components unchecked. The Tandem design also checked those components in early systems, but since 1981 the approach has been to use off-chip logic to create perfectly checked microprocessor pairs.

In the industry, accepted wisdom is that single chip microprocessors are sufficiently reliable, that failures are rare and that other mechanisms—timeouts and software detection, for instance—will discover the few problems that do occur. This philosophy depends on continual reliability improvements; in reality, predicting chip technology failure mechanisms is problematic. Future CMOS technologies are expected to be less reliable; latches and dynamic logic will become more susceptible to alpha particles and other cosmic radiation [14]. Latent manufacturing defects and design errors that manifest themselves as transient problems also are sources of customer-visible errors.

Even today there is an exposure to malfunctions in the hardware resulting in incorrect data being written to memory. The model proposed by Horst et al. [22] projects a pessimistic bound of one in 75 unchecked microprocessors causing a data corruption error each year. Horst et al.'s prediction is based on an optimistic evaluation of software detection. It is true that, in some cases, timeouts and software detection will prevent data integrity problems; however, software does not reliably fail fast when errors are injected, resulting in corrupted data [23]. Even when the detection is successful, a system with no recovery will usually hang, causing application downtime, unacceptable for systems designed for high availability. zSeries and NonStop servers are the most well-known commercial systems designed to prevent data corruption by including extensive error checking in all microprocessor functional elements—combinatorial logic as well as arrays.

There are subtle advantages of building in a high level of integrity checking and retry/recovery logic to handle errors. For example, the resulting system is relatively resilient to design flaws. It may be possible to ship a system with one or more remaining occasional transient design errors if the occurrence rate is relatively low and the cost of recovery is not visible at the application level. Another advantage is that test and bringup time can be significantly reduced. The ability to fail-fast often makes it much easier to determine failure sources and uncover design and other problems.

Another important design trade off is "build versus buy." Particularly for subsystems which are compliant with industry standards, like I/O subsystems, there are notable

advantages in using high volume components. Commodity components are used wherever they provide a cost or time-to-market advantage, if the developers are able to provide the expected level of data integrity. For instance, most I/O adapters do not meet zSeries or NonStop server needs for memory protection; that is, inadequate checking is provided to assure that data does not get written to or read from an incorrect location. Some level of customization, through techniques such as lockstepping and end-to-end checksums, is added to incorporate commodity components into the two systems described herein.

6.1 zSeries

zSeries runs multiple operating systems—VM, Linux, and TPF as well as its flagship high-availability operating system, z/OS, the successor of MVS. Similarly to the hardware design principles, z/OS recognizes the inevitability of errors, resulting in a comprehensive approach of error isolation, identification, and recovery. z/OS provides many software reliability and availability features, quite a few of which can also be used by applications which run on z/OS.

Much of the z/OS kernel operating system is dedicated to providing advanced reliability, availability, and serviceability capabilities. All z/OS platform code is written to a set of guidelines to assure that errors can be detected, isolated, and documented. For example, "all code must be protected by a recovery routine including the code of recovery routines themselves; all control areas and queues must be verified before continuing; recovery and retry must be attempted if there is hope of success; all failures which cannot be transparently recovered must be isolated to the smallest possible unit, e.g., the current request, a single task (thread), or single address space (process); diagnostic data must be provided which, as an objective, will allow the problem to be identified and fixed after a single occurrence. The diagnostic data is provided even when retry is attempted and successful" [24].

Approximately 30 percent of the z/OS kernel operating system code is written to adhere to these guidelines, in addition to component code supporting the guidelines or code specifically designed to handle processor, storage, or I/O error recovery. Overall, about half of the z/OS kernel code is devoted to error handling. In addition to adhering to the guidelines, z/OS system software is committed to ensuring system integrity. Data and system functions are protected from unauthorized access, whether accidental or deliberate.

6.2 NonStop Server

Percentages of recovery code are not available for the NonStop server kernel, but the intent is similar to that of z/OS: to identify and contain the scope of faults in order to minimize application-level impact while preserving system and data integrity. Software consistency checking was built into the Tandem operating system from the beginning. As an example, the process ready list is kept as a doubly linked list, with the immediate links checked on every insertion and deletion. Data being written to disk is checkpointed to the backup disk process, but data being written to comm devices is not checkpointed as the performance penalty has been deemed to be too high and comm protocols have built-in recovery mechanisms. Data being written to disk includes host-generated sector checksums, which introduces a small I/O performance penalty and, until recently, required the drives to be specially formatted using 514-byte sectors.

7 The People Factor

In 1985, Gray [12] analyzed a set of Tandem failure data and published his results. His conclusion was that the key to high availability is tolerating operations and software faults—even then, fault-tolerant systems were very effective at masking hardware faults. Almost 20 years later, his conclusion still is valid. Recent evaluation of three very large Internet sites finds operator error to be the number one contributor to system downtime [25]. If anything, increased system complexity has offered even greater opportunities for fatal operational mistakes.

The best tool for reducing operator errors is automation, both to prevent operator errors and to better handle fault analysis and recovery. Automation can help the latter in three ways: It can analyze a long series of events more quickly and thoroughly than a human can, it can speed up recovery by initiating corrective action, and it can reduce the chance of an already perilous situation turning into a full-blown outage due to "fat-finger" mistakes. In cases where an automated event analysis subsystem cannot pinpoint the faulty component, it can at least narrow the options that the human needs to consider. Ideally, a fault-tolerant system also is a self-tending system, monitoring its own health, making corrections as necessary, and alerting the outside world where appropriate.

7.1 zSeries

In 2001, IBM introduced its Autonomic Computing initiative, a longterm strategy to invent, design and deploy computing systems, which operate to achieve business-level policies [26]. The realization of Autonomic Computing will be systems that demonstrate self-CHOP capabilities, that is, self-configuring, healing, optimizing, and protecting. Certain characteristics of zSeries servers exemplify what is meant by each of these.

Dynamic CPU Sparing, a process by which a failed CPU can be transparently replaced by a spare, was earlier described and is a good example of self-healing. Spare CPUs can also be brought online to meet increased customer resource requirements, a process known as Capacity Upgrade on Demand, and an example of self-configuring. LPAR, earlier mentioned as a mechanism for running multiple concurrent operating systems on one mainframe and keeping them isolated from one another, is also a self-optimizing feature. LPAR will maximize resource utilization in the server based on user established priorities. If a high priority partition is not using its allowed resources, LPAR will make them available to lower priority partitions. It will attempt to keep the mainframe as busy as possible all the time. A partition can also be capped, meaning that, even if all other operating system partitions are in a wait state, the capped partition can have only a certain amount of resources. This is useful for service providers who are "renting" capacity to others, but can also serve as a self-protecting mechanism to prevent denial of service attacks from taking over the entire mainframe.

7.2 NonStop Server

Automation is easier to construct if there is a unified logging subsystem to provide access to an integrated stream of information about what is going on in the system. In the mid 1980s, Tandem built a Distributed System Management subsystem (DSM) [27] to meet this need. DSM provides a consolidated log and logging infrastructure that are available to partners and end user applications as well as to the system provider, and includes a call-home capability. DSM events are tokenized, and can be viewed in filtered form to meet specific needs for programmatic response, Web viewing, PDA viewing, hardware incident analyzers, and so forth. However, the logging infrastructure alone is not sufficient; subsystems need to be intelligently instrumented to use it, and incident analysis software needs to be kept updated as new patterns are seen in the field. NonStop servers have long had a high degree of self-tending built into the system, both within the operating system and individual subsystems and, especially for hardware, in the form of incident analyzers.

8 Conclusion

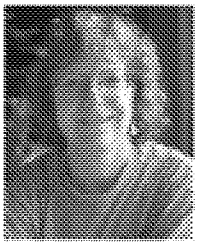
Fault tolerance and rapid fault detection for both hardware and software are key building blocks for constructing continuously available applications, but are not in themselves sufficient to accomplish the task. Similarly, built-in error recovery is important for handling the increasing rate of transient hardware and software errors. But, for builders of highly available systems, much of the current challenge lies in addressing the "second 90 percent" of the problem —the network, the environment, cyber attacks and, most importantly, the people and operational procedures responsible for keeping applications and systems running on an ongoing basis. The move toward self-tending systems by the producers of today's highly available commercial systems is a recognition of that need, and a trend that the authors expect to see continue.

Acknowledgments

Wendy Bartlett would like to thank Joel Bartlett, Bob Taylor, and Kate Wilson. Lisa Spainhower would like to thank Bob Abrams.

References

[1] G.M. Amdahl, G.A. Blaauw, and F.B. Brooks, "Architecture of the IBM System/360," *IBM J. Research and Development*, vol. 8, pp. 87-101, 1964.
[2] W.B. Bartlett and B. Ball, "Tandem's Approach to Fault Tolerance," *Tandem Systems Rev.*, vol. 8, pp. 84-95, Feb. 1988.
[3] L. Spainhower, "System/360 to zSeries: Dependability in IBM Mainframes," *Dependable Computing Systems: Paradigms, Performance Issues and Applications*, 2004.
[4] L. Spainhower and T.A. Gregg, "IBM S/390 Parallel Enterprise Server G5 Fault Tolerance: A Historical Perspective," *IBM J. Research and Development*, vol. 43, nos. 5/6, pp. 863-874, 1999.
[5] P.R. Turgeon, P. Mak, M.A. Blake, M.F. Fee, C.B. Ford III, P.J. Meaney, R. Seigler, and W.W. Shen, "The G5/G6 Binodal Cache," *IBM J. Research and Development*, vol. 43, nos. 5/6, pp. 661-670, 1999.
[6] T.A. Gregg, "S/390 CMOS Server I/O: The Continuing Evolution," *IBM J. Research and Development*, vol. 41, nos. 4/5, pp. 449-462, 1997.
[7] J.A. Katzman, "System Architecture for NonStop Computing," *Proc. IEEE CS Int'l Conf. Technologies for the Information Superhighway*, pp. 77-80, 1977.
[8] J.F. Bartlett, "A 'NonStop' Operating System," *Proc. Hawaii Int'l Conf. System Sciences*, pp. 103-119, 1978.
[9] J.F. Bartlett, "A NonStop Kernel," *Proc. Eighth SIGOPS European Workshop*, pp. 22-29, 1981.
[10] E.W. Dijkstra, "The Structure of the 'THE' Multiprogramming System," *Comm. ACM*, vol. 11, pp 341-346, 1968.
[11] P.B. Hansen, "The Nucleus of a Multi-Programming System," *Comm. ACM*, vol. 13, pp. 238-241, Apr. 1970.
[12] J. Gray, "Why Do Computers Stop and What Can Be Done About It?" Technical Report TR85.7, Tandem Computers, Cupertino, Calif., June 1985.
[13] I. Lee, "Software Dependability in the Tandem Guardian System," *IEEE Trans. Software Eng.*, vol. 8, pp 455-467, May 1995.
[14] C. Constantinescu, "Impact of Deep Submicron Technology on Dependability of VLSI Circuits," *Proc. 2002 Int'l Conf. Dependable Systems and Networks*, pp. 205-209, 2002.
[15] C.F. Webb and J.S. Liptay, "A High-Frequency Custom CMOS S/390 Microprocessor," *IBM J. Research and Development*, vol. 41, nos. 4/5, pp. 463-473, 1997.
[16] D. Siewiorek and R. Swarz, *Reliable Computer Systems: Design and Evaluation*. Burlington, Mass.: Digital Press, pp. 586-648, 1992.
[17] J.M. Nick, B.B. Moore, J.Y. Chung, and N.S. Bowen, "S/390 Cluster Technology: Parallel Sysplex," *IBM Systems J.*, vol. 36, no. 2, pp. 172-201, 1997.
[18] B. Horst, "TNet: A Reliable System Area Network," *IEEE Micro*, vol. 15, no. 1, pp. 37-45, Feb. 1994.
[19] D. Garcia and W. Watson, "ServerNet II," *Proc. Parallel Computing, Routing, and Comm. Workshop*, 1997.
[20] Tandem Computers, *EXPAND Reference Manual*. Cupertino, Calif., 1986.
[21] Tandem Computers, *Introduction to Pathway*. Cupertino, Calif., 1985.
[22] R. Horst, D. Jewett, and D. Lenoski, "The Risk of Data Corruption in Microprocessor-Based Systems," *Proc. Fault-Tolerant Computing Symp.*, pp. 576-585, 1993.
[23] S. Chandra and P.M. Chen, "How Fail-Stop Are Faulty Programs?" *Proc. 28th Int'l Symp. Fault Tolerant Computing*, pp. 240-249, 1998.
[24] S. Turner, C. Henry, G. Horvath, and J. Kibble, "Selecting a Server—Value of S/390," *IBM Redbook SG24-4812-01*, 1999.
[25] D.A. Patterson and D. Oppenheimer, "Architecture and Dependability of Large-Scale Internet Services," *IEEE Internet Computing*, pp. 41-49, Sept.-Oct. 2002.
[26] J.O. Kephart and D.M. Chess, "The Vision of Autonomic Computing," *Computer*, vol. 36, no. 1, pp. 41-50, Jan. 2003.
[27] P. Homan, B. Malizia, and E. Reisner, "Overview of DSM," *Tandem Systems Rev.*, vol. 4, no. 3, Oct. 1988.



Wendy Bartlett received the BS degree in statistics and the MS degree in computer science: computer engineering, both from Stanford University. She is a distinguished technologist in Hewlett Packard's NonStop Enterprise Server Division. She was employed at Four Phase from 1976-1978, and has been employed by Tandem Computers, Compaq, and HP since then. She currently heads the indestructible scalable computing initiative for the NonStop server. She is a member of the IEEE Computer Society.



Lisa Spainhower is a graduate of the University of Michigan. She is a distinguished engineer in the Systems Technology and Architecture organization in IBM's Systems and Technology Group in Poughkeepsie, New York. She is a member of the IBM Academy of Technology, the IEEE, and the IEEE Computer Society.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.

Appendix C

This claim chart compares claims 1-4 to U.S. Patent Publication No. U.S. Patent No. 7,085,959 to Safford ("*Safford*") (Ex. 1007) in light of U.S. Patent No. 5,491,788 to Cepulis ("*Cepulis*"). Safford qualifies as prior art under pre-AIA § U.S.C. 102(a) because it was filed in the United States on July 3, 2002, and was published on January 8, 2004, less than one year before the '302 patent was filed. Cepulis qualifies as prior art because it was filed as an application on September 10, 1993, and issued on February 13, 1996, which is before the earliest claimed effective filing date of October 25, 2004.

The table below reflects SNQ3 presented in the corresponding Request for Reexamination. Further, corroborated evidence reflecting the perspective knowledge of a person having ordinary skill in the art at the time of the invention of the '302 Patent (a "POSITA") is provided through the declaration of Andrew Wolfe ("Wolfe Decl."), (Ex. 1003).

References
Safford (Ex. 1007)
Cepulis (Ex. 1014)

Claim Language	Relevant Prior Art Disclosures and Argument
[1P] A method comprising:	To the extent the preamble is limiting, Safford discloses a "method . . . used to enhance recovery from loss of lock step in a multi-processor computer system." *Safford* (Ex. 1007), Abstract.
[1A] detecting loss of lockstep (LOL) for a processor in a multi-processor system;	Safford discloses, or at least renders obvious, this limitation by disclosing (1) two "processors [that] may operate . . . in a lock step mode" and (2) associated "error detection and signaling logic" that are configure to "detect an error that will cause a loss of lock step." *Safford* (Ex. 1007) at 3:1-45. Additionally, Safford discloses "[detecting] a loss of lock step. . . [by comparing] outputs from [two] microprocessor cores" and detecting "[a] difference in processing" that "is by definition a loss of lock step." *Id.* at 1:51-55. Safford further discloses "detecting a loss of lock step initiating event in a processor." *Id.*, claim 13. More specifically, Safford discloses that "the processor 111 detects an error event that indicates an impending loss of lock step. In block 215, the processor 111 signals the node controller 130 that the first processor pair 111/113 has broken lock step and that the first processor pair 111/113 should be taken 'off-line.'" *Safford* (Ex. 1007) at 4:35-40.

[1B] determining that the processor for which said LOL is detected is assigned a role of boot processor;	Safford discloses, or at least renders obvious, this limitation by disclosing a pair of processors "operating in a lock step mode" that initially "appear to [a] computer system . . . to be . . . a logical CPU 0" but, later, have the logical CPU 0 designation reassigned to another processor. *See Safford* (Ex. 1007) at 3:4-5, 4:52-53 ("The processor pair 125/127 then becomes the logical CPU 0 in the computer system 100"). A POSITA would have recognized "logical CPU 0" as being the designation used by computing system 100 to identify its boot processor for at least the reason that, in multi-processor system architectures like that described in Safford, "logical CPU 0" is commonly used to indicate that a processor has been selected as a boot processor. *See Cepulis* (Ex. 1014) at 2:52-64, *see also Wolfe Decl.* (Ex. 1003) ¶ 172. For example, during boot-processor selection, > If CPU P0 is not functioning properly and cannot perform even the most basic functions, it is designated as inoperative. The hardware system then awakens CPU P1 and repeats the process of testing the CPU. If CPU P1 is operational, then it is designated as CPU L0, and it boots the remainder of the system. On the other hand, if CPU P1 also fails, it is also given an inoperative designation. The computer system then turns on CPU P2, and repeats the process. The process repeats until an operational microprocessor is found to perform the CPU L0 functions.[1] > *Cepulis* (Ex. 1014) at 2:52-64. A POSITA would have also recognized "CPU 0" and "CPU 1" as being physical designations used by computing system 100 to identify processors 111 and 113. The use of both logical and physical designators by a computing system is generally necessary when the role of boot processor can be assigned and reassigned to a processor regardless of its physical position within a system. *See Wolfe Decl.* (Ex. 1003) ¶ 173, *see also Safford* (Ex. 1007) at 4:52-53 ("The processor pair 125/127 then becomes the logical CPU 0 in the computer system 100").

[1] Cepulis is provided herein to explain that a POSITA would have understood the meaning of the term "CPU 0" used in Safford to be the "boot processor." *See* MPEP § 2131.01.

	Thus, a POSITA would have understood "logical CPU 0" to be a boot-processor designation used by an operating system to determine which of its processors is a boot processor and/or to distinguish its boot processor from its non-boot processors. *See Wolfe Decl.* (Ex. 1003) ¶ 174. As such, *determining if* a processor is *assigned the role of boot processor* is performed by the operating system through recognition of the boot processor designation. *See Wolfe Decl.* (Ex. 1003) ¶ 174. Accordingly, Safford discloses, or at least renders obvious, *determining if* a processor is *assigned the role of boot processor* by disclosing a pair of processors "operating in a lock step mode" that initially "appear to [a] computer system . . . to be . . . a logical CPU 0" but, later, have the logical CPU 0 designation reassigned to another processor. *See Safford* (Ex. 1007) at 3:4-5, 4:52-53 ("The processor pair 125/127 then becomes the logical CPU 0 in the computer system 100"), *see also Wolfe Decl.* (Ex. 1003) ¶ 174.
[1C] changing a system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected, to a different value which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor	Safford discloses, or at least renders obvious, this limitation by disclosing a pair of processors "operating in a lock step mode" that initially "appear to [a] computer system . . . to be . . . a logical CPU 0" but, later, have the logical CPU 0 designation reassigned to another processor. *See Safford* (Ex. 1007) at 3:4-5, 4:52-53 ("The processor pair 125/127 then becomes the logical CPU 0 in the computer system 100"). A POSITA would have recognized "logical CPU 0" as being the designation used by computing system 100 to identify its boot processor for at least the reason that, in multi-processor system architectures like that described in Safford, "logical CPU 0" is commonly used to indicate that a processor has been selected as a boot processor. *See Cepulis* (Ex. 1014) at 2:52-64, *see also Wolfe Decl.* (Ex. 1003) ¶ 175. For example, during boot-processor selection, > If CPU P0 is not functioning properly and cannot perform even the most basic functions, it is designated as inoperative. The hardware system then awakens CPU P1 and repeats the process of testing the CPU. If CPU P1 is operational, then it is designated as CPU L0, and it boots the remainder of the system. On the other hand, if CPU P1 also fails, it is also given an inoperative designation. The computer system then turns on CPU P2, and repeats the process. The process repeats until an operational microprocessor is found to perform the CPU L0 functions. *Cepulis* (Ex. 1014) at 2:52-64. Moreover, Safford indicates that, after a first pair of processors has been taken offline, another pair of processors assumes the designation or identifier of CPU0, causing the operating system to no longer recognize processor pair 111 and 113 as the boot processor (i.e., CPU0):

	Alternatively, the node controller 130 may copy the state of the first processor pair 111/113 directly from the processors 111 and 113 to the processors 125 and 127 without any intermediate storage of the architected state in the node controller 130, or other component of the computer system 100. The processor pair 125/127 then becomes the logical CPU 0 in the computer system 100, and the computer system 100 operates without a hot standby processor pair. *Safford* (Ex. 1007) at 4:47-54
[**1D**] changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to a spare processor without shutting down the system's operating system	Safford indicates that, after a first pair of processors has been taken offline, another pair of processors assumes the designation or identifier of CPU0, thereby causing the operating system to recognize a new processor pair 125/127 as the boot processor (i.e., CPU0): Alternatively, the node controller 130 may copy the state of the first processor pair 111/113 directly from the processors 111 and 113 to the processors 125 and 127 without any intermediate storage of the architected state in the node controller 130, or other component of the computer system 100. The processor pair 125/127 then becomes the logical CPU 0 in the computer system 100, and the computer system 100 operates without a hot standby processor pair. *Safford* (Ex. 1007) at 4:47-54. To the extent that Safford in view of Cepulis does not disclose an operating system, a POSITA would recognize that "each processor [would have] its own memory and [would run] its own copy of the operating system." *Commercial Fault Tolerance* (Ex. 1010), section 4.2, *see also Wolfe Decl.* (Ex. 1003) ¶ 178. Moreover, to the extent that Safford in view of Cepulis does not explicitly state that the transition from boot to spare processor is performed without shutting down the operating system, a POSITA would recognize that the intent of Safford's recovery system is to recover from loss of lockstep without crashing or restarting the operating system. *Wolfe Decl.* (Ex. 1003) ¶ 178.
[**2**] wherein the spare processor is a hot spare that is held in idle until switched the role of boot processor.	Safford states: "The processor 125 may be designated as a "**hot standby**" and is **sitting idle** in lock step mode with the processor 127. Should one of the processors 111, 113, 121, and 123 suffer an error, the hot standby processors 125, 127 may be used to **speed recovery** from the resulting loss of lock step. *Safford* (Ex. 1007) at 4:25-29).
[**3**] wherein the spare processor is an active spare.	Safford teaches that the spare processor is an active spare by disclosing "moving the architected state comprises **copying the architected state directly** from the loss of lock step processor unit **to the spare processor unit**." *Id.* at 5:23-26).

[4] wherein said switching further comprises: causing the system's operating system to idle the active spare.	Safford teaches that switching further includes causing the system's operating system to idle the active spare by disclosing a "means for moving an architected state of a processor unit having the loss of lock step initiating event to a **spare processor unit**, the spare processor unit **operating idle** in lock step." *Id.* at 6:2-5.

US007624302B2

(12) **United States Patent**
Michaelis et al.

(10) **Patent No.: US 7,624,302 B2**
(45) **Date of Patent: *Nov. 24, 2009**

(54) **SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR**

(75) Inventors: **Scott L. Michaelis**, Plano, TX (US); **Anurupa Rajkumari**, Round Rock, TX (US); **William B. McHardy**, Fort Collins, CO (US)

(73) Assignee: **Hewlett-Packard Development Company, L.P.**, Houston, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 513 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **10/972,588**

(22) Filed: **Oct. 25, 2004**

(65) **Prior Publication Data**

US 2006/0090064 A1 Apr. 27, 2006

(51) **Int. Cl.**
G06F 11/00 (2006.01)

(52) **U.S. Cl.** **714/12**; 714/11

(58) **Field of Classification Search** 714/10–13; 712/31

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,358,823	A		11/1982	McDonald et al.
5,226,152	A		7/1993	Klug et al.
5,249,188	A		9/1993	McDonald
5,537,655	A		7/1996	Truong
5,675,794	A		10/1997	Meredith
5,751,932	A		5/1998	Horst et al.
5,758,058	A		5/1998	Milburn
5,764,660	A		6/1998	Mohat
5,896,523	A		4/1999	Bissett et al.
5,915,082	A	*	6/1999	Marshall et al. 714/11
5,991,900	A		11/1999	Garnett
6,065,135	A		5/2000	Marshall et al.
6,141,770	A		10/2000	Fuchs et al.
6,148,348	A		11/2000	Garnett et al.
6,220,581	B1		4/2001	Mueller
6,263,452	B1		7/2001	Jewett et al.
6,438,687	B2		8/2002	Klein
6,473,869	B2		10/2002	Bissett et al.
6,560,682	B1		5/2003	Miller et al.

(Continued)

OTHER PUBLICATIONS

U.S. Appl. No. 10/972,796, filed concurrently herewith, entitled "System and Method for Maintaining in a Multi-Processor System a Spare Processor That is in Lockstep for Use in Recovering From Loss of Lockstep for Another Processor".

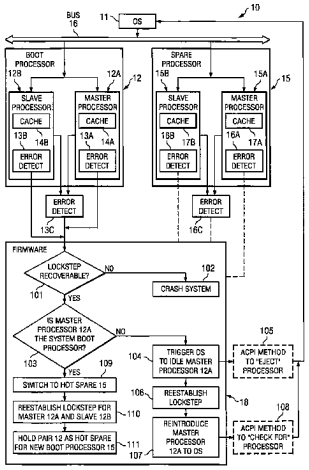
(Continued)

Primary Examiner—Dieu-Minh Le

(57) **ABSTRACT**

According to one embodiment, a method comprises detecting loss of lockstep (LOL) for a processor in a multi-processor system. The method further comprises determining that the processor for which the LOL is detected is assigned the role of boot processor, and switching the role of boot processor to a spare processor without shutting down the system's operating system. In another embodiment, a method comprises system firmware determining that an LOL is detected for a lockstep pair of processors that are assigned the role of boot processor in a system. The method further comprises determining one of the lockstep pair of processors that is not the cause of the LOL, and copying the state of the determined one of the lockstep pair of processors that is not the cause of the LOL to a spare processor. The method further comprises switching the role of boot processor to the spare processor.

27 Claims, 4 Drawing Sheets



U.S. PATENT DOCUMENTS

6,604,177	B1	8/2003	Kondo et al.	
6,615,366	B1	9/2003	Grochowski et al.	
6,625,749	B1 *	9/2003	Quach	714/10
6,675,324	B2	1/2004	Marisetty et al.	
6,687,851	B1	2/2004	Somers et al.	
6,754,787	B2	6/2004	Miller et al.	
6,920,581	B2 *	7/2005	Bigbee et al.	714/10
6,948,092	B2	9/2005	Kondo et al.	
7,003,691	B2	2/2006	Stafford et al.	
7,020,800	B2	3/2006	Fu et al.	
7,085,959	B2 *	8/2006	Safford	714/11
7,134,047	B2	11/2006	Quach	
7,155,721	B2	12/2006	Stafford et al.	
7,191,292	B2	3/2007	Chaudhry et al.	
7,225,355	B2	5/2007	Yamazaki et al.	
7,237,144	B2	6/2007	Stafford et al.	
7,251,746	B2 *	7/2007	Fox et al.	714/13
2002/0144177	A1	10/2002	Kondo et al.	
2002/0152420	A1	10/2002	Chaudhry et al.	
2003/0051190	A1	3/2003	Marisetty et al.	
2003/0070050	A1	4/2003	Miller et al.	
2003/0126498	A1	7/2003	Bigbee et al.	
2003/0135711	A1	7/2003	Shoemaker et al.	
2003/0140281	A1 *	7/2003	Fu et al.	714/38
2004/0006722	A1	1/2004	Safford	
2004/0019771	A1	1/2004	Quach	
2004/0078650	A1	4/2004	Stafford et al.	
2004/0078651	A1	4/2004	Stafford et al.	
2004/0153857	A1	8/2004	Yamazaki et al.	
2005/0172164	A1 *	8/2005	Fox et al.	714/13
2005/0240806	A1	10/2005	Bruckert et al.	
2005/0240811	A1	10/2005	Stafford et al.	
2005/0240829	A1 *	10/2005	Safford et al.	714/43
2006/0085677	A1	4/2006	Stafford et al.	
2006/0136672	A1	6/2006	Chaudhry et al.	
2006/0248384	A1	11/2006	Safford	
2006/0248684	A1	11/2006	Vosbikian et al.	

OTHER PUBLICATIONS

U.S. Appl. No. 10/973,077, filed concurrently herewith, entitled "System and Method for Establishing a Spare Processor for Recovering From Loss of Lockstep in a Boot Processor".

U.S. Appl. No. 10/972,835, filed concurrently herewith, entitled "System and Method for Using Information Relating to a Detected Loss of Lockstep for Determining a Responsive Action".

U.S. Appl. No. 10/973,076, filed concurrently herewith, entitled "System and Method for Providing Firmware Recoverable Lockstep Protection".

U.S. Appl. No. 10/973,004, filed concurrently herewith, entitled "System and Method for Configuring Lockstep Mode of a Prcessor Module".

U.S. Appl. No. 10/973,003, filed concurrently herewith, entitled "System and Method for Reestablishing Lockstep for a Processor Module for Which Loss of Lockstep is Detected".

U.S. Appl. No. 10/972,888, filed concurrently herewith, entitled "System and Method for System Firmware Causing an Operating System to Idle a Processor".

U.S. Appl. No. 10/973,075, filed concurrently herewith, entitled "System and Method for Reintroducing a Processor Module to an Operating System After Lockstep Recovery".

Notice of Allowance for U.S. Appl. No. 10/972,835 dated Nov. 20, 2008 in the United States.

Notice of Allowance for U.S. Appl. No. 10/972,835 dated Aug. 1, 2008 in the United States.

Office Action for U.S. Appl. No. 10/972,835 dated Mar. 3, 2008 in the United States.

Office Action for U.S. Appl. No. 10/972,835 dated Jul. 12, 2007 in the United States.

Office Action for U.S. Appl. No. 10/973,003 dated Mar. 5, 2008 in the United States.

Office Action for U.S. Appl. No. 10/973,003 dated Sep. 12, 2007 in the United States.

Notice of Allowance for U.S. Appl. No. 10/973,076 dated Nov. 28, 2008 in the United States.

Notice of Allowance for U.S. Appl. No. 10/973,076 dated Jul. 16, 2008 in the United States.

Notice of Allowance for U.S. Appl. No. 10/973,076 dated Jun. 3, 2008 in the United States.

Final Office Action for U.S. Appl. No. 10/973,076 dated Feb. 27, 2008 in the United States.

Office Action for U.S. Appl. No. 10/973,076 dated Sep. 12, 2007 in the United States.

Notice of Allowance for U.S. Appl. No. 10/972,888 dated Nov. 14, 2007 in the United States.

Office Action for U.S. Appl. No. 10/972,888 dated Jul. 18, 2007 in the United States.

Notice of Allowance for U.S. Appl. No. 10/973,004 dated Aug. 1, 2007 in the United States.

Office Action for U.S. Appl. No. 10/973,004 dated Apr. 10, 2007 in the United States.

Final Office Action for U.S. Appl. No. 10/973,075 dated May 12, 2009 in the United States.

Office Action for U.S. Appl. No. 10/973,075 dated Oct. 29, 2008 in the United State.

Office Action for U.S. Appl. No. 10/973,075 dated Mar. 12, 2008 in the United States.

Office Action for U.S. Appl. No. 10/973,075 dated Jul. 2, 2007 in the United States.

Notice of Allowance for U.S. Appl. No. 10/972,796 dated Dec. 19, 2007 in the United States.

Office Action for U.S. Appl. No. 10/972,796 dated Jul. 12, 2007 in the United States.

Shoichi Araki, Real-Time Tracking of Multiple Moving Object Contours in a Moving Camera Image Sequence, vol. E83-D IEICE Trans. Inf. & Syst, Jul. 7, 2000, 9 pages.

Jiang-Bin Zheng, Proceedings of the First International Conference on Machine Learning and Cybernetics, Beijing, Nov. 4-5, 2002, 4 pages.

Jong Ryul Kim, Automatic Localization and Trackign of Moving Objects Using Adaptive Snake Algorithm, ICICS-PCM 2003, Dec. 15-18, 2003, 5 pages, Singapore.

Jianhoa Meng, CVEPS-A Compressed Video Editing and Parsing System, Nov. 18, 1996, pp. 1-11.

Notice of Allowance dated Feb. 11, 2009 for U.S. Appl. No. 10/973,077 in the United States.

Notice of Allowance dated Sep. 23, 2008 for U.S. Appl. No. 10/973,077 in the United States.

Office Action dated Mar. 19, 2008 for U.S. Appl. No. 10/973,077 in the United States.

Office Action dated Sep. 20, 2007 for U.S. Appl. No. 10/973,077 in the United States.

Final Office Action for U.S. Appl. No. 10/972,835 dated Nov. 21, 2007 in the United States.

Office Action for U.S. Appl. No. 10/973,075 dated Dec. 13, 2007 in the United States.

Notice of Allowance, dated Jul. 23, 2009, U.S. Appl. No. 10/973,077, 20 pages.

* cited by examiner

FIG. 1

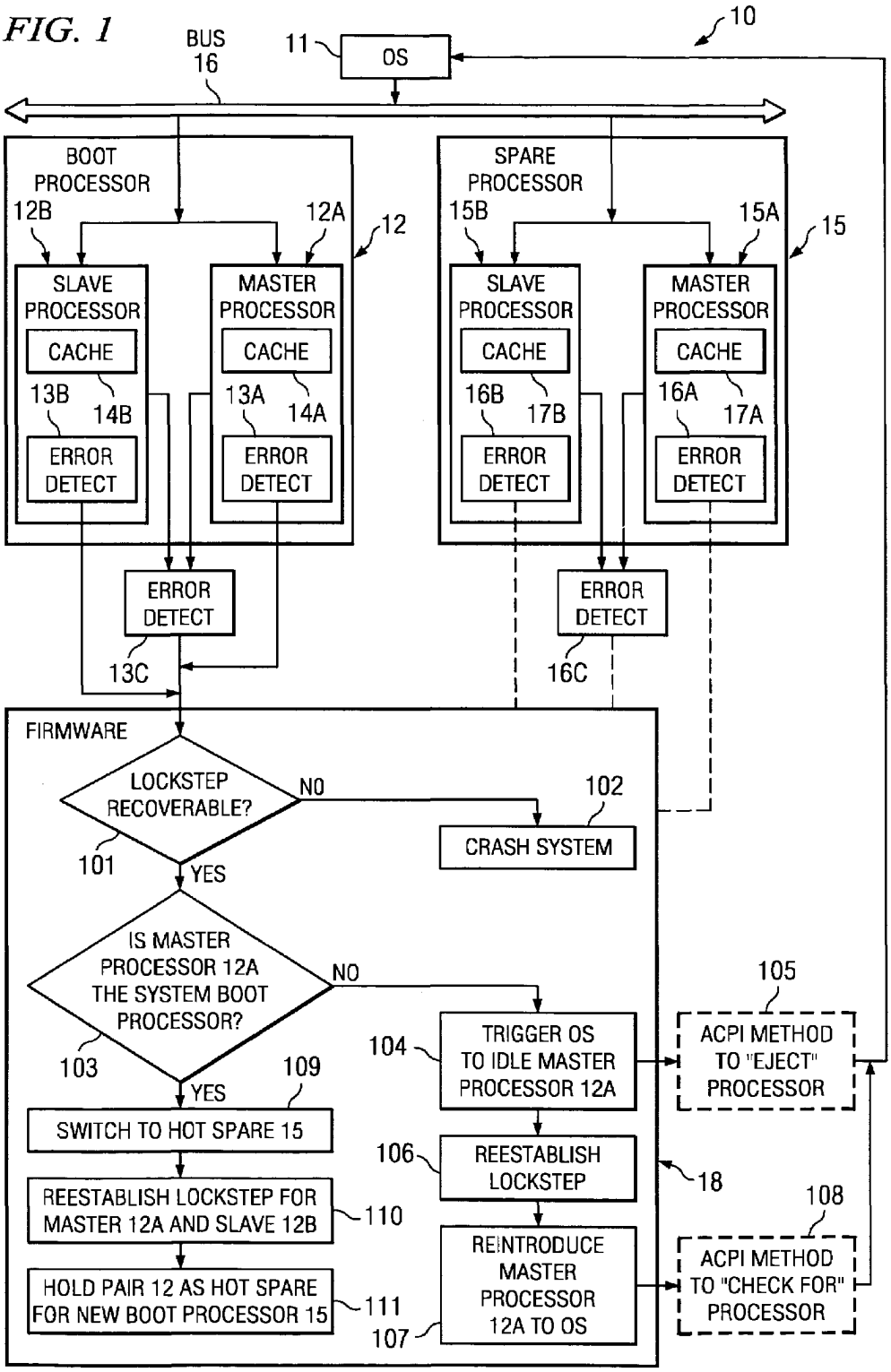

BUS
16
11
OS
10
BOOT PROCESSOR
12B
12A
12
SPARE PROCESSOR
15B
15A
15
SLAVE PROCESSOR
CACHE
13B
14B
ERROR DETECT
MASTER PROCESSOR
CACHE
13A
14A
ERROR DETECT
SLAVE PROCESSOR
CACHE
16B
17B
ERROR DETECT
MASTER PROCESSOR
CACHE
16A
17A
ERROR DETECT
ERROR DETECT
13C
ERROR DETECT
16C
FIRMWARE
LOCKSTEP RECOVERABLE?
NO
102
CRASH SYSTEM
101
YES
IS MASTER PROCESSOR 12A THE SYSTEM BOOT PROCESSOR?
NO
105
ACPI METHOD TO "EJECT" PROCESSOR
104
TRIGGER OS TO IDLE MASTER PROCESSOR 12A
103
YES
109
SWITCH TO HOT SPARE 15
106
REESTABLISH LOCKSTEP
18
REESTABLISH LOCKSTEP FOR MASTER 12A AND SLAVE 12B
110
108
ACPI METHOD TO "CHECK FOR" PROCESSOR
REINTRODUCE MASTER PROCESSOR 12A TO OS
HOLD PAIR 12 AS HOT SPARE FOR NEW BOOT PROCESSOR 15
111
107

FIG. 2

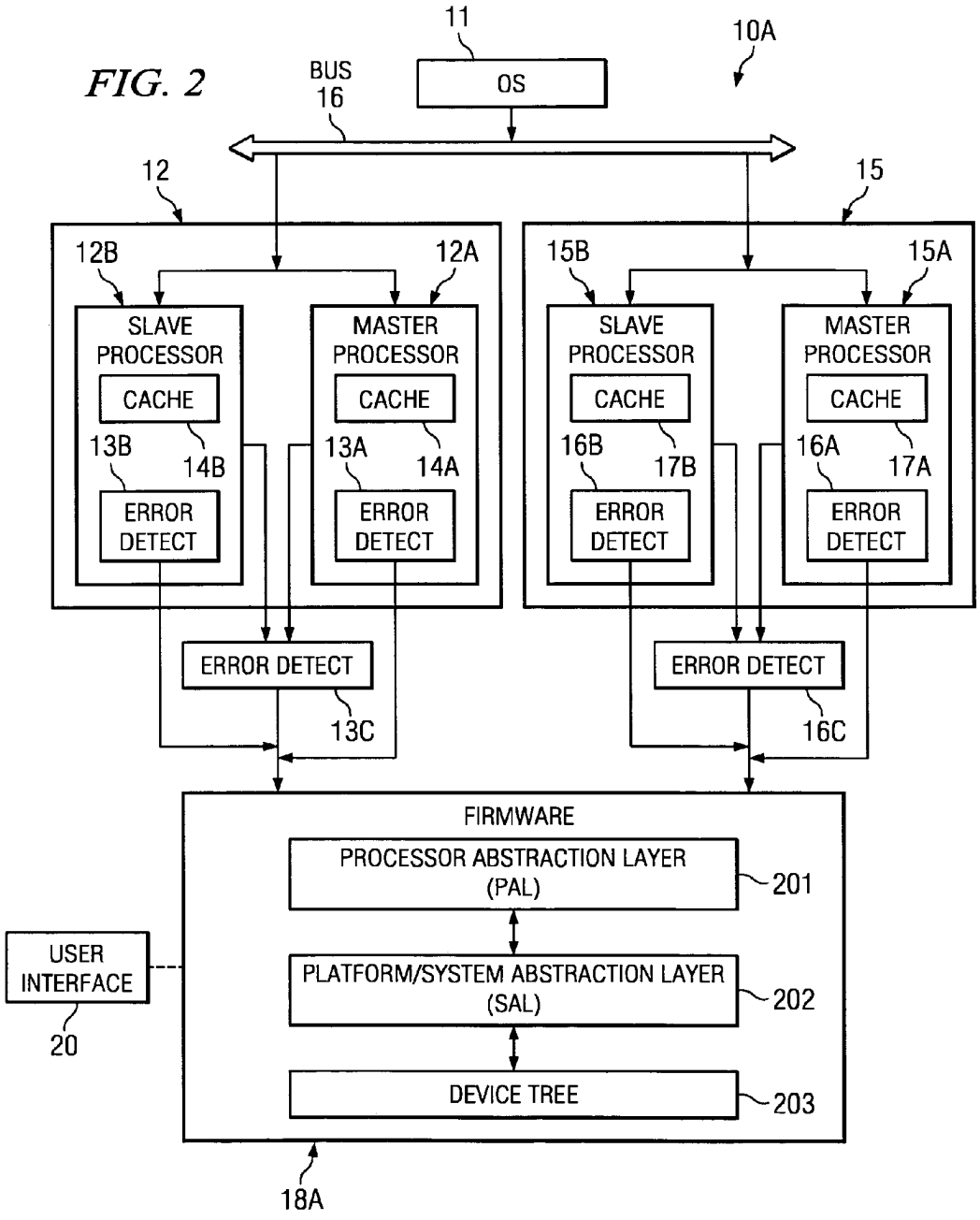

11
10A
BUS
16
OS
12
15
12B
12A
15B
15A
SLAVE PROCESSOR
MASTER PROCESSOR
CACHE
13B
14B
13A
14A
16B
17B
16A
17A
ERROR DETECT
ERROR DETECT
13C
16C
FIRMWARE
PROCESSOR ABSTRACTION LAYER (PAL)
201
PLATFORM/SYSTEM ABSTRACTION LAYER (SAL)
202
DEVICE TREE
203
USER INTERFACE
20
18A

FIG. 3

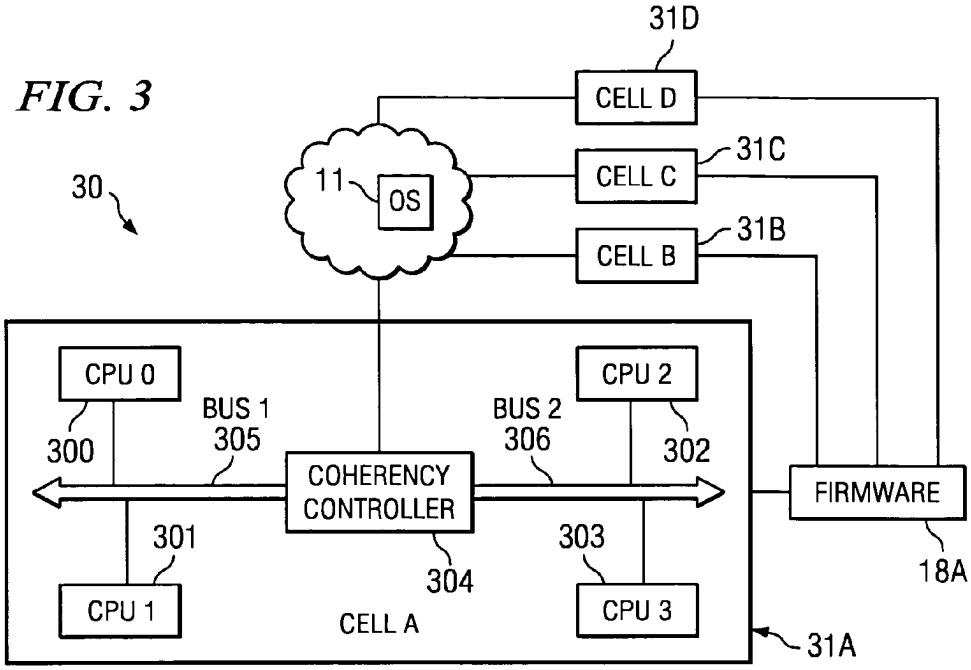

31D
CELL D
31C
CELL C
31B
CELL B
30
11
OS
CPU 0
300
BUS 1
305
BUS 2
306
CPU 2
302
COHERENCY
CONTROLLER
304
FIRMWARE
18A
301
CPU 1
303
CPU 3
CELL A
31A

FIG. 4

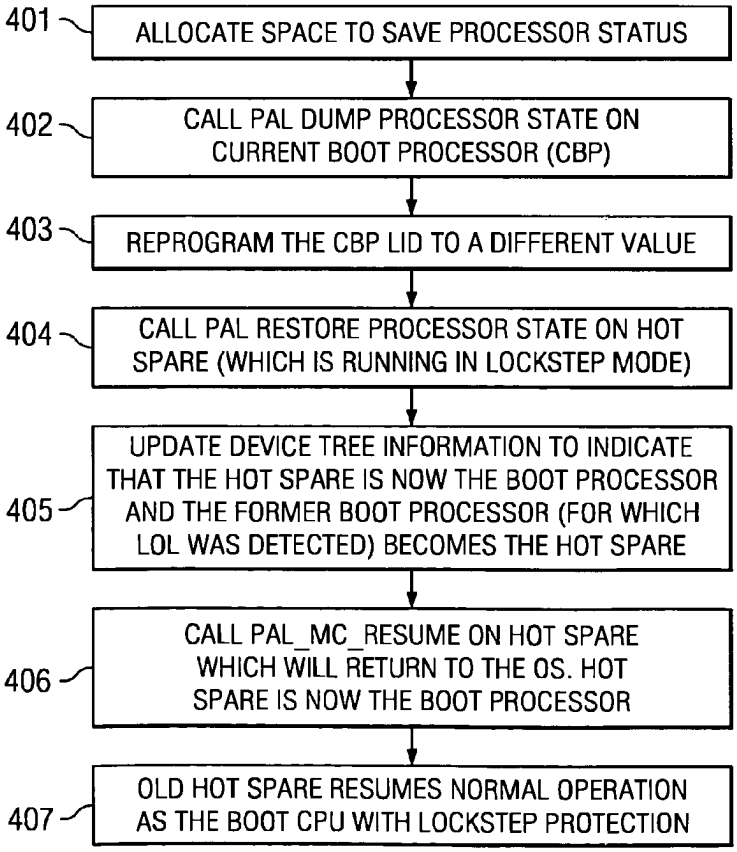

401
ALLOCATE SPACE TO SAVE PROCESSOR STATUS
402
CALL PAL DUMP PROCESSOR STATE ON CURRENT BOOT PROCESSOR (CBP)
403
REPROGRAM THE CBP LID TO A DIFFERENT VALUE
404
CALL PAL RESTORE PROCESSOR STATE ON HOT SPARE (WHICH IS RUNNING IN LOCKSTEP MODE)
405
UPDATE DEVICE TREE INFORMATION TO INDICATE THAT THE HOT SPARE IS NOW THE BOOT PROCESSOR AND THE FORMER BOOT PROCESSOR (FOR WHICH LOL WAS DETECTED) BECOMES THE HOT SPARE
406
CALL PAL_MC_RESUME ON HOT SPARE WHICH WILL RETURN TO THE OS. HOT SPARE IS NOW THE BOOT PROCESSOR
407
OLD HOT SPARE RESUMES NORMAL OPERATION AS THE BOOT CPU WITH LOCKSTEP PROTECTION

FIG. 5

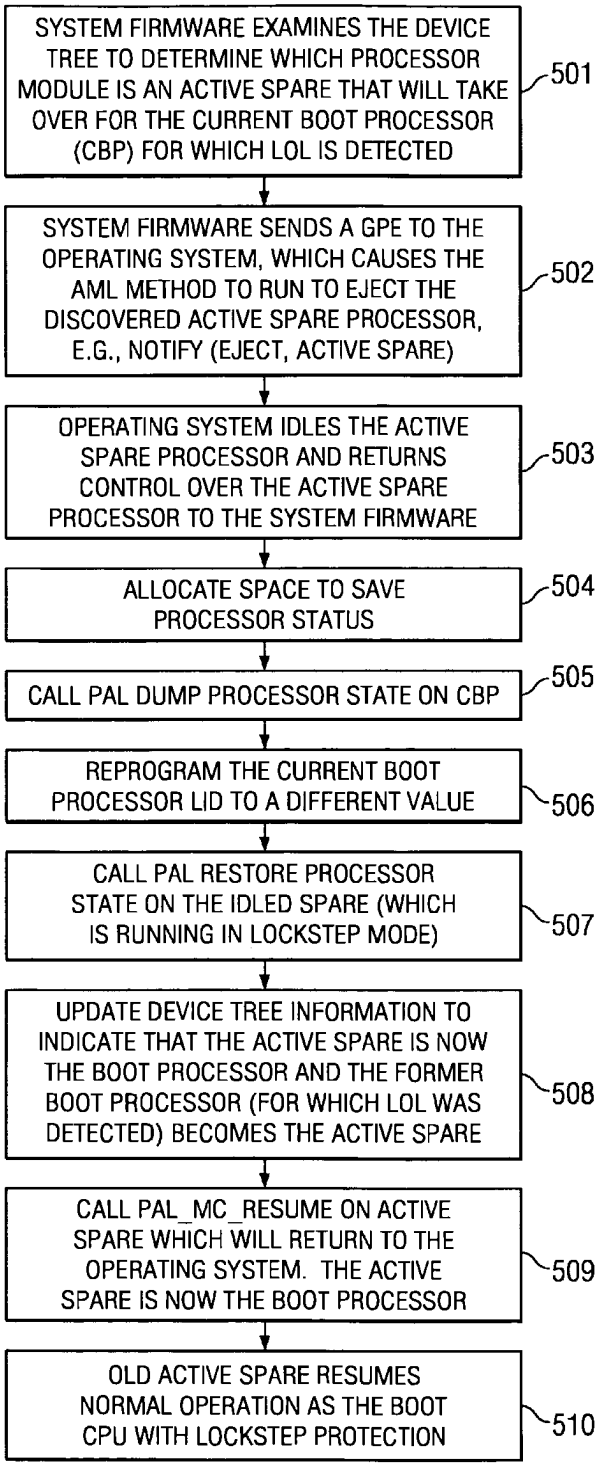
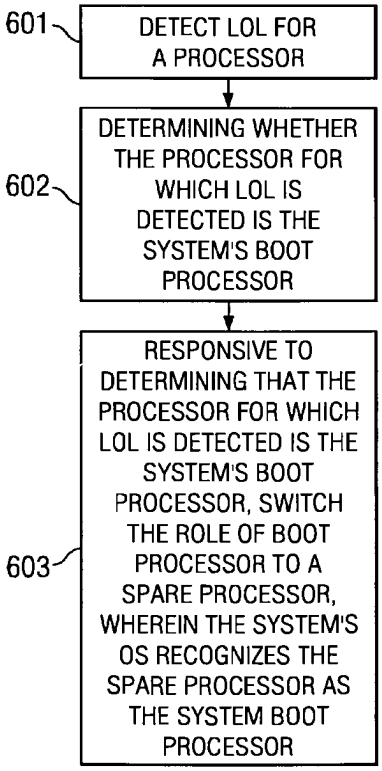

SYSTEM FIRMWARE EXAMINES THE DEVICE TREE TO DETERMINE WHICH PROCESSOR MODULE IS AN ACTIVE SPARE THAT WILL TAKE OVER FOR THE CURRENT BOOT PROCESSOR (CBP) FOR WHICH LOL IS DETECTED
501
SYSTEM FIRMWARE SENDS A GPE TO THE OPERATING SYSTEM, WHICH CAUSES THE AML METHOD TO RUN TO EJECT THE DISCOVERED ACTIVE SPARE PROCESSOR, E.G., NOTIFY (EJECT, ACTIVE SPARE)
502
OPERATING SYSTEM IDLES THE ACTIVE SPARE PROCESSOR AND RETURNS CONTROL OVER THE ACTIVE SPARE PROCESSOR TO THE SYSTEM FIRMWARE
503
ALLOCATE SPACE TO SAVE PROCESSOR STATUS
504
CALL PAL DUMP PROCESSOR STATE ON CBP
505
REPROGRAM THE CURRENT BOOT PROCESSOR LID TO A DIFFERENT VALUE
506
CALL PAL RESTORE PROCESSOR STATE ON THE IDLED SPARE (WHICH IS RUNNING IN LOCKSTEP MODE)
507
UPDATE DEVICE TREE INFORMATION TO INDICATE THAT THE ACTIVE SPARE IS NOW THE BOOT PROCESSOR AND THE FORMER BOOT PROCESSOR (FOR WHICH LOL WAS DETECTED) BECOMES THE ACTIVE SPARE
508
CALL PAL_MC_RESUME ON ACTIVE SPARE WHICH WILL RETURN TO THE OPERATING SYSTEM. THE ACTIVE SPARE IS NOW THE BOOT PROCESSOR
509
OLD ACTIVE SPARE RESUMES NORMAL OPERATION AS THE BOOT CPU WITH LOCKSTEP PROTECTION
510

FIG. 6


601
DETECT LOL FOR A PROCESSOR
602
DETERMINING WHETHER THE PROCESSOR FOR WHICH LOL IS DETECTED IS THE SYSTEM'S BOOT PROCESSOR
603
RESPONSIVE TO DETERMINING THAT THE PROCESSOR FOR WHICH LOL IS DETECTED IS THE SYSTEM'S BOOT PROCESSOR, SWITCH THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR, WHEREIN THE SYSTEM'S OS RECOGNIZES THE SPARE PROCESSOR AS THE SYSTEM BOOT PROCESSOR

SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

CROSS-REFERENCE TO RELATED APPLICATIONS

The present application is related to the following concurrently filed and commonly assigned U.S. Patent Applications: U.S. patent application Ser. No. 10/973,077 (U.S. Published Patent Application 20060107112) titled "SYSTEM AND METHOD FOR ESTABLISHING A SPARE PROCESSOR FOR RECOVERING FROM LOSS OF LOCKSTEP IN A BOOT PROCESSOR"; U.S. patent application Ser. No. 10/973,004 (U.S. Published Patent Application 20060107117, now U.S. Pat. No. 7,308,566) titled "SYSTEM AND METHOD FOR CONFIGURING LOCKSTEP MODE OF A PROCESSOR MODULE"; U.S. patent application Ser. No, 10/972,835 (U.S. Published Patent Application 20060107114) titled "SYSTEM AND METHOD FOR USING INFORMATION RELATING TO A DETECTED LOSS OF LOCKSTEP FOR DETERMINING A RESPONSIVE ACTION"; U.S. patent application Ser. No. 10/972,588 (U.S. Published Patent Application 20060090064) titled "SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR"; U.S. patent application Ser. No. 10/973,003 (U.S. Published Patent Application 20060107116) titled "SYSTEM AND METHOD FOR REESTABLISHING LOCKSTEP FOR A PROCESSOR MODULE FOR WHICH LOSS OF LOCKSThP IS DETECTED"; U.S. patent application Ser. No. 10/972,888 (U.S. Published Patent Application 20060107115) titled "SYSTEM AND METHOD FOR SYSTEM FIRMWARE CAUSING AN OPERATING SYSTEM TO IDLE A PROCESSOR"; U.S. patent application Ser. No. 10/973,075 (U.S. Published Patent Application 20060107111) titled "SYSTEM AND METHOD FOR REINTRODUCING A PROCESSOR MODULE TO AN OPERATING SYSTEM AFTER LOCKSTEP RECOVERY"; U.S. patent application Ser. No. 10/972,796 (U.S. Published Patent Application 20060107106) titled "SYSTEM AND METHOD FOR MAINTAINING IN A MULTI-PROCESSOR SYSTEM A SPARE PROCESSOR THAT IS IN LOCKSTEP FOR USE IN RECOVERING FROM LOSS OF LOCKSTEP FOR ANOTHER PROCESSOR", the disclosures of which arc hereby incorporated herein by reference.

DESCRIPTION OF RELATED ART

Silent Data Corruption ("SDC") is a difficult problem in the computing industry. In general, SDC refers to data that is corrupt, but which the system does not detect as being corrupt. SDCs primarily occur due to one of two factors: a) a broken hardware unit or b) a "cosmic" event that causes values to change somewhere in the system. Broken hardware means that a "trusted" piece of hardware is silently giving wrong answers. For example, the arithmetic unit in a processor is instructed to add 1+1 and it returns the incorrect answer 3 instead of the correct answer 2. An example of a cosmic event is when a charged particle (e.g., alpha particle or cosmic ray) strikes a region of a computing system and causes some bits to change value (e.g., from a 0 to a 1 or from a 1 to a 0).

Numerous techniques have been developed for detecting SDC to prevent the SDC from remaining "silent" or "undetected" within a system, as well as preventing such SDC from propagating through the system. Examples of these techniques include parity-based mechanisms and error correcting codes (ECCs) on buses and memory locations, as well as checksums and/or cyclic redundancy checks (CRC) over regions of memory. Parity-based mechanisms are often employed in processors, wherein a parity bit is associated with each block of data when it is stored. The parity bit is set to one or zero according to whether there is an odd or even number of ones in the data block. When the data block is read out of its storage location, the number of ones in the block is compared with the parity bit. A discrepancy between the values indicates that the data block has been corrupted. ECCs are parity-based mechanisms that track additional information for each data block. The additional information allows the corrupted bit(s) to be identified and corrected.

Parity/ECC mechanisms have been employed extensively for caches, memories, and similar data storage arrays. In the remaining circuitry on a processor, such as data paths, control logic, execution logic, and registers (the "execution core"), it is more difficult to apply parity/ECC mechanisms for SDC detection. Thus, there is typically some unprotected area on a processor in which data corruption may occur and the parity/ECC mechanisms do not prevent the corrupted data from actually making it out onto the system bus. One approach to SDC detection in an execution core (or other unprotected area of the processor chip) is to employ "lockstep processing." Generally, in lockstep processing two processors are paired together, and the two processors perform exactly the same operations and the results are compared (e.g., with an XOR gate). If there is ever a discrepancy between the results of the lockstep processors, an error is signaled. The odds of two processors experiencing the exact same error at the exact same moment (e.g., due to a cosmic event occurring in both processors at exactly the same time or due to a mechanical failure occurring in each processor at exactly the same time) is nearly zero.

A pair of lockstep processors may, from time to time, lose their lockstep. "Loss of lockstep" (or "LOL") is used broadly herein to refer to any error in the pair of lockstep processors. One example of LOL is detection of data corruption (e.g., data cache error) in one of the processors by a parity-based mechanism and/or ECC mechanism. Another example of LOL is detection of the output of the paired processors not matching, which is referred to herein as a "lockstep mismatch." It should be recognized that in some cases the data in the cache of a processor may become corrupt (e.g., due to a cosmic event), which once detected (e.g., by a parity-based mechanism or ECC mechanism of the processor) results in LOL. Of course, unless such corrupt data is acted upon by the processor, the output of that processor will not fail to match the output of its paired processor and thus a "lockstep mismatch" will not occur. For example, suppose that a value of "1" is stored to first location of cache in each of a pair of lockstep processors and a value of "1" is also stored to a second location of cache in each of the pair of lockstep processors. Further suppose that a cosmic event occurs for a first one of the processors, resulting in the first location of its cache being changed from "1" to "0", and thus corrupted. This data corruption in the first processor is a LOL for the pair. An error detection mechanism of this first processor may detect the data corruption, thus detecting the LOL. If the processors are instructed to act on the data of their first cache locations, then a lockstep mismatch will occur as the output of each of the processors will not match. For instance, if the processors each add the data

stored to the first location of their respective cache with the data stored to the second location of their respective cache, the first processor (having the corrupt data) will output a result of "1" (0+1=1) while the second processor outputs a result of "2" (1+1=2), and thus their respective outputs will not match.

By employing such techniques as parity-based error detection mechanisms and output comparisons for lockstep paired processors, SDC detection can be enhanced such that practically no SDC occurring in a processor goes undetected (and thus such SDC does not remain "silent") but instead results in detection of LOL. However, the issue then becomes how best for the system to respond to detected LOL. The traditional response to detected LOL has been to crash the system to ensure that the detected error is not propagated through the system. That is, LOL in one pair of lockstep processors in a system halts processing of the system even if other processors that have not encountered an error are present in the system. However, with the increased desire for many systems to maintain high availability, crashing the system each time LOL is detected is not an attractive proposition. This is particularly unattractive for large systems having many processors because cosmic events typically occur more frequently as the processor count goes up, which would result in much more frequent system crashes in those large systems. High availability is a major desire for many customers having large, multi-processor systems, and thus having their system crash every few weeks is not an attractive option. Of course, permitting corrupt data to propagate through the system is also not a viable option.

Prior solutions attempting to resolve at least some detected SDCs without requiring the system to be crashed have been Operating System ("OS") centric. That is, in certain solutions the OS has been implemented in a manner to recover from a detected LOL without necessarily crashing the system. This OS-centric type of solution requires a lot of processor and platform specific knowledge to be embedded in the OS, and thus requires that the OS provider maintain the OS up-to-date as changes occur in later versions of the processors and platforms in which the OS is to be used. This is such a large burden that most commonly used OSs do not support lockstep recovery.

Certain solutions have attempted to recover from a LOL without involving the OS in such recovery procedure. For instance, in one technique upon LOL being detected, firmware is used to save the state of one of the processors in a lockstep pair (the processor that is considered "good") to memory, and then both processors of the pair are reset and reinitialized. Thereafter, the state is copied from the memory to each of the processors in the lockstep pair. This technique makes the processors unavailable for an amount of time without the OS having any knowledge regarding this unavailability, and if the amount of time required for recovery is too long, the system may crash. That is, typically, if a processor is unresponsive for X amount of time, the OS will assume that the processor is hung and will crashdump the system so that the problem can be diagnosed. Further, in the event that a processor in the pair cannot be reset and reinitialized (e.g., the processor has a physical problem and fails to pass its self-test), this technique results in crashing the system.

BRIEF SUMMARY OF THE INVENTION

According to one embodiment, a method comprises detecting loss of lockstep (LOL) for a processor in a multi-processor system. The method further comprises determining that the processor for which the LOL is detected is assigned the role of boot processor, and switching the role of boot processor to a spare processor without shutting down the system's operating system.

According to one embodiment, a method comprises system firmware determining that loss of lockstep (LOL) is detected for a lockstep pair of processors that are assigned the role of boot processor in a system. The method further comprises determining one of the lockstep pair of processors that is not the cause of the LOL, and copying the state of the determined one of the lockstep pair of processors that is not the cause of the LOL to a spare processor. The method further comprises switching the role of boot processor to the spare processor.

According to one embodiment, a system comprises a plurality of processor modules that each include a master processor and a slave processor that operate in lockstep. The system further comprises an operating system, and error detection logic operable to detect loss of lockstep (LOL) for at least one of the processor modules. The system further comprises system firmware operable, responsive to detection of LOL for a first of the processor modules, to determine whether the first processor module is assigned a role of system boot processor. If determined that the first processor module is assigned the role of system boot processor, the system firmware is further operable to cause the operating system to recognize another processor module as system boot processor without shutting down the operating system.

According to one embodiment, a system comprises means for detecting loss of lockstep (LOL) for a processor in a multi-processor system. The system further comprises means for determining whether the processor for which the LOL is detected is assigned the role of boot processor, and means for switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which the LOL is detected is assigned the role of boot processor.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** shows an example multi-processor system in which firmware utilizes a spare processor for recovering from LOL for the system's boot processor;

FIG. **2** shows a block diagram of one embodiment implemented for the IA-64 processor architecture;

FIG. **3** shows an example system having multi-processor cells in which an embodiment for switching the role of boot processor to a spare processor responsive to detected LOL in the boot processor may be employed;

FIG. **4** shows an exemplary operational flow diagram for switching the role of boot processor to a hot spare processor according to one embodiment;

FIG. **5** shows an exemplary operational flow diagram for switching the role of boot processor to an active spare processor according to one embodiment; and

FIG. **6** shows a more general operational flow diagram for certain embodiments.

DETAILED DESCRIPTION OF THE INVENTION

As described further herein and in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/973,076 (U.S. Published Patent Appplication 20060107107) titled "SYSTEM AND METHOD FOR PROVIDING FIRMWARE RECOVERABLE LOCKSTEP PROTECTION," the disclosure of which is incorporated herein by reference, certain techniques are provided for recovering from LOL detected for a boot processor in a multi-processor system, in

which a spare processor is utilized for such recovery. For instance, upon LOL being detected for the system's boot processor, the spare processor assumes the role of boot processor, and then recovery of lockstep for the original boot processor is attempted. Accordingly, for such lockstep recovery technique, a spare processor is used for attempting to recover from LOL detected for the boot processor. Exemplary techniques are described in concurrently flied and commonly assigned U.S. patent application Ser. No. 10/973,077 (U.S. Published Patent Application 20060107112) titled "SYSTEM AND METHOD FOR ESTABLISHING A SPARE PROCESSOR FOR RECOVERING FROM LOSS OF LOCKSTEP IN A BOOT PROCESSOR," the disclosure of which is incorporated herein by reference, for establishing a spare processor for the boot processor. Such exemplary techniques include techniques for establishing a hot spare processor that is held in reserve until such time as it is used to assume the role of boot processor responsive to a detected LOL in the boot processor. The exemplary techniques further include techniques for establishing an active spare processor that is not held in reserve, but is instead made available to the OS for executing instructions during normal system operation and is dynamically transformed into a hot spare (i.e., it is idled) such that it can assume the role of the boot processor responsive to detection of LOL in the boot processor.

Embodiments are provided herein for switching the role of boot processor to an established spare processor responsive to detection of LOL in the boot processor. For instance, exemplary actions by system firmware for switching the role of boot processor to an established spare processor and thus causing the OS to recognize the established spare processor as the system's boot processor are provided in accordance with certain embodiments hereof.

As described further in concurrently filed and commonly assigned U.S. patent application Ser. No. U.S. patent application Ser. No. 10/973,077 (U.S. Published patent application 20060107112) titled "SYSTEM AND METHOD FOR ESTABLISHING A SPARE PROCESSOR FOR RECOVERING FROM LOSS OF LOCKSTEP IN A BOOT PROCESSOR," in certain embodiments, information (referred to herein as a "device tree") is stored to a multi-processor system, wherein such information identifies the processor in the system that is the boot processor and identifies the processor that is the spare for the boot processor. The information is stored such that it is accessible by the system's firmware, and the firmware, upon detecting LOL for the boot processor, determines the spare processor from the device tree and assigns the spare processor the role of boot processor. The firmware then attempts to recover lockstep for the processor for which LOL was detected. If the recovery of lockstep is successful, the processor for which lockstep was recovered (i.e., the former boot processor) may be assigned the role of spare for the current boot processor. Thus, the multi-processor system may recover from LOL detected for the boot processor without requiring a crash of the system (at least for certain types of "recoverable" LOLs).

In some implementations described herein, the spare processor for the boot processor is a "hot" spare, in which it is idled and not made available to the OS for processing instructions during normal system operation. That is, the hot spare processor is held in reserve, idling until such time as it assumes the role of boot processor responsive to detected LOL for the boot processor. In other implementations, the spare processor is not held in reserve, but is instead an "active" processor that is made available to the OS for executing instructions during normal system operation. Upon LOL being detected for the boot processor, the designated spare processor is dynamically transformed into a hot spare (i.e., it is idled) such that it can assume the role of the boot processor. For instance, upon the system's firmware detecting a LOL for the boot processor, the firmware accesses the device tree and determines the designated spare processor, and then the firmware instructs the OS to idle the designated spare processor (i.e., stop scheduling instructions for the processor) to make the "active" spare a "hot" spare, which then assumes the role of boot processor.

Concurrently filed and commonly assigned U.S. patent application Ser. No. 10/973,076 (U.S. Published Patent Application 20060107107) titled "SYSTEM AND METHOD FOR PROVIDING FIRMWARE RECOVERABLE LOCKSTEP PROTECTION," discloses a technique in which lockstep can be recovered for processors (particularly, non-boot processors) without requiring use of a spare processor. Rather, upon LOL being detected for a lockstep pair of processors, the firmware instructs the OS to idle (or "eject") the lockstep pair of processors for which LOL was detected and then the firmware attempts to reestablish lockstep (e.g., by resetting the processors). If successful in reestablishing lockstep, the firmware triggers the OS to recognize the lockstep pair of processors as again being available for servicing instructions.

However, for various reasons, in certain system architectures problems arise in attempting to idle (or eject) the boot processor from the system. Thus, in certain embodiments, a spare processor is used for recovering from LOL for the system's boot processor. Again, embodiments are provided herein for switching the role of boot processor to an established spare processor responsive to detection of LOL for the boot processor. In certain embodiments, the spare processor is used for recovering from LOL only in the case in which LOL is detected for the boot processor. For instances in which LOL is detected for non-boot processors, the lockstep may be recovered without the use of a spare by using the recovery technique described in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/973,076 (U.S. Published Patent Application 20060107107 titled "SYSTEM AND METHOD FOR PROVIDING FIRMWARE RECOVERABLE LOCKSTEP PROTECTION," for example.

Turning to FIG. **1**, an example embodiment of a system **10** in which a spare processor is used in recovering from detected loss of lockstep (LOL) for a system boot processor is shown. System **10** includes OS **11**, as well as master processor **12**A and slave processor **12**B (collectively referred to as a lockstep processor pair **12**). In certain implementations the lockstep processor pair **12** may be implemented on a single silicon chip, which is referred to as a "dual core processor" in which master processor **12**A is a first core and slave processor **12**B is a second core. Further, lockstep processor pair **12** may be referred to as a processor or CPU "module" because it includes a plurality of processors (**12**A and **12**B) in such module. As described further herein, to certain portions of system **10**, such as its OS **11**, the processor module **12** may appear as a single processor because slave processor **12**B is merely used for redundant processing for checking the output of master processor **12**A. In this example, lockstep processor pair **12** is designated as the boot processor for system **10**. As described further herein, in certain embodiments lockstep processor pair **12** is designated as the system boot processor by storing information to the system that specifies such designation.

Master processor **12**A includes cache **14**A, and slave processor **12**B includes cache **14**B. OS **11** and lockstep processor pair **12** are communicatively coupled to bus **16**. Typically, master processor **12**A and slave processor **12**B are coupled to

bus **16** via an interface that allows each of such processors to receive the same instructions to process, but such interface only communicates the output of master processor **12**A back onto bus **16**. The output of slave processor **12**B is used solely for checking the output of mater processor **12**A.

In this example, master processor **12**A includes error detect logic **13**A, and slave processor **12**B includes error detect logic **13**B. While shown as included in each of the processors **12**A and **12**B in this example, in certain embodiments the error detect logic **13**A and **13**B may be implemented external to processors **12**A and **12**B. Error detect logic **13**A and **13**B include logic for detecting errors, such as data cache errors, present in their respective processors **12**A and **12**B. Examples of error detect logic **13**A and **13**B include known parity-based mechanisms and ECC mechanisms. Error detect logic **13**C is also included, which may include an XOR (exclusive OR) gate, for detecting a lockstep mismatch between master processor **12**A and slave processor **12**B. As mentioned above, a lockstep mismatch refers to the output of master processor **12**A and slave processor **12**B failing to match. While shown as external to the lockstep processor pair **12** in this example, in certain embodiments error detect logic **13**C may be implemented on a common silicon chip with processors **12**A and **12**B.

Lockstep mismatch is one way of detecting a LOL between the master processor **12**A and slave processor **12**B. A detection of an error by either of error detect logic **13**A and **13**B also provides detection of LOL in the processors **12**A and **12**B. Because the detection of LOL by error detect logic **13**A and **13**B may occur before an actual lockstep mismatch occurs, the detection of LOL by error detect logic **13**A and **13**B may be referred to as a detection of a "precursor to lockstep mismatch". In other words, once an error (e.g., corrupt data) is detected by error detect logic **13**A or **13**B, such error may eventually propagate to a lockstep mismatch error that is detectable by error detect logic **13**C.

System **10** further includes a second lockstep processor pair **15** that is communicatively coupled to bus **16**. Lockstep processor pair **15** includes master processor **15**A and slave processor **15**B. Master processor **15**A includes cache **17**A, and slave processor **15**B includes cache **17**B. As with error detect logic **13**A-**13**C implemented for lockstep processor pair **12**, lockstep processor pair **15** has error detect logic **16**A-**16**C. While only two lockstep processor pairs **12** and **15** are shown for simplicity in the example of FIG. **1**, system **10** may include any number of such lockstep processor pairs. As one specific example, in one implementation system **10** includes **64** lockstep processor pairs, wherein the master processors of the pairs perform parallel processing for the system. In the specific example of FIG. **1**, lockstep processor pair **15** is designated as a spare for boot processor **12**, as described further herein.

In certain embodiments described herein, techniques are provided for switching the role of boot processor to the designated spare processor responsive to detection of LOL for the boot processor. Thus, the spare processor is used for recovery when LOL is detected for the system's boot processor. That is, the designated spare processor assumes the role of boot processor in the event that LOL is detected for the boot processor. In accordance with certain embodiments, upon LOL being detected for a boot processor, a hot spare processor (i.e., an idling processor that is available in the system) is transferred the role of boot processor, and then the old boot processor having LOL is reset to reestablish its lockstep. In other embodiments, a processor is designated as a spare processor for the boot processor, but such designated spare processor is not held in reserve (idled) during normal system operation. Rather, the designated spare is utilized as an active processing resource made available to OS **11** for processing instructions that OS **11** schedules for such "active" spare processor. Upon LOL being detected for the boot processor, the designated spare processor is idled (thereby, transforming such active processor into a hot spare), and is transferred the role of boot processor. Thereafter, the former boot processor for which the LOL was detected may be reset to reestablish its lockstep. Thus, in certain embodiments, a designated spare processor is not required to be held in reserve, but is instead an available resource for use within the system and is only transformed into an actual spare processor when needed for recovering from LOL for the boot processor.

Firmware **18** is also included in system **10**, which in this embodiment is invoked upon an error being detected by any of the error detect logics **13**A-**13**C and **16**A-**16**C. In certain embodiments, processors **12**A, **12**B, **15**A, and **15**B are processors from the Itanium Processor Family (IPF). IPF is a 64-bit processor architecture co-developed by Hewlett-Packard Company and Intel Corporation, which is based on Explicitly Parallel Instruction Computing (EPIC). IPF is a well-known family of processors. IPF includes processors such as those having the code names of MERCED, MCKINLEY, and MADISON. In addition to supporting a 64-bit processor bus and a set of 128 registers, the 64-bit design of IPF allows access to a very large memory (VLM) and exploits features in EPIC. While a specific example implementation of one embodiment is described below for the IPF architecture, embodiments described herein for establishing a spare processor for recovering from LOL detected for a boot processor are not limited in application to an IPF architecture, but may be applied as well to other architectures (e.g., 32-bit processor architectures, etc.).

Processor architecture generally comprises corresponding supporting firmware, such as firmware **18** of system **10**. For example, as described further below in conjunction with the specific example of FIG. **2**, the IPF processor architecture comprises such supporting firmware as Processor Abstraction Layer (PAL), System Abstraction Layer (SAL), and Extended Firmware Interface (EFI). Such supporting firmware may enable, for example, the OS to access a particular function implemented for the processor. For instance, the OS may query the PAL as to the size of the cache implemented for the processor, etc. Other well-known functions provided by the supporting firmware (SAL, EFI) include, for example: (a) performing I/O configuration accesses to discover and program the I/O Hardware (SAL_PCI_CONFIG_READ and SAL_PCI_CONFIG_WRITE); (b) retrieving error log data from the platform following a Machine Check Abort (MCA) event (SAL_GET_STATE_INFO); (c) accessing persistent store configuration data stored in non-volatile memory (EFI variable services: GetNextVariableName, GetVariable and SetVariable); and accessing the battery-backed real-time clock/calendar (EFI GetTime and SetTime). Accordingly, the supporting firmware, such as the PAL, is implemented to provide an interface to the processor(s) for accessing the functionality provided by such processor(s). Each of those interfaces provide standard, published procedure calls that are supported.

In the example system **10** of FIG. **1**, firmware **18** utilizes the above-mentioned hot spare technique for recovering from LOL for a boot processor. That is, processor **15** is designated as a hot spare for boot processor **12**, and such processor **15** is held as an idle, spare processor and used in the manner described further herein for assuming the role of boot processor in the event that a LOL is detected for boot processor **12**. Of course, as described further herein, the above-mentioned

technique in which a designated spare is not held in reserve but is instead dynamically transformed into a spare processor may be implemented by firmware **18** in alternative embodiments. As described further herein, in certain embodiments, a device tree is stored to a data storage device (e.g., SRAM) such that it is accessible to the firmware **18**, wherein the device tree includes information identifying processor **12** as the system's boot processor and identifying processor **15** as the spare. The information specifying the boot processor and/or spare may be received as input from a user in certain embodiments. For instance, in certain embodiments a user interface is provided that enables a system administrator to designate the boot processor and/or spare processor in the multi-processor system. In other embodiments, the firmware **18** selects the boot processor and/or spare.

Suppose that during the system boot-up procedure, master processor **12**A assumes the role of boot processor, as shown in the example of FIG. **1**. In this example embodiment, lockstep processor pair **15** is held as a hot spare for recovering from a LOL that may be detected for the boot processor **12**A. Of course, additional lockstep processor pairs may be included in system **10** (not specifically shown in the example of FIG. **1**), and those additional lockstep processor pairs, which are not the system boot processor, may recover from LOL in a different manner that does not utilize a hot spare processor. Thus, in accordance with certain embodiments hereof, hot spare processor pair **15** is not needed for recovering from LOL detected for any non-boot processor, but may instead be used for recovery of LOL only if such LOL is detected for the boot processor (processor **12**A in this example).

Firmware **18** is included in this example, and upon detection of LOL by any of error detect logics **13**A-**13**C, firmware **18** determines, in operational block **101**, whether the detected LOL is a recoverable LOL. That is, firmware **18** determines in block **101** whether the detected LOL is of a type from which the firmware can recover lockstep for the lockstep processor pair **12** without crashing the system. As described further herein, lockstep is recoverable for certain detected LOLs (which may be referred to as "recoverable LOLs"), while lockstep is not recoverable for other detected LOLs (which may be referred to as "non-recoverable LOLs"). If the lockstep is not recoverable from the detected LOL, then in the example of FIG. **1** firmware **18** crashes the system in block **102**.

If the lockstep is recoverable, then operation advances to block **103** whereat the firmware determines whether the processor for which the LOL was detected is the system's boot processor. Because the LOL is detected for the lockstep processor pair **12** in this example, firmware **18** determines whether master processor **12**A is the system's boot processor. This can be determined, for example, by accessing the device tree **203** of FIG. **2** (described further below). The device tree may, in certain implementations, include a field for each processor indicating whether such processor is the system's boot processor. This field in the device tree may be set by the firmware during the boot-up process to identify the corresponding processor that is used as the system's boot processor or it may be pre-set by a system administrator (e.g., via a user interface), as examples. If determined in block **103** that the master processor **12**A is not the system's boot processor, then operation advances in this example to block **104**, and a lockstep recovery process is performed that does not utilize hot spare **15**.

In the specific example shown in FIG. **1**, if the lockstep is determined to be recoverable in block **101** and the processor for which the LOL is detected is determined in block **103** to not be the system boot processor, firmware **18** cooperates with OS **11** via standard OS methods to recover the lockstep. For instance, in the example embodiment of FIG. **1**, Advanced Configuration and Power Interface (ACPI) methods are used by firmware **18** to cooperate with OS **11**. Accordingly, no processor or platform specific knowledge is required to be embedded in OS **11**, but instead any ACPI-compatible OS may be used, including without limitation HP-UX and Open VMS operating systems. Example techniques that may be used for recovering from LOL for non-boot processors of a system are described further in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/973,076 titled "SYSTEM AND METHOD FOR PROVIDING FIRMWARE RECOVERABLE LOCKSTEP PROTECTION," the disclosure of which is incorporated herein by reference.

In the example embodiment of FIG. **1**, if determined in block **103** that the processor for which the LOL is detected is not the system boot processor, firmware **18** triggers OS **11** to idle the master processor **12**A in operational block **104**. In this embodiment, firmware **18** utilizes an ACPI method **105** to "eject" master processor **12**A, thereby triggering OS **11** to idle the master processor **12**A (i.e., stop scheduling tasks for the processor). Of course, by idling master processor **12**A, slave processor **12**B will in turn be idled. Thus, idling master processor **12**A results in idling the lockstep processor pair **12**. In this example embodiment, OS **11** is not aware of the presence of slave processor **12**B, but is instead aware of master processor **12**A. The interface of lockstep processor pair **12** to bus **16** manages copying to slave processor **12**B the instructions that are directed by OS **11** to master processor **12**A. Thus, firmware **18** need not direct OS **11** to eject slave processor **12**B, as OS **11** is not aware of such slave processor **12**B in this example implementation. Again, by idling master processor **12**A, slave processor **12**B is also idled as it merely receives copies of the instructions directed to master processor **12**A. Of course, if in a given implementation OS **11** is aware of slave **12**B, firmware **18** may be implemented to also direct OS **11** to idle such slave processor **12**B in a manner similar to that described for idling master processor **12**A.

Firmware **18** then attempts to recover lockstep for the lockstep processor pair **12** in block **106**. For instance, firmware **18** resets the processor pair **12**. During such reset of processor pair **12**, system **10** can continue to operate on its remaining available processors. Once the processor pair **12** is reset and lockstep is recovered, firmware **18** reintroduces master processor **12**A to OS **11** in operational block **107**. In this embodiment, firmware **18** updates the ACPI device table information for master processor **12**A to indicate that such master processor **12**A is "present, functioning and enabled." As discussed in the ACPI 2.0 specification for the _STA status method of a device, the _STA (status) object returns the status of a device, which can be one of the following: enabled, disabled, or removed. In this respect, in the result code (bitmap) bit **0** is set if the device is present; bit **1** is set if the device is enabled and decoding its resources; bit **2** is set if the device should be shown in the UI; bit **3** is set if the device is functioning properly (cleared if the device failed its diagnostics); bit **4** is set if the battery is present; and bits **5**-**31** are reserved. A device can only decode its hardware resources if both bits **0** and **1** are set. If the device is not present (bit **0** cleared) or not enabled (bit **1** cleared), then the device must not decode its resources. Bits **0**, **1** and **3** are the "present, enabled and functioning" bits mentioned above. Firmware **18** utilizes an ACPI method **108** to trigger OS **11** to "check for" master processor **12**A, thereby reintroducing the master processor **12**A to OS **11**. As a result of checking for master processor **12**A, OS **11**

will recognize that such master processor **12**A is again available and will thus begin scheduling tasks for master processor **12**A once again.

In view of the above, a recovery technique is provided that does not require utilizing a spare processor for recovering from LOL detected for non-boot processors. For various reasons, in certain system architectures problems arise in attempting to idle (or eject) the boot processor from the system. Thus, in certain embodiments, a hot spare processor is used for recovering from LOL for the system's boot processor. For instance, in the example of FIG. **1**, if determined in block **103** that the master processor **12**A is the system's boot processor, operation advances to block **109** whereat the state of the "good" processor in the lockstep processor pair **12** is copied over to each of processors **15**A and **15**B in the spare lockstep processor pair **15**. Exemplary techniques that may be utilized by firmware **18** for switching the role of boot processor to the hot spare are described further herein below.

Thus, operational block **109** essentially makes the spare processor pair **15** the system's boot processor, and then in certain embodiments firmware **18** resets the lockstep processor pair **12** and reestablishes its lockstep in block **110**. Once lockstep is reestablished for lockstep processor pair **12**, that pair is held in operational block **111** as a hot spare for the new boot processor pair **15**. That is, firmware **18** updates the device tree **203** of FIG. **2** to reflect that lockstep processor pair **12** is a hot spare for the boot processor pair **15**. Thus, should a recoverable LOL be detected for the new boot processor **15**, the above process may be used by firmware **18** to make the spare lockstep processor pair **12** the boot processor and then recover lockstep for pair **15**. An example technique for resetting the lockstep processor pair **12** and reestablishing its lockstep in which the lockstep processor pair **12** is then held as a hot spare is described further in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/972,796 (U.S. Published Patent Application 20060107106) titled "SYSTEM AND METHOD FOR MAINTAINING IN A MULTI-PROCESSOR SYSTEM A SPARE PROCESSOR THAT IS IN LOCKSTEP FOR USE IN RECOVERING FROM LOSS OF LOCKSTEP FOR ANOTHER PROCESSOR", the disclosure of which is hereby incorporated herein by reference.

FIG. **2** shows a block diagram of one embodiment of the above system **10**, which is implemented for the IPF processor architecture and is labeled as system $\mathbf{10}_A$. The quintessential model of the traditional IPF architecture is given in the *Intel IA-64 Architecture Software Developer's Manual, Volume 2: IA-64 System Architecture*, in section 11.1 *Firmware Model*, the disclosure of which is hereby incorporated herein by reference. Accordingly, in this example embodiment of system $\mathbf{10}_A$, firmware **18**, labeled as firmware $\mathbf{18}_A$, includes processor abstraction layer (PAL) **201** and platform/system abstraction layer (SAL) **202**. In general, PAL **201** is firmware provided by Intel for its processors, and SAL **202** is developed by an original equipment manufacturer (OEM) for the specific system/platform in which the processors are to be employed. PAL **201**, SAL **202**, as well as an extended firmware interface (EFI) layer (not shown), together provide, among other things, the processor and system initialization for an OS boot in an IPF system.

The boot-up process of a traditional IPF system, for example, proceeds as follows: When the system is first powered on, there are some sanity checks (e.g., power on self-test) that are performed by microprocessors included in the system platform, which are not the main system processors that run applications. After those checks have passed, power and clocks are given to a boot processor (which may, for example, be master processor **12**A). The boot processor begins executing code out of the system's Read-Only Memory (ROM) (not specifically shown in FIG. **2**). The code that executes is the PAL **201**, which gets control of system **10**. PAL **201** executes to acquire all of the processors in system $\mathbf{10}_A$ (recall that there may be many lockstep processor pairs, such as pairs **12** and **15**) such that the processors begin executing concurrently through the same firmware.

After it has performed its duty of initializing the processor(s), PAL **201** passes control of system $\mathbf{10}_A$ to SAL **202**. It is the responsibility of SAL **202** to discover what hardware is present on the system platform, and initialize it to make it available for the OS **11**. When main memory is initialized and functional, the firmware $\mathbf{18}_A$ is copied into the main memory. Then, control is passed to EFI (not shown), which is responsible for activating boot devices, which typically includes the disk. The EFI reads the disk to load a program into memory, typically referred to as an operating system loader. The EFI loads the OS loader into memory, and then passes it control of system $\mathbf{10}_A$ by branching the boot processor into the entry point of such OS loader program.

The OS loader program then uses the standard firmware interfaces to discover and initialize system $\mathbf{10}_A$ further for control. One of the things that the OS loader typically has to do in a multi-processor system is to retrieve control of the other processors (those processors other than the boot processor). For instance, at this point in a multi-processor system, the other processors may be executing in do-nothing loops. In an ACPI-compatible system, OS **11** makes ACPI calls to parse the ACPI tables to discover the other processors of a multi-processor system in a manner as is well-known in the art. Then OS **11** uses the firmware interfaces to cause those discovered processors to branch into the operating system code. At that point, OS **11** controls all of the processors and the firmware $\mathbf{18}_A$ is no longer in control of system $\mathbf{10}_A$.

As OS **11** is initializing, it has to discover from the firmware $\mathbf{18}_A$ what hardware is present at boot time. And in the ACPI standards, it also discovers what hardware is present or added or removed at run-time. Further, the supporting firmware (PAL, SAL, and EFI) are also used during system runtime to support the processor. For example, OS **11** may access a particular function of master processor **12**A via the supporting firmware $\mathbf{18}_A$, such as querying PAL **201** for the number, size, etc., of the processor's cache **14**A. Some other well-known firmware functions that OS **11** may employ during runtime include: (a) PAL **201** may be invoked to configure or change processor features such as disabling transaction queuing (PAL_BUS_SET_FEATURES); (b) PAL **201** may be invoked to flush processor caches (PAL_CACHE_FLUSH); (c) SAL **202** may be invoked to retrieve error logs following a system error (SAL_GET_STATE_INFO, SAL_CLEAR_STATE_INFO); (d) SAL **202** may be invoked as part of hot-plug sequences in which new I/O cards are installed into the hardware (SAL_PCI_CONFIG_READ, SAL_PCI_CONFIG_WRIT); (e) EFI may be invoked to change the boot device path for the next time the system reboots (SetVariable); (f) EFI may be invoked to change the clock/calendar hardware settings; and (g) EFI may be invoked to shutdown the system (ResetSystem).

A "device tree" is provided, which is shown as device tree **203** in this example. Device tree **203** is stored in SRAM (Scratch RAM) on the cell, which is RAM that is reinitialized. Firmware **18**A builds the device tree **203** as it discovers what hardware is installed in the system. Firmware then converts this information to the ACPI tables format and presents it to OS **11** so that OS **11** can know what is installed in the system. The ACPI device tables (not shown) are only consumed by

OS **11** at boot time, so they are never updated as things change. For OS **11** to find the current status, it calls an ACPI "method" to discover the "current status". The _STA method described above is an example of such an ACPI method. When _STA is called, the AML can look for properties on the device specified in the firmware device tree and convert that into the Result Code bitmap described above. So, if lockstep has been lost on a processor, firmware **15**A will set the device tree property that indicates loss of lockstep, then when OS **11** calls _STA for that device, the "lockstep lost" property directs the AML code to return to "0" in the "functioning properly" bit so that OS **11** can know there is a problem with that processor.

If a lost lockstep is recovered in accordance with the recovery technique described herein, firmware **18**A can indicate that lockstep has been recovered in the device tree **203**. Then when _STA is called on that device responsive to the OS receiving the "check for device" ACPI method, the present, enabled and functioning bits will all be set and OS **11** will know the CPU is safe to use.

Further, device tree **203** includes information designating which processor is the boot processor, as well as a spare processor for the boot processor. A simple example of device tree **203** is shown below in Table 1:

TABLE 1

Device	Status	Lockstep Enabled	Boot Processor?	Spare for Boot Processor?
Processor A	Present, Enabled, and Functioning	Yes	Yes	No
Processor B	N/A	Yes	No	Yes
Processor C	Present, Enabled, and Functioning	Yes	No	No
...	...	...	...	...

In the above example of Table 1, 3 processors "Processor A," "Processor B," and "Processor C" are identified in the device tree as being included in the multi-processor system. The status of each of processors Processor A and Processor B is identified as "Present, Enabled, and Functioning." In this example Processor B is a hot spare, so it is held in reserve and does not show up in the static ACPI tables that are built when the system boots. Accordingly the OS does not know about Processor B and does not have a device handle to call the _STA method on. As described above, the _STA method is a known ACPI method for checking the status of the corresponding device. Therefore, the status of Processor B is designated N/A (not applicable) in this example because the _STA method will not be called for this processor as long as it is the hot spare. Of course, upon such spare assuming the role of boot processor, its status is updated to "Present, Enabled, and Functioning." In other embodiments in which Processor B is an active spare, the OS will be aware of such processor and accordingly its status may be also indicated as "Present, Enabled, and Functioning."

Further, lockstep is identified in this example as enabled for these processors. In certain embodiments, the system administrator can designate, via user interface **20**, whether lockstep mode is to be enabled for the system's processors. Further, Processor A is designated as the system's boot processor in the example device tree shown in Table 1, and Processor B is designated as the spare processor. Thus, upon detection of LOL for one of the processors in the system, the firmware can access this device tree to determine if the processor for which LOL is detected is the system's boot processor; and if determined that the processor for which LOL is detected is the system's boot processor, the firmware can further determine from this device tree the designated spare processor to use in the lockstep recovery process.

In the example of FIG. **2**, a user interface **20** is included, which allows a user (e.g., system administrator) to designate the system's boot processor and spare processor. In certain embodiments, the user interface **20** further allows the user to designate other information, such as whether lockstep mode is enabled for the processors.

One example system in which firmware $\mathbf{18}_A$ of FIG. **2** may be employed is shown in FIG. **3**. FIG. **3** shows system **30** that includes multi-processor cells A-D, labeled $\mathbf{31}_A$-$\mathbf{31}_D$, respectively. Cell A $\mathbf{31}_A$ is shown in more detail, and should be understood that cells B-D $\mathbf{31}_B$-$\mathbf{31}_D$ have substantially the same architecture as that of cell A $\mathbf{31}_A$ in this example. The architecture of cell A $\mathbf{31}_A$ described hereafter corresponds to that of Hewlett-Packard's Superdome™ systems. As shown, multi-processor cell A $\mathbf{31}_A$ includes coherency controller **304** that is communicatively coupled to two buses, labeled **305** and **306**, respectively. A plurality of processor modules are included within cell A $\mathbf{31}_A$, shown as CPU **0**, CPU **1**, CPU **2** and CPU **3** (and labeled **300**-**303**, respectively). More particularly, a plurality of processors modules are communicatively coupled to each of the buses **305** and **306**. In this example, CPU **0** (**300**) and CPU **1** (**301**) are each communicatively coupled to the first bus **305**, and CPU **2** (**302**) and CPU **3** (**303**) are each communicatively coupled to the second bus **306**. It should be understood that each of CPU **0** (**300**), CPU **1** (**301**), CPU **2** (**302**), and CPU **3** (**303**) is in actuality a lockstep processor pair, such as the lockstep processor pair **12** of FIGS. **1**-**3**. Thus, for instance, CPU **0** (**300**) includes both a master processor and a slave processor, as with processors **12**A and **12**B included in the lockstep processor pair **12** of FIGS. **1**-**2**.

Coherency controller **304** decodes the address an interrupt is targeted towards and determines which bus (**305** or **306**) the interrupt packet should be delivered to. Each CPU on the destination bus sees the interrupt and compares the interrupt target address with its internal "LID" register to determine if it is the targeted CPU. If the match is correct, the CPU responds to the interrupt. If the match is not correct, the CPU ignores the interrupt packet.

As shown, the example firmware $\mathbf{18}_A$ described above with FIGS. **1**-**2** may be employed to manage the recovery from detected LOLs for the processors of cells A-D ($\mathbf{31}_A$-$\mathbf{31}_D$). In this example architecture, when designating a spare for the boot processor, such spare is selected to be another processor communicatively coupled to the same bus as the boot processor. For instance, suppose that during the boot-up of system **30**, CPU **2** (**302**) is designated as the boot processor. Accordingly, firmware $\mathbf{18}^A$ will identify another processor that is communicatively coupled to bus **306** to designate as a spare for such boot processor, such as CPU **3** (**303**). During runtime if a LOL is detected for CPU **2** (**302**), then firmware $\mathbf{18}_A$ may operate according to the process described above in FIG. **1** to switch the designation of boot processor to a designated spare, CPU **3** (**303**). In performing this switch the LID register of the CPU that lost lockstep is copied into the LID register of the spare CPU and the spare CPU pair will then assume the ID of the failed CPU and start responding to interrupts directed to that ID. The LID of the failed CPU is quickly re-written to a different value so that it will no longer respond to interrupts directed to that ID. CPU **2** (**302**) can then be reset to recover its lockstep, and such CPU **2** (**302**) may then be designated as a spare for the new boot processor CPU **3** (**303**).

It should be recognized that in the example system of FIG. **3**, if a hot spare were used for recovering from LOL for all

processors, rather than just for the boot processor, much of the potential processing resources would be required to be held in reserve as spare resources. For instance, a hot spare would be required on each of the buses of each cell. Because there are two processors coupled to each bus in this example (i.e., CPU **0** and CPU **1** coupled to bus **305**, and CPU **2** and CPU **3** coupled to bus **306**), reserving a hot spare processor for each bus would result in holding half of the total processing resources in reserve. This is an undesirably expensive and wasteful solution. Thus, according to the illustrated embodiment, a hot spare processor is reserved only for the boot processor. Thus, continuing with the above example in which CPU **2** is the boot processor and CPU **3** is held as a "hot" spare, both of CPU **0** and CPU **1** are active (i.e., neither are held in spare). Further, none of the processors of cells B-D need to be held in reserve as hot spares. In the event of a recoverable LOL occurring for any of non-boot processors of the cells, the lockstep recovery process of FIG. **1** (i.e., operational blocks **104-108**) can be utilized without requiring a spare for lockstep recovery for those non-boot processors.

While in the above example embodiment, a hot spare CPU is maintained for recovery from LOL for the system's boot processor, in an alternative embodiment a hot spare need not be held as such, but rather a processor can be dynamically made a "spare" when needed for recovery from LOL encountered for the system boot processor. That is, assuming as in the above example of FIG. **3** that CPU **2** is the system's boot processor, CPU **3** does not have to be held in reserve (idled) as a hot spare for CPU **2**, but instead CPU **3** can be active and dynamically idled and made a "spare" upon detection of LOL in CPU **2**. Accordingly, CPU **3** may be designated as a spare for the boot processor, but such CPU **3** need not be held in reserve during normal system operation. Instead, the CPU **3** may be dynamically (during system run-time) transformed into a spare processor (e.g., by causing the OS to idle such processor) in the event that a LOL is detected for the boot processor.

Thus, the resource of this spare CPU need not be wasted during normal runtime, but can be dynamically turned into a spare processor that is available for recovering from LOL encountered on CPU **2** in the manner described above. As an example of this embodiment, the firmware may send an "eject request" on a healthy processor pair (e.g., CPU **3**), and lie to the OS by indicating (e.g., in response to an ACPI _STA method for the healthy processor pair) that such processor pair is not functioning. This would cause the OS to eject that processor pair as though it had encountered a LOL. This processor would become the "hot spare" and be used to replace the boot processor. The system boot processor role would be assumed by the idled healthy processor pair, just as described above in the example in which the healthy CPU **3** is held idle for the system boot processor, CPU **2**. The only difference being that now the firmware had to take some initial action to turn the healthy processor pair into a "hot spare" by idling it and making it appear to the OS as though it was not functioning. Thus, the LIDs would be swapped. That is, the LID register of the boot processor that lost lockstep is copied into the LID register of the now spare processor pair (CPU **3** in this example), and the LID register of this spare processor pair would be copied into the LID register of the original boot processor. Lockstep is then reestablished on the original boot processor, and it can be reintroduced to the OS with the ID of the original non-boot processor that replaced it as the system boot processor. The processor that was formerly the boot processor may be designated, in device tree **203**, as the spare for the now boot processor.

Turning to FIG. **4**, an exemplary operational flow diagram for switching the role of boot processor to a hot spare processor according to one embodiment is shown. In this case, an LOL has been detected for the boot processor. In operational block **401**, the system firmware allocates space to save the status of the good processor of the lockstep pair of processors for which LOL was detected. Exemplary techniques for determining which of the lockstep pair of processors had the error are disclosed in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/972,835 (U.S. Published Patent Application 20060107114) titled "SYSTEM AND METHOD FOR USING INFORMATION RELATING TO A DETECTED LOSS OF LOCKSTEP FOR DETERMINING A RESPONSIVE ACTION," the disclosure of which is hereby incorporated herein by reference. Accordingly, an appropriate amount of data storage space for storing the cache of the "good" processor in the lockstep pair of processors is allocated in block **401**. For instance, an amount of SRAM may be allocated for storing the registers from the good processor.

In operational block **402**, the state of the good processor is stored to the allocated storage space. For instance, in this illustrative example, PAL is called to dump the processor state of the good processor of the current boot processor (CBP) module (for which LOL was detected) to the allocated storage space.

In operational block **403**, the LID of the current boot processor (for which LOL was detected) is reprogrammed to a different value. For instance, the current boot processor may be assigned a LID that is reserved just for this situation (i.e., for switching the role of boot processor). Accordingly, the boot processor for which LOL was detected will not respond to the interrupts addressed for its original LID because the boot processor's LID value is reprogrammed to a different value.

In operational block **404**, the state of the good processor that was stored in block **402** to the allocated storage space is stored to the hot spare processor. In this example, the PAL is called to "Restore Processor State" on the hot spare processor, wherein the PAL is provided a pointer to the address to which the good processor's state was stored in block **402**. Accordingly, all the values from that storage space will be read into the registers of the spare processor. As a result, the LID value of the spare processor is programmed with the value of the boot processor that was saved to the allocated storage space. Accordingly, the spare processor automatically, as a result of receiving this stored state, receives the Local ID corresponding to the boot processor for interrupt handling.

In operational block **405**, the system firmware updates its device tree information to reflect that the hot spare is assuming the role of boot processor. The system firmware further updates its device tree, in this example, to indicate that the former boot processor (for which LOL was detected) is assuming the role of hot spare for the new boot processor.

In operational block **406**, PAL_MC_RESUME (which is a standard PAL call for resuming normal execution after a machine check (MC)) is called for the hot spare (which is assuming the role of boot processor) to cause the OS to recognize the hot spare processor as available for processing instructions, wherein the OS recognizes the hot spare as the boot processor. Thus, the hot spare has now assumed the role of the boot processor, taking the place of the former boot processor for which LOL was detected. Accordingly, in operational block **407** the former hot spare (which has assumed the role of boot processor) resumes normal operation as the system's boot processor with its lockstep protection enabled. Thus, in this exemplary embodiment, upon

detection of LOL for a boot processor, the role of boot processor is switched to a hot spare that has lockstep protection, wherein the system effectively recovers lockstep for its boot processor without requiring a system shut down (i.e., without shutting down the OS). In certain implementations, the firmware may go on to take action to attempt to reestablish lockstep for the former boot processor (e.g., by resetting the former boot processor), and if successful in reestablishing its lockstep, such former boot processor may be held in reserve as a hot spare processor for the new boot processor in the event that LOL is detected for this new boot processor.

Turning to FIG. **5**, an exemplary operational flow diagram for switching the role of boot processor to an active spare processor according to one embodiment is shown. Again, an LOL has been detected for the boot processor. In operational block **501**, the system firmware, responsive to such detection of LOL for the boot processor, examines the device tree to determine which processor module is assigned the role of active spare. In operational block **502** the system firmware (e.g., SAL) asserts a General Purpose Event Interrupt (GPE) by writing to the appropriate GPE register in the ACPI register space. This causes the OS to execute its interrupt handler, which will run. an AML Method associated with this particular interrupt. In this example, this AML method that is executed responsive to the interrupt generated by the GPE includes an AML instruction called "Notify". This "Notify" operation indicates the determined active spare processor object and that an Eject is being requested as arguments. Accordingly, an AML method is used to request the OS to eject the active spare that was determined in block **501**. In response, the OS idles the active spare processor and returns control over the active spare processor to the system firmware, in block **503**.

In operational block **504**, the system firmware allocates space to save the status of the good processor of the lockstep pair of processors for which LOL was detected (i.e., the lockstep pair of processors of the CBP). Exemplary techniques for determining which of the lockstep pair of processors had the error are disclosed in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/972,835 (U.S. Published Patent Application No. 20060107114) titled "SYSTEM AND METHOD FOR USING INFORMATION RELATING TO A DETECTED LOSS OF LOCKSTEP FOR DETERMINING A RESPONSIVE ACTION," the disclosure of which is hereby incorporated herein by reference. Accordingly, an appropriate amount of data storage space for storing the cache of the "good" processor in the lockstep pair of processors is allocated in block **504**. For instance, an amount of SRAM may be allocated for storing the registers from the good processor.

In operational block **505**, the state of the good processor is stored to the allocated storage space. For instance, in this illustrative example, PAL is called to dump the processor state of the good processor of the current boot processor (CBP) module (for which LOL was detected) to the allocated storage space.

In operational block **506**, the LID of the current boot processor (for which LOL was detected) is reprogrammed to a different value. For instance, the current boot processor may be assigned a LID that is reserved just for this situation (i.e., for switching the role of boot processor). Accordingly, the boot processor for which LOL was detected will not respond to the interrupts addressed for its original LID because the boot processor's LID value is reprogrammed to a different value.

In operational block **507**, the state of the good processor that was stored in block **505** to the allocated storage space is stored to the idled spare processor. In this example, the PAL is called to "Restore Processor State" on the active spare processor, wherein the PAL is provided a pointer to the address to which the good processor's state was stored in block **505**. Accordingly, all the values from that storage space will be read into the registers of the spare processor. As a result, the LID value of the spare processor is programmed with the value of the boot processor that was saved to the allocated storage space. Accordingly, the spare processor automatically, as a result of receiving this stored state, receives the Local ID (LID) corresponding to the boot processor for interrupt handling.

In operational block **508**, the system firmware updates its device tree information to reflect that the active spare is assuming the role of boot processor. The system firmware further updates its device tree, in this example, to indicate that the former boot processor (for which LOL was detected) is assuming the role of active spare for the new boot processor.

In operational block **509**, PAL_MC_RESUME (which is a standard PAL call for resuming normal execution after a machine check (MC)) is called for the active spare (which is assuming the role of boot processor) to cause the OS to recognize the active spare processor as available for processing instructions, wherein the OS recognizes the active spare as the boot processor. Thus, the active spare has now assumed the role of the boot processor, taking the place of the former boot processor for which LOL was detected. Accordingly, in operational block **510** the former active spare (which has assumed the role of boot processor) resumes normal operation as the system's boot processor with its lockstep protection enabled. Thus, in this exemplary embodiment, upon detection of LOL for a boot processor, an active spare processor is idled, and the role of boot processor is switched to such idled spare that has lockstep protection, wherein the system effectively recovers lockstep for its boot processor without requiring a system shut down (i.e., without shutting down the OS). In certain implementations, the firmware may go on to take action to attempt to reestablish lockstep for the former boot processor (e.g., by resetting the former boot processor), and if successful in reestablishing its lockstep, such former boot processor may be reintroduced to the OS as an active spare processor for the new boot processor in the event that LOL is detected for this new boot processor.

The exemplary procedures of FIGS. **4** and **5** switch the role of boot processor to a spare processor in accordance with certain embodiments. While exemplary operational flows are provided in FIGS. **4** and **5**, embodiments hereof are not limited to those illustrative examples. For instance, FIG. **6** shows a more general operational flow diagram for certain embodiments. In operational block **601**, LOL is detected for a processor in a multi-processor system. In operational block **602**, the system (e.g., system firmware) determines whether the processor for which LOL is detected is the system's boot processor. As described above, this is determined in certain embodiments by accessing the node of the device tree corresponding to the processor for which LOL is detected and determining if the information in that node of the device tree indicates that the processor is the system's boot processor.

In operational block **603**, responsive to determining that the processor for which LOL is detected is the system's boot processor, the system's firmware switches the role of boot processor to a spare processor, which may be either a hot spare or an active spare, as described above. Accordingly, the system's OS recognizes the spare processor as the system boot processor. In view of the above, embodiments hereof enable the role of boot processor to be dynamically switched, during system runtime, to a spare processor responsive to a

detected LOL for the boot processor, thus enabling recovery from the LOL without requiring that the system be crashed or re-booted.

What is claimed is:

1. A method comprising:

detecting loss of lockstep (LOL) for a processor in a multi-processor system;

determining that the processor for which said LOL is detected is assigned a role of boot processor;

changing a system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected, to a different value, which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor; and

changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to a spare processor without shutting down the system's operating system.

2. The method of claim **1** wherein the spare processor is a hot spare that is held in idle until switched the role of boot processor.

3. The method of claim **1** wherein the spare processor is an active spare.

4. The method of claim **3** wherein said switching further comprises:

causing the system's operating system to idle the active spare.

5. The method of claim **1** wherein the spare processor is a processor module that includes a master and slave processors that operate in lockstep.

6. The method of claim **1** wherein said determining that the processor for which said LOL is detected is assigned the role of boot processor comprises:

accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor.

7. The method of claim **1** further comprising:

determining the processor that is assigned the role of spare processor.

8. The method of claim **7** wherein said determining the processor that is assigned the role of spare processor comprises:

accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of spare processor.

9. The method of claim **1** wherein said determining step is performed by system firmware.

10. The method of claim **1** wherein said switching step is performed by system firmware.

11. The method of claim **1** wherein the processor for which said LOL is detected includes a lockstep pair of processors, and wherein said switching comprises:

determining one of the lockstep pair of processors that is not the cause of said LOL; and

copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to said spare processor.

12. The method of claim **11** wherein said spare processor includes a lockstep pair of processors, and wherein said copying comprises:

copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to each of said lockstep pair of processors of said spare processor.

13. A method comprising:

system firmware determining that loss of lockstep (LOL) is detected for a lockstep pair of processors that are assigned the role of boot processor in a system;

determining one of said lockstep pair of processors that is not the cause of said LOL;

copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to a spare processor;

changing a system identifier, used by the system's operating system to recognize the boot processor, of said lockstep pair of processors that are assigned the role of boot processor, to a different value, which causes the s system's operating system to not recognize the lock step pair of processors for which said LOL is detected as the processor; and

changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to said spare processor.

14. The method of claim **13** wherein the spare processor is a hot spare that is held in idle until switched the role of boot processor.

15. The method of claim **13** wherein the spare processor is an active spare.

16. The method of claim **15** further comprising:

causing the system's operating system to idle the active spare.

17. The method of claim **13** wherein said switching causes the system's operating system to recognize said spare processor as said boot processor without shutting down said operating system.

18. The method of claim **13** wherein said determining step is performed by system firmware.

19. The method of claim **13** wherein said copying step is performed by system firmware.

20. The method of claim **13** wherein said switching step is performed by system firmware.

21. A system comprising:

a plurality of processor modules that each include a master processor and a slave processor that operate in lockstep;

an operating system;

error detection logic operable to detect loss of lockstep (LOL) for at least one of said processor modules; and

system firmware operable, responsive to the detection of LOL for a first of said processor modules, to determine whether said first processor module is assigned a role of system boot processor;

wherein if determined that said first processor module is assigned the role of system boot processor, said system firmware is further operable to cause said operating system to recognize another processor module as system boot processor without shutting down said operating system.

22. The system of claim **21** wherein the another processor module is a processor module assigned a role of spare processor.

23. The system of claim **21** wherein the another processor module is held in idle until said system firmware causes said operating system to recognize said another processor module as said system boot processor.

24. The system of claim **21** wherein the system firmware causes the operating system to idle the another processor module before causing said operating system to recognize said another processor module as said system boot processor.

25. The system of claim 21 further comprising:

non-volatile data storage storing information that assigns the role of system boot processor to one of said plurality of processor modules.

26. A system comprising:

system firmware means for detecting loss of lockstep (LOL) for a processor in a multi-processor system;

means for determining whether the processor for which said LOL is detected is assigned the role of boot processor; and

system firmware means for switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which said LOL is detected is assigned the role of boot processor.

27. The system of claim 26 wherein said processor in said multi-processor system is an Itanium Processor Family (IPF) processor.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 7,624,302 B2
APPLICATION NO. : 10/972588
DATED : November 24, 2009
INVENTOR(S) : Scott L. Michaelis et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

In column 20, line 13, in Claim 13, after "the" delete "s".

In column 20, line 16, in Claim 13, before "processor;" insert -- boot --.

In column 21, line 8, in Claim 26, before "means" insert -- system firmware --.

Signed and Sealed this

Sixteenth Day of February, 2010

David J. Kappos

David J. Kappos
Director of the United States Patent and Trademark Office

US007328371B1

(12) **United States Patent**
Kalyanasundharam et al.

(10) **Patent No.: US 7,328,371 B1**
(45) **Date of Patent: Feb. 5, 2008**

(54) **CORE REDUNDANCY IN A CHIP MULTIPROCESSOR FOR HIGHLY RELIABLE SYSTEMS**

(75) Inventors: **Vydhyanathan Kalyanasundharam**, San Jose, CA (US); **William A. Hughes**, San Jose, CA (US); **Philip E. Madrid**, Round Rock, TX (US); **Scott A. White**, Austin, TX (US); **Ajay Naini**, Saratoga, CA (US)

(73) Assignee: **Advanced Micro Devices, Inc.**, Sunnyvale, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 557 days.

(21) Appl. No.: **10/966,466**

(22) Filed: **Oct. 15, 2004**

(51) **Int. Cl.**
G06F 11/00 (2006.01)

(52) **U.S. Cl.** **714/11**; 712/22; 712/34

(58) **Field of Classification Search** 714/11; 712/22, 34
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,799,022	A	8/1998	Williams	
5,903,717	A *	5/1999	Wardrop	714/12
6,393,582	B1 *	5/2002	Klecka et al.	714/11
6,571,363	B1	5/2003	Steiss	
6,862,693	B2 *	3/2005	Chaudhry et al.	714/11
6,928,583	B2 *	8/2005	Griffin et al.	714/11
7,003,691	B2 *	2/2006	Safford et al.	714/11
7,085,959	B2 *	8/2006	Safford	714/11
2002/0144177	A1 *	10/2002	Kondo et al.	714/13
2005/0273653	A1 *	12/2005	Zubkow	714/11
2005/0278567	A1 *	12/2005	Wolfe et al.	714/11
2006/0248384	A1 *	11/2006	Safford	714/11

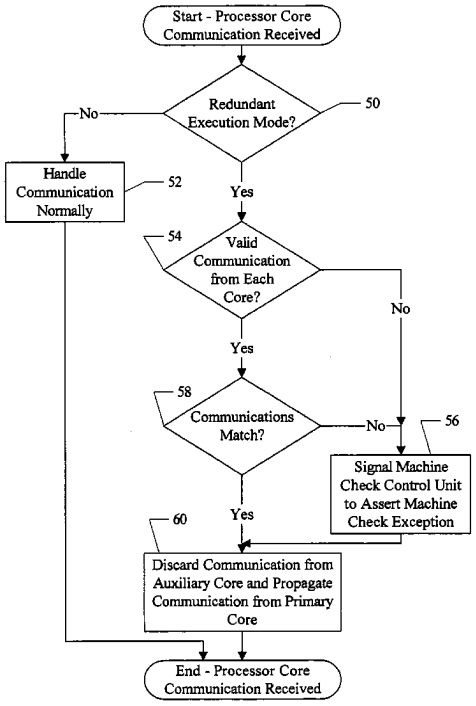
* cited by examiner

Primary Examiner—Christopher McCarthy
(74) *Attorney, Agent, or Firm*—Lawrence J. Merkel; Meyertons, Hood, Kivlin, Kowert & Goetzel, P.C.

(57) **ABSTRACT**

In one embodiment, a node comprises a plurality of processor cores and a node controller coupled to the processor cores. The node controller is configured to route communications from the processor cores to other devices in a computer system. The node controller comprises a circuit coupled to receive the communications from the processor cores. In a redundant execution mode in which at least a first processor core is redundantly executing code that a second processor core is also executing, the circuit is configured to compare communications from the first processor core to communications from the second processor core to verify correct execution of the code. In some embodiments, the processor cores and the node controller may be integrated onto a single integrated circuit chip as a CMP. A similar method is also contemplated.

19 Claims, 7 Drawing Sheets



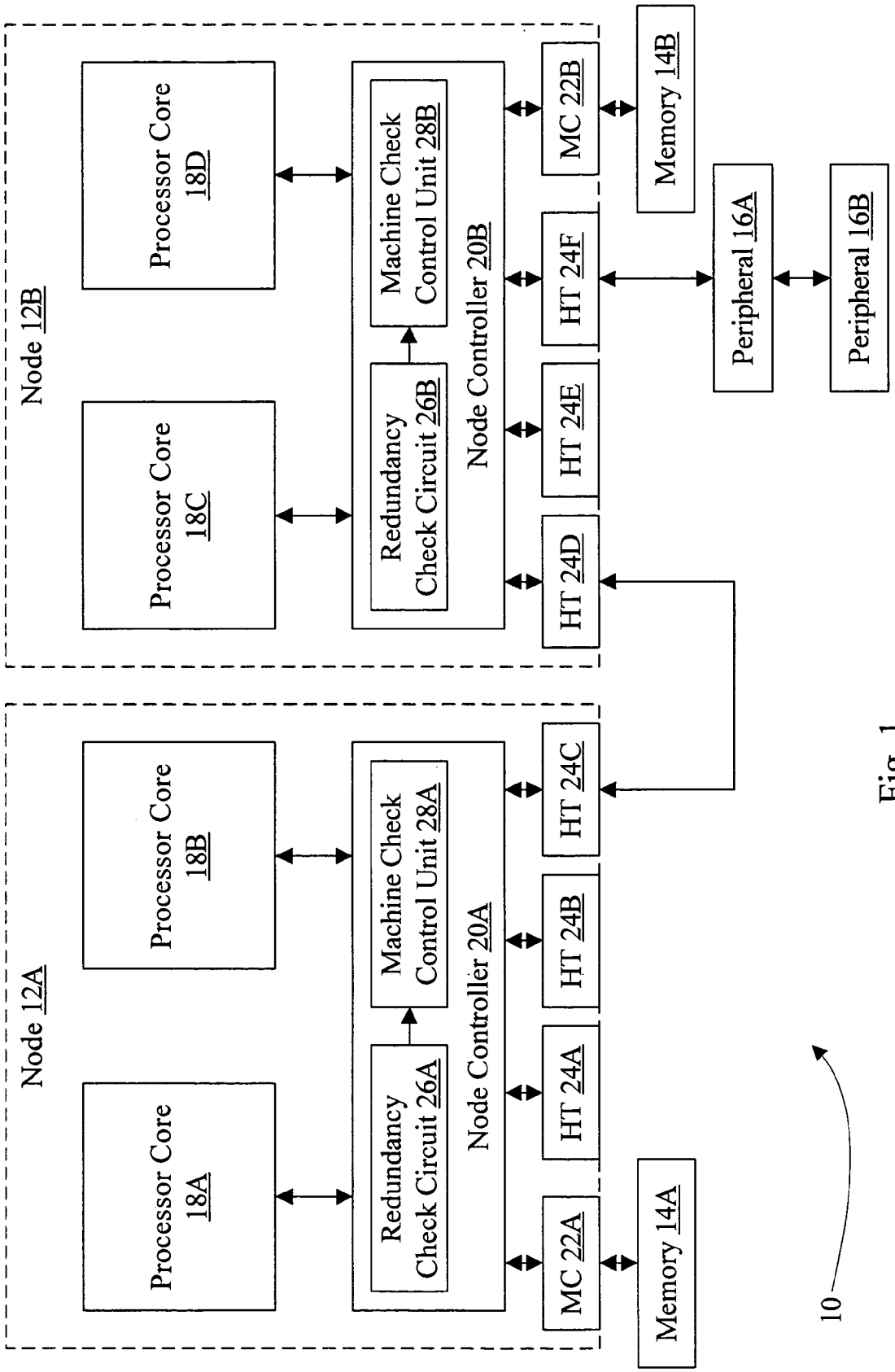

Node 12A
Processor Core 18A
Processor Core 18B
Redundancy Check Circuit 26A
Machine Check Control Unit 28A
Node Controller 20A
MC 22A
HT 24A
HT 24B
HT 24C
Memory 14A
Node 12B
Processor Core 18C
Processor Core 18D
Redundancy Check Circuit 26B
Machine Check Control Unit 28B
Node Controller 20B
HT 24D
HT 24E
HT 24F
MC 22B
Memory 14B
Peripheral 16A
Peripheral 16B
10

Fig. 1

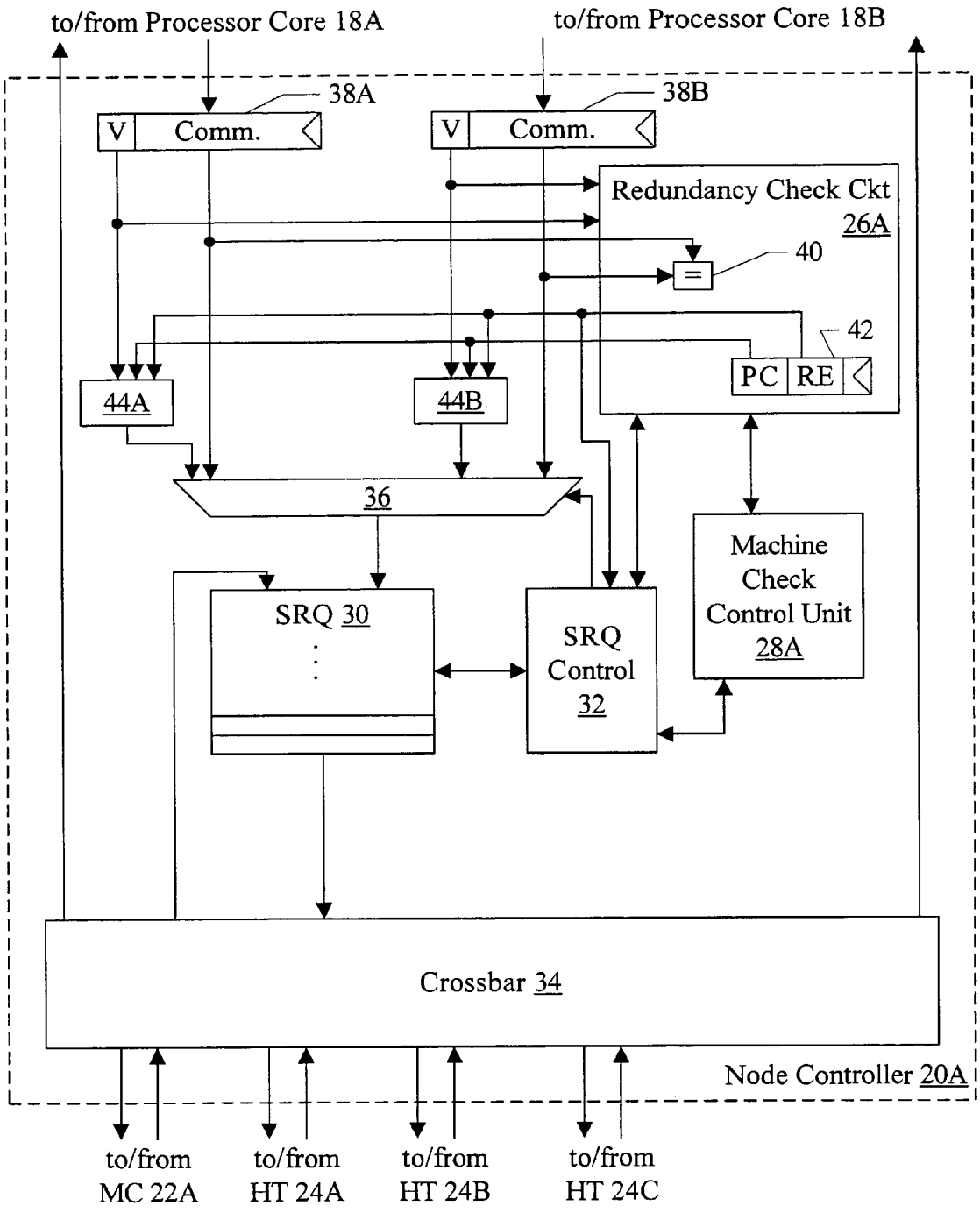

to/from Processor Core 18A
to/from Processor Core 18B
38A
38B
V
Comm.
V
Comm.
Redundancy Check Ckt
26A
40
=
42
PC
RE
44A
44B
36
Machine
Check
Control Unit
28A
SRQ 30
SRQ
Control
32
Crossbar 34
Node Controller 20A
to/from
MC 22A
to/from
HT 24A
to/from
HT 24B
to/from
HT 24C

Fig. 2

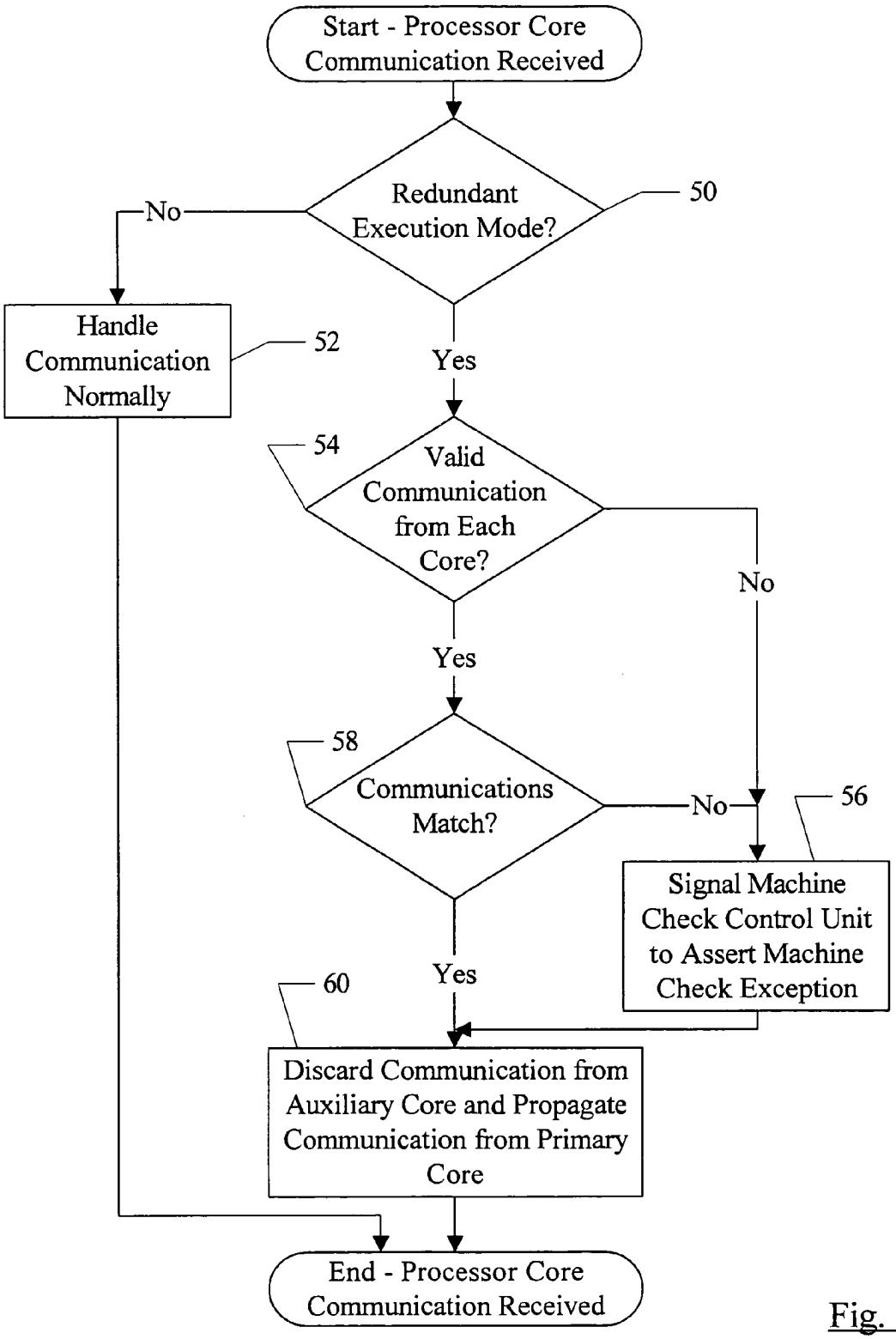

Start - Processor Core Communication Received
Redundant Execution Mode?
50
No
Handle Communication Normally
52
Yes
Valid Communication from Each Core?
54
No
Yes
Communications Match?
58
No
56
Signal Machine Check Control Unit to Assert Machine Check Exception
Yes
60
Discard Communication from Auxiliary Core and Propagate Communication from Primary Core
End - Processor Core Communication Received

Fig. 3

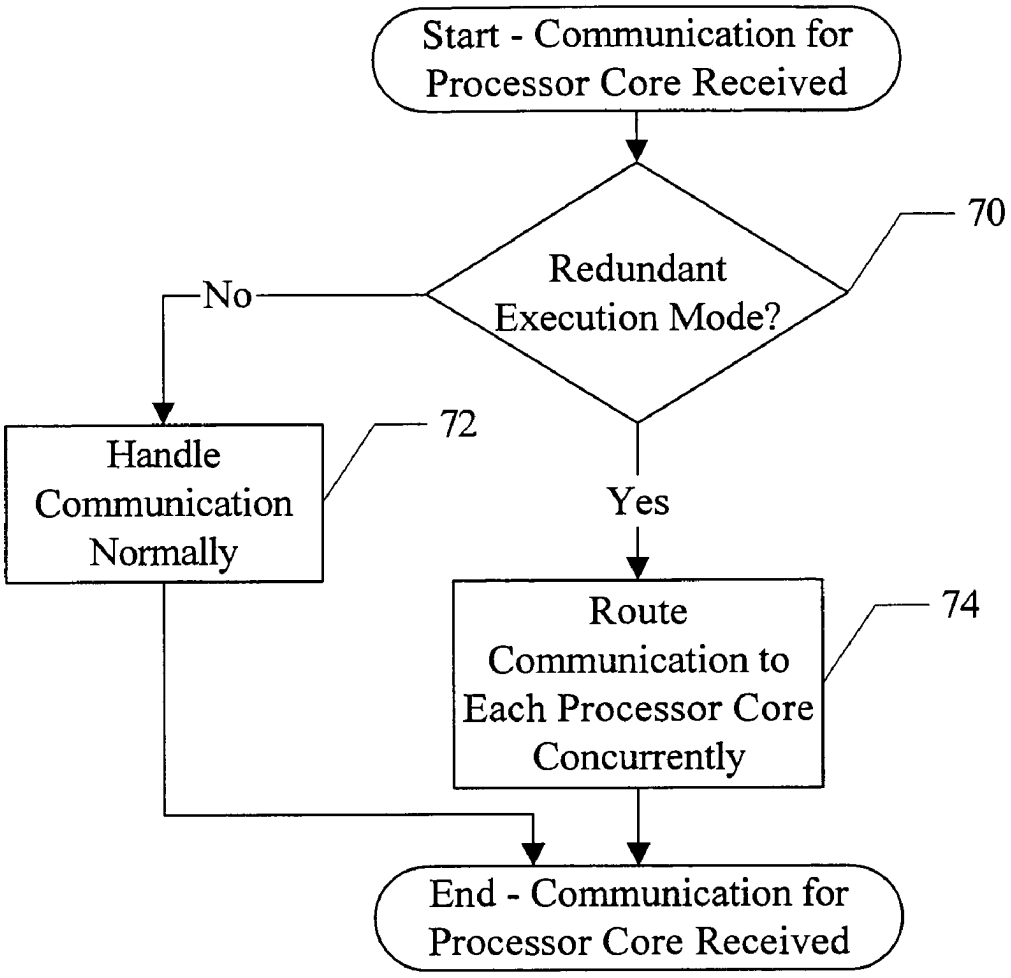

Start - Communication for Processor Core Received
Redundant Execution Mode?
70
No
Yes
Handle Communication Normally
72
Route Communication to Each Processor Core Concurrently
74
End - Communication for Processor Core Received

Fig. 4

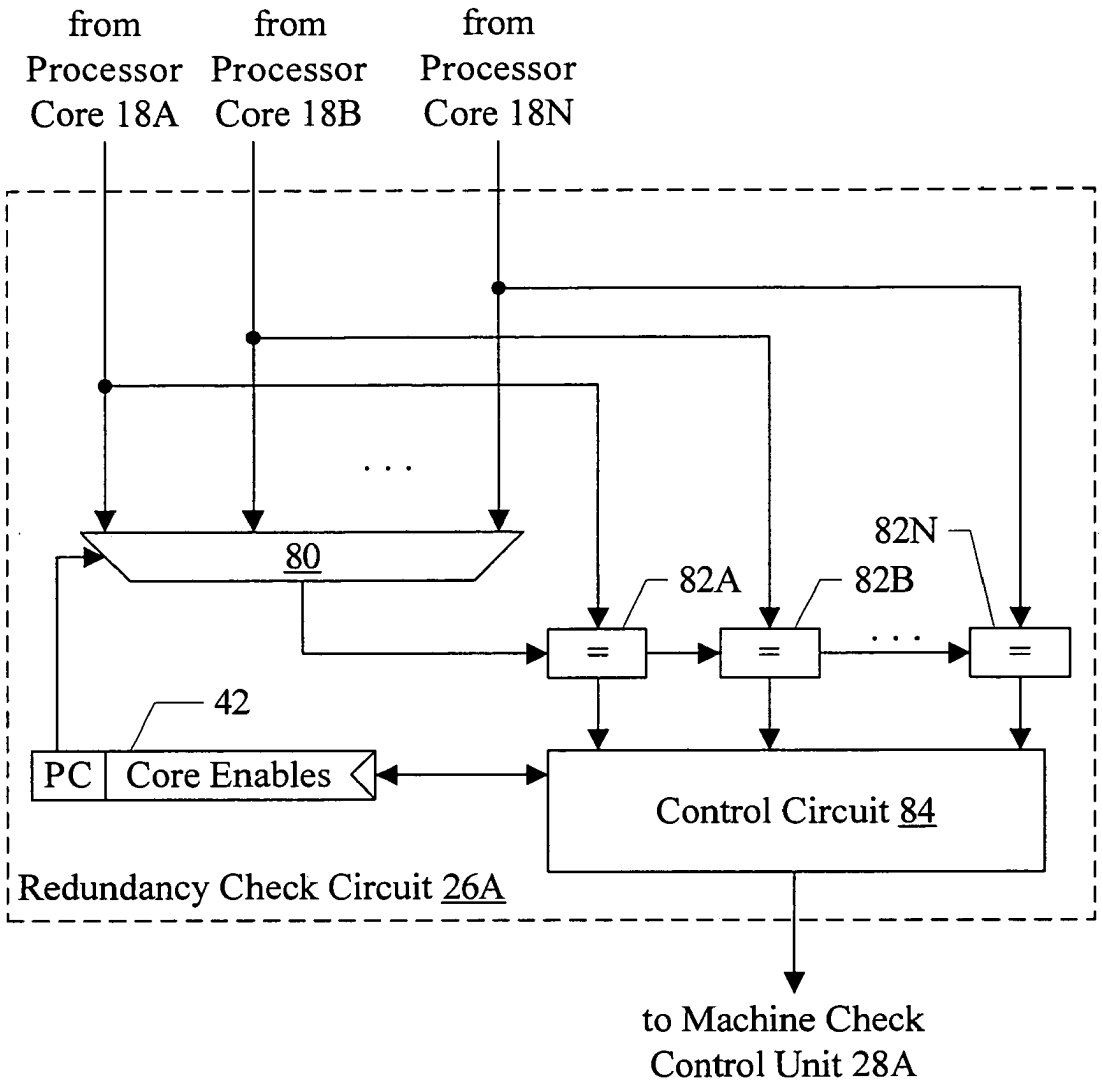

from Processor Core 18A
from Processor Core 18B
from Processor Core 18N
80
82A
82B
82N
=
=
=
42
PC
Core Enables
Control Circuit 84
Redundancy Check Circuit 26A
to Machine Check Control Unit 28A

Fig. 5

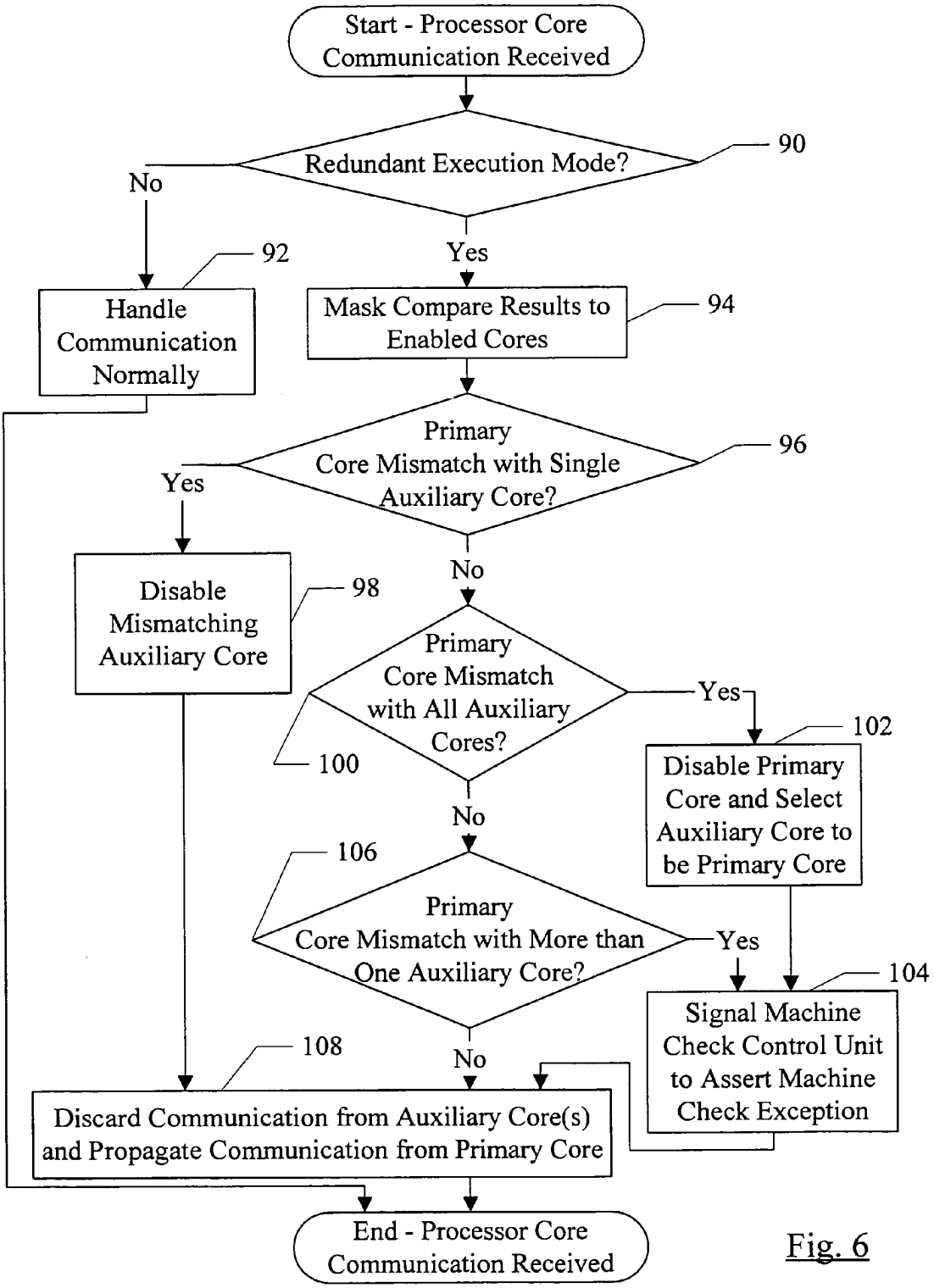

Start - Processor Core Communication Received
Redundant Execution Mode?
90
No
Yes
92
Handle Communication Normally
Mask Compare Results to Enabled Cores
94
Primary Core Mismatch with Single Auxiliary Core?
96
Yes
No
Disable Mismatching Auxiliary Core
98
Primary Core Mismatch with All Auxiliary Cores?
Yes
100
102
Disable Primary Core and Select Auxiliary Core to be Primary Core
No
106
Primary Core Mismatch with More than One Auxiliary Core?
Yes
104
Signal Machine Check Control Unit to Assert Machine Check Exception
108
No
Discard Communication from Auxiliary Core(s) and Propagate Communication from Primary Core
End - Processor Core Communication Received

Fig. 6

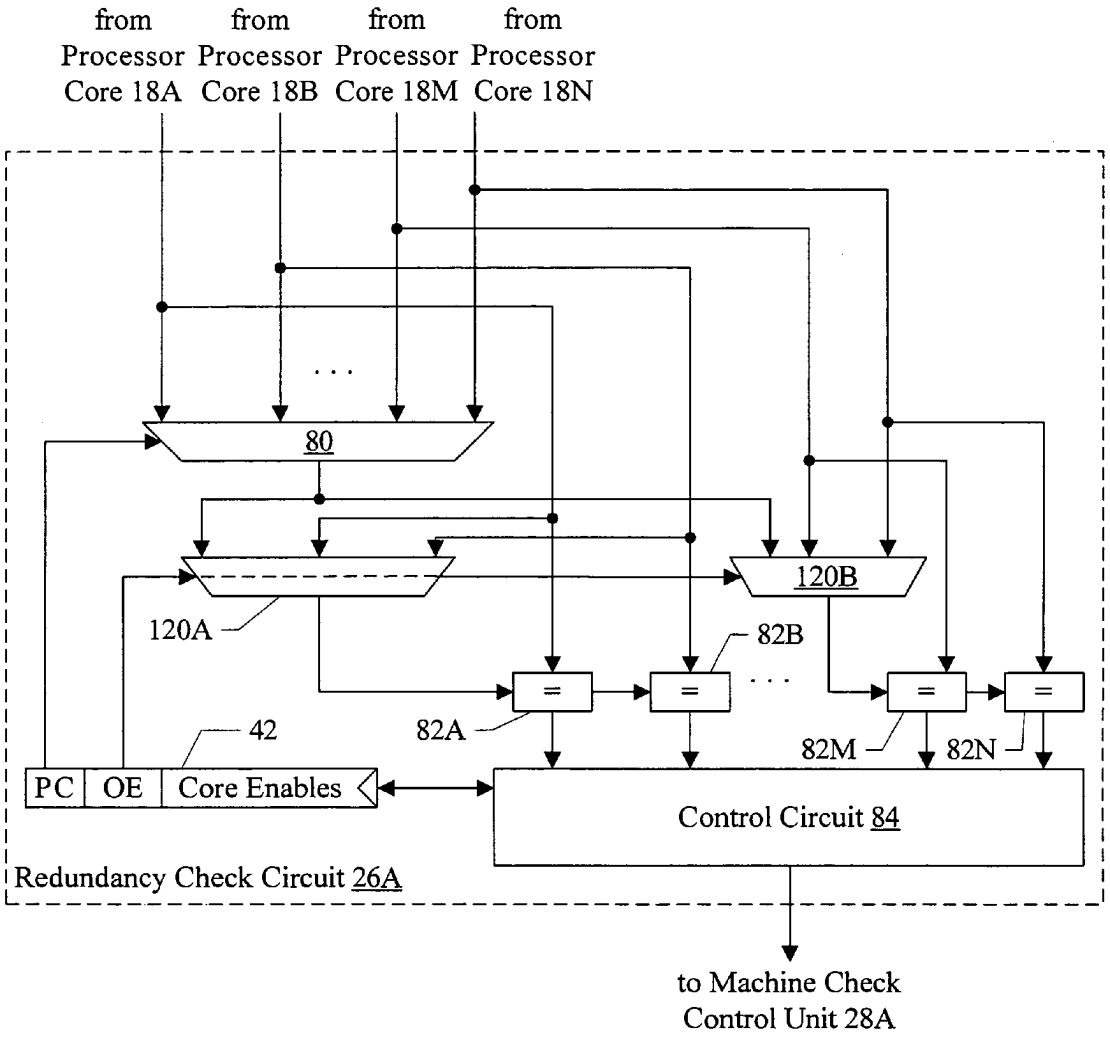

from Processor Core 18A
from Processor Core 18B
from Processor Core 18M
from Processor Core 18N
80
120A
120B
82B
82A
82M
82N
42
PC
OE
Core Enables
Control Circuit 84
Redundancy Check Circuit 26A
to Machine Check Control Unit 28A

Fig. 7

CORE REDUNDANCY IN A CHIP MULTIPROCESSOR FOR HIGHLY RELIABLE SYSTEMS

BACKGROUND

1. Field of the Invention

This invention is related to the field of multiprocessors and, more particularly, to chip multiprocessors (CMPs) and implementing redundant execution for high reliability in CMPs.

2. Description of the Related Art

Fault-tolerant computer systems are used in a variety of applications in the computing world. For example, many servers are implemented on fault-tolerant systems so that the servers may be available as close to 24 hours a day as possible. Any mission-critical computing functions may benefit from being assigned to fault-tolerant computer systems.

One of the aspects of fault-tolerant computer systems is high reliability through the early detection of errors in operation. For example, some fault-tolerant systems may implement high reliability by including multiple processors and having them redundantly execute the same software code. The results of executing the code can be compared, and any discrepancies may be detected as an error. The processors are discrete, individual integrated circuits, and thus such a solution is costly both in terms of the number of integrated circuits needed and the circuit board space required to connect the processors to the rest of the computer system. Additionally, since the processors are discrete, controlling the processors to execute in lockstep or to otherwise synchronize is complex and error prone, and typically requires additional circuitry to be included in the computer system (external from the processors) to carry out.

SUMMARY

In one embodiment, a node comprises a plurality of processor cores and a node controller coupled to the processor cores. The node controller is configured to route communications from the processor cores to other devices in a computer system. The node controller comprises a circuit coupled to receive the communications from the processor cores. In a redundant execution mode in which at least a first processor core is redundantly executing code that a second processor core is also executing, the circuit is configured to compare communications from the first processor core to communications from the second processor core to verify correct execution of the code. In some embodiments, the processor cores and the node controller may be integrated onto a single integrated circuit chip as a CMP. A similar method is also contemplated.

BRIEF DESCRIPTION OF THE DRAWINGS

The following detailed description makes reference to the accompanying drawings, which are now briefly described.

FIG. **1** is a block diagram of one embodiment of a computer system.

FIG. **2** is a block diagram of one embodiment of a node controller shown in FIG. **1**.

FIG. **3** is a flowchart illustrating operation of one embodiment of a node controller shown in FIGS. **1** and **2** in response to receiving a communication from a processor core.

FIG. **4** is a flowchart illustrating operation of one embodiment of a node controller shown in FIGS. **1** and **2** in response to receiving a communication to be routed to the processor cores.

FIG. **5** is a block diagram of one embodiment of the redundancy check circuit for use with "N" processor cores.

FIG. **6** is a flowchart illustrating operation of one embodiment of the node controller including the redundancy check circuit shown in FIG. **5** in response to receiving a communication from a processor core.

FIG. **7** is a block diagram of another embodiment of the redundancy check circuit for use with "N" processor cores.

While the invention is susceptible to various modifications and alternative forms, specific embodiments thereof are shown by way of example in the drawings and will herein be described in detail. It should be understood, however, that the drawings and detailed description thereto are not intended to limit the invention to the particular form disclosed, but on the contrary, the intention is to cover all modifications, equivalents and alternatives falling within the spirit and scope of the present invention as defined by the appended claims.

DETAILED DESCRIPTION OF EMBODIMENTS

Turning now to FIG. **1**, a block diagram of one embodiment of a computer system **10** is shown. In the illustrated embodiment, the computer system **10** includes nodes **12**A-**12**B, memories **14**A-**14**B, and peripheral devices **16**A-**16**B. The nodes **12**A-**12**B are coupled, and the node **12**B is coupled to the peripheral devices **16**A-**16**B. Each of the nodes **12**A-**12**B are coupled to respective memories **14**A-**14**B. The node **12**A comprises processor cores **18**A-**18**B coupled to a node controller **20**A which is further coupled to a memory controller **22**A and a plurality of HyperTransport™ (HT) interface circuits **24**A-**24**C. The node **12**B similarly comprises processor cores **18**C-**18**D coupled to a node controller **20**B which is further coupled to a memory controller **22**B and a plurality of HT interface circuits **24**D-**24**F. The HT circuits **24**C-**24**D are coupled (via an HT interface, in this embodiment) and the HT circuit **24**F is coupled to the peripheral device **16**A, which is coupled to the peripheral device **16**B in a daisy-chain configuration (using HT interfaces, in this embodiment). The memory controllers **22**A-**22**B are coupled to the respective memories **14**A-**14**B. In one embodiment, each of the nodes **12**A-**12**B may be a single integrated circuit chip comprising the circuitry shown therein in FIG. **1**. That is, each node **12**A-**12**B may be a chip multiprocessor (CMP). Other embodiments may implement the nodes **12**A-**12**B as two or more separate integrated circuits, as desired. Any level of integration or discrete components may be used.

In the illustrated embodiment, each of the node controllers **20**A-**20**B includes a redundancy check circuit and a machine check control unit. For example, the node controller **20**A includes a redundancy check circuit **26**A and a machine check control unit **28**A, and the node controller **20**B includes a redundancy check circuit **26**B and a machine check control unit **28**B. The operation of the redundancy check circuit **26**A and the machine check control unit **28**A will be described in more detail herein, and the operation of the redundancy check circuit **26**B and the machine check control unit **28**B may be similar.

The node controller **20**A may generally be configured to receive communications from the processor cores **18**A-**18**B, the memory controller **22**A, and the HT circuits **24**A-**24**C and to route those communications to the processor cores

18A-18B, the HT circuits 24A-24C, and the memory controller 22A dependent upon the communication type, the address in the communication, etc. In one embodiment, the node controller 20A includes a system request queue (SRQ) into which received communications are written by the node controller 20A. The node controller 20A may schedule communications from the SRQ for routing to the destination or destinations among the processor cores 18A-18B, the HT circuits 24A-24C, and the memory controller 22A. The node controller 20B may be similar with regard to the processor cores 18C-18D, the HT circuits 24D-24F, and the memory controller 22B. Operation of the node 12A and its components will be discussed in more detail below. Operation of the node 12B and its components may be similar.

Generally, the processor cores 18A-18B may use the interface(s) to the node controller 20A to communicate with other components of the computer system 10 (e.g. peripheral devices 16A-16B, processor cores 18C-18D, the memory controllers 22A-22B, etc.). The interface may be designed in any desired fashion. Cache coherent communication may be defined for the interface, in some embodiments. In one embodiment, communication on the interfaces between the node controller 20A and the processor cores 18A-18B may be in the form of packets similar to those used on the HT interfaces. In other embodiments, any desired communication may be used (e.g. transactions on a bus interface, packets of a different form, etc.). Generally, the communications from the processor cores 18A-18B may include requests such as read operations (to read a memory location or a register external to the processor core) and write operations (to write a memory location or external register), responses to probes (for cache coherent embodiments), interrupt acknowledgements, system management messages, etc. In the illustrated embodiment, each processor core 18A-18B may have a separate communication path to the node controller 20A. The communication path may be implemented in a variety of fashions (e.g. a bus, point-to-point interconnect in each direction, etc.).

In one embodiment, the node 12A may support a redundant execution mode in which the processor cores 18A-18B redundantly execute the same software code on the same data. The processor cores 18A-18B may be copies of the same processor design (with the same size caches, the same register files, etc.). Accordingly, if no errors in execution have occurred, communications generated by the processor core 18A should match corresponding communications generated by the processor core 18B in redundant execution mode. For example, the same address should be included in corresponding communications, the same data should be included, etc. Errors that may occur in execution may include soft errors such as soft memory errors (e.g. those detectable using an error checking code (ECC) scheme) or soft communication errors such as noise or interference in the transmission of communications. Errors may also include hard errors such as errors caused by hardware failures.

The redundancy check circuit 26A receives communications generated by the processor cores 18A-18B and, in redundant execution mode, compares the corresponding communications to verify correct execution of the software code. If a mismatch is detected in the comparison, the redundancy check circuit 26A may signal an execution error. An execution error may be any indication that an error has occurred in the execution of the software code, detected via the redundant execution. In the embodiment of FIG. 1, it may not be clear which processor core 18A-18B experienced the error, but the error is detected. Diagnostic software may be used to determine which processor core 18A-18B experienced the error and what corrective measures are to be taken. In various embodiments, the execution error may be signalled in a variety of fashions. For example, in the present embodiment, the redundancy check circuit 26A may signal a machine check exception to the machine control unit 28A. The machine check control unit 28A may cause a machine check exception in at least one of the processor cores 18A-18B (e.g. the processor core 18A-18B designated as the primary processor core described below). Other embodiments may use other exceptions, external interrupts, logging of execution errors to be checked by software, etc. to signal an execution error. Still other embodiments may use any combination of the above to signal an execution error.

In one embodiment, the processor cores 18A-18B may execute in lockstep (that is, may execute the same instructions on the same clock cycle). In CMP embodiments of the node 12A, the processor cores 18A-18B may execute at the same clock frequency and may be driven from the same clock source (derived from a clock input to the node 12A, not shown in FIG. 1). Thus, once the processor cores 18A-18B initiate execution of the software in synchronization, they may remain synchronized as long as they receive communications (e.g. responses to requests) from the node controller 20A concurrently. That is, hardware external to the node 12A may not be required to run the processor cores 18A-18B in lockstep. In embodiments in which the processor cores 18A-18B execute in lockstep, the redundancy check circuit 26A may check that corresponding communications are concurrently received from the processor cores 18A-18B (e.g. received on the same clock cycle). For example, if a valid communication is received from one of the processor cores 18A-18B and a valid communication is not concurrently received from the other processor core 18A-18B, an execution error may be detected.

The node controller 20A may be configured to propagate communications from one of the processor cores 18A-18B (e.g. from the primary processor core) and to discard communications from the other processor core 18A-18B (e.g. from the auxiliary processor core). The node controller 20A may route the propagated communications as appropriate (e.g. to the memory controller 22A or one or more of the HT circuits 24A-24C). Some communications may result in responses to be returned to the processor cores 18A-18B. For example, read requests may result in responses that return the read data. The node controller 20A may concurrently provide the response to both processor cores 18A-18B. Similarly, for external communications received by the node controller 20A that are to be routed to the processor cores 18A-18B (such as probes, system management commands, etc.), the node controller 20A may concurrently provide the communications to the processor cores 18A-18B. For a probe, the probe response from one of the processor cores 18A-18B may be forwarded to the probe source and the other probe response from the other processor core 18A-18B may be discarded after being used to verify the forwarded probe response.

Generally, one of the processor cores 18A-18B may be designated as a primary processor core and the other processor core 18A-18B is referred to as the auxiliary processor core. The primary processor core may be the processor core whose execution results will be used as the result of executing the redundantly-executed software code. The auxiliary processor core is used to verify correct execution by the primary processor core. In other embodiments, more than two processor cores may be provided in a node. In such embodiments, one processor core may be the primary core

and two or more of the remaining processor cores may be auxiliary processor cores. In one implementation, all the remaining processor cores may be used as auxiliary processor cores.

The identification of the primary processor core may be programmable (e.g. in a register in the redundancy check circuit **26**A, in one embodiment). The programmable embodiment is used as an example for the remainder of the disclosure. Alternatively, the primary processor core may be fixedly defined (e.g. via fuses in the node **12**A, pins to the node **12**A, etc.).

In a CMP embodiment, the node **12**A may provide a low cost approach to redundant execution. The node **12**A may be a single integrated circuit chip, occupying one location on a circuit board included in the computer system **10**. Additionally, no external hardware may be required, in some embodiments, as mentioned above. The embodiment of the computer system **10** may be a multiprocessor computer system (one "processor" per node **12**) with redundancy for each "processor" implemented internally. Other embodiments of the computer system **10** may have a single node **12**A and may be uniprocessor system with redundancy. Additionally, in some embodiments, the computer system **10** may execute with redundant execution mode disabled, providing an execution environment without redundancy and each processor core **18**A-**18**B may be used independently.

As mentioned above, the software code that is to be executed redundantly may be executed in lockstep, in some embodiments. By providing responses, probes, and other communications to the processor cores **18**A-**18**B concurrently, the processor cores **18**A-**18**B may naturally remain in lockstep. Synchronization of the launching of execution (to get the processor cores **18**A-**18**B executing in lockstep) may be performed in a variety of fashions. For example, a software semaphores may be used to control the beginning of execution of the software code, and the response that provides the semaphore data that releases execution may be provided concurrently to the processor cores **18**A-**18**B. In other embodiments, the processor cores **18**A-**18**B may be put in lockstep from the first instruction fetch after reset, or the first fetch of the software code to be executed redundantly, again by providing the response for the first fetch to each processor core **18**A-**18**B concurrently.

The memories **14**A-**14**B may comprise any suitable memory devices. For example, a memory **14**A-**14**B may comprise one or more RAMBUS DRAMs (RDRAMs), synchronous DRAMs (SDRAMs), double data rate (DDR) SDRAM, static RAM, etc. The address space of the computer system **10** may be divided among memories **14**A-**14**B. Each node **12**A-**12**B may include a memory map (e.g. in the node controller **20**A) used to determine which addresses are mapped to which memories **14**A-**14**B, and hence to which node **12**A-**12**B a memory request for a particular address should be routed. The memory controllers **22**A-**22**B may comprise control circuitry for interfacing to the memories **14**A-**14**B. Additionally, the memory controllers **22**A-**22**B may include request queues for queuing memory requests, etc.

The HT circuits **24**A-**24**F may comprise a variety of buffers and control circuitry for receiving packets from an HT link and for transmitting packets upon an HT link. The HT interface comprises unidirectional links for transmitting packets. Each HT circuit **24**A-**24**F may be coupled to two such links (one for transmitting and one for receiving). A given HT interface may be operated in a cache coherent fashion (e.g. between nodes **12**A-**12**B) or in a non-coherent fashion (e.g. to/from peripheral devices **16**A-**16**B). In the illustrated embodiment, the HT circuits **24**C and **24**D are coupled via coherent HT links for communicating between the nodes **12**A-**12**B. The HT circuits **24**A-**24**B and **24**E are not in use, and the HT circuit **24**F is coupled via non-coherent links to the peripheral devices **16**A-**16**B.

The peripheral devices **16**A-**16**B may be any type of peripheral devices. For example, the peripheral devices **16**A-**16**B may include devices for communicating with another computer system to which the devices may be coupled (e.g. network interface cards, circuitry similar to a network interface card that is integrated onto a main circuit board of a computer system, or modems). Furthermore, the peripheral devices **16**A-**16**B may include video accelerators, audio cards, hard or floppy disk drives or drive controllers, SCSI (Small Computer Systems Interface) adapters and telephony cards, sound cards, and a variety of data acquisition cards such as GPIB or field bus interface cards. It is noted that the term "peripheral device" is intended to encompass input/output (I/O) devices.

Generally, a processor core **18**A-**18**D may comprise circuitry that is designed to execute instructions defined in a given instruction set architecture. That is, the processor core circuitry may be configured to fetch, decode, execute, and store results of the instructions defined in the instruction set architecture. The processor cores **18**A-**18**D may comprise any desired configurations, including superpipelined, superscalar, or combinations thereof. Other configurations may include scalar, pipelined, non-pipelined, etc. Various embodiments may employ out of order speculative execution or in order execution. The processor core may include microcoding for one or more instructions or other functions, in combination with any of the above constructions. Various embodiments may implement a variety of other design features such as caches, translation lookaside buffers (TLBs), etc.

It is noted that, while the present embodiment uses the HT interface for communication between nodes and between a node and peripheral devices, other embodiments may use any desired interface or interfaces for either communication. For example, other packet based interfaces may be used, bus interfaces may be used, various standard peripheral interfaces may be used (e.g. peripheral component interconnect (PCI), PCI Express™, etc.), etc.

It is noted that, while the computer system **10** illustrated in FIG. **1** comprises two nodes **12**A-**12**B, other embodiments may implement one node or more than two nodes. Similarly, each node **12**A-**12**B may include two or more processor cores, in various embodiments. Various embodiments of the computer system **10** may include different numbers of HT interfaces per node **12**A-**12**B, differing numbers of peripheral devices coupled to one or more of the nodes, etc.

The present discussion refers to a communication from the primary processor core and a corresponding communication from an auxiliary processor core (or corresponding communications from multiple auxiliary processor cores). For a given communication, the corresponding communication may be the communication that, in the absence of any errors, should match the given communication. If the processor cores are running in lock step, the given communication and its corresponding communication(s) may be received in the node controller concurrently (e.g. on the same clock cycle).

Turning now to FIG. **2**, a block diagram illustrating one embodiment of a portion of the node controller **20**A is shown. In the illustrated embodiment, the node controller **20**A includes a system request queue (SRQ) **30**, an SRQ

control unit **32**, a crossbar **34**, a multiplexor (mux) **36**, a set of input flops **38**A-**38**B, the redundancy check circuit **26**A, and the machine check control unit **28**A. The flops **38**A-**38**B are coupled to receive communications from the processors **18**A-**18**B, respectively, and are coupled to the mux **36** and the redundancy check circuit **26**A. The mux **36** is coupled to receive selection controls from the SRQ control unit **32** and is coupled to the SRQ **30**. The SRQ control unit **32** is coupled to the SRQ **30**, the redundancy check circuit **26**A, and the machine check control unit **28**A. The SRQ **30** is coupled to the crossbar **34**, which is further coupled to receive and provide communications to the HT circuits **24**A-**24**C and the memory controller **22**A, and to provide communications to the processor cores **18**A-**18**B. In the illustrated embodiment, the redundancy check circuit **26**A may include a comparator **40** and a configuration register **42**. A pair of valid bit update circuits **44**A-**44**B are also shown in FIG. **2**, coupled to receive valid bits from the flops **38**A-**38**B (respectively), coupled to receive a PC bit and RE bit from the configuration register **42**, and coupled to the mux **36**.

The SRQ **30** is configured to store external communications (e.g. requests, responses, etc.) received by the node **12**A (e.g. through the HT circuits **24**A-**24**C) as well as internal communications generated by the memory controller **22**A (e.g. probes for read/write requests transmitted to the memory controller **22**A, read responses providing data for a read request, etc.) and the processor cores **18**A-**18**B (e.g. requests, responses, etc.). Generally, the SRQ **30** comprises multiple entries, each entry configured to store a different communication. The SRQ control unit **32** may be configured to allocate entries for received communications, and may be configured to schedule communications for transmission through the crossbar **34**. Additionally, the SRQ control unit **32** may be configured to update state in the SRQ **30** for a given communication in response to completion of various internal activities. For example, if a probe is in the SRQ **30** and the corresponding responses from the processors **18**A-**18**B are received for the probe, the SRQ control unit **32** may update the state in the entry storing the probe to record the receipt of the response and to record the coherency state reported in the response.

The crossbar **34** may comprise circuitry configured to route communications between the various sources and destinations. The sources may include the SRQ **30** (a request scheduled by the SRQ control unit **32**), the HT circuits **24**A-**24**C, and the memory controller **22**A. The destinations may include the SRQ **30**, the HT circuits **24**A-**24**C, the memory controller **22**A, and the processor cores **18**A-**18**B. Generally, received communications are routed to the SRQ **30** for later scheduling by the SRQ control unit **32**. Communications scheduled from the SRQ **30** and transmitted to the crossbar **34** may include information identifying the destination or destinations for the communication, which the crossbar **34** may use to route the communications.

In the redundant execution mode, the SRQ control unit **32** may be configured to schedule communications to be routed to the processor cores **18**A-**18**B concurrently for routing to both the processor cores **18**A-**18**B. If redundant execution mode is not active, the SRQ control unit **32** may schedule communications for routing to each of the processor cores **18**A-**18**B independently for each core, if desired. If the illustrated embodiment, the control register **42** may store a redundant mode enable (RE) bit which may indicate whether or not redundant execution mode is active (e.g. active when set, not active when clear, although the opposite meanings may also be used). The SRQ control unit **32** may receive the RE bit for use in scheduling decisions.

In the illustrated embodiment, the SRQ **30** may include a write port shared by the processor cores **18**A-**18**B and a write port (or ports) used by the crossbar **34**. The processor cores **18**A-**18**B may be configured to transmit a communication (e.g. request or response) to the node controller at most once every other clock cycle. The communications may be captured in the flops **38**A-**38**B and provided to the mux **36**. The SRQ control unit **32** may control the mux **36** to write received communications from the flops **38**A-**38**B into the SRQ **30** (or to update an existing entry of the SRQ **30**, in the case of a response). If only one communication is received in a given clock cycle, the SRQ control unit **32** may select the corresponding flop **38**A-**38**B through the mux **36**. If two communications are received in a given clock cycle, the SRQ control unit **32** may select one flop **38**A-**38**B in one clock cycle and, in the following clock cycle, select the other flop **38**A-**38**B. With the maximum transmission of one communication every other clock cycle, there is time to handle both communications before a new communication may be received. In other embodiments, a direct path from the processor cores **18**A-**18**B may be provided to the mux **36** in addition to the flops **38**A-**38**B, if desired. In still other embodiments, a write port for each processor core **18**A-**18**B may be provided on the SRQ **30** and the mux **36** may be eliminated.

More generally, a node **12**A-**12**B may include N processor cores, where N is an integer greater than one. The processor cores may be configured to transmit a communication at most once every N clock cycles. Thus, N simultaneous communications may be captured and written to the SRQ **30** using a shared write port, similar to the above discussion.

In the redundant execution mode, the communications from the primary processor core may be propagated and the communications from the auxiliary processor core(s) may be discarded. In one embodiment, each communication may include a valid bit (V in the flops **38**A-**38**B) that indicates whether or not a valid communication has been received. The valid bit update circuits **44**A-**44**B may be used to selectively clear the valid bit for communications from the auxiliary processor core. In this embodiment, the PC bit from the configuration register **42** may identify the primary processor core. Accordingly, if redundant execution mode is enabled and the processor core **18**A is not identified by the PC bit as the primary processor core, the valid bit update circuit **44**A may ensure that the valid bit from the flop **38**A is cleared when passed to mux **36**. Otherwise, the valid bit from the flop **38**A is passed unmodified. Similarly, if redundant execution mode is enabled and the processor core **18**B is not identified by the PC bit as the primary processor core, the valid bit update circuit **44**B may ensure that the valid bit from the flop **38**B is cleared when passed to mux **36**. Otherwise, the valid bit from the flop **38**B is passed unmodified. In other embodiments, the valid bits may be selectively cleared at the input to the flops **38**A-**38**B, or communications may be propagated/discarded in other fashions. In embodiments that include more than two processor cores, the PC field may be multiple bits identifying the primary processor core and additional valid bit update circuits similar to the circuits **44**A-**44**B may be included for communications from each processor core.

In the illustrated embodiment, the comparator circuit **40** may receive the communications stored in the flops **38**A-**38**B for comparison. If valid communications are received and a mismatch is detected by the comparator circuit **40**, the redundancy check circuit **26**A may signal an execution error.

The valid bits may also be received by the redundancy check circuit **26**A, to verify that the communications are concurrently received.

The machine check control unit **28**A may implement machine check functionality, at least for those machine check errors that are detected outside of the processor cores **18**A-**18**B. Generally, machine check errors may be errors detected in the operation of the hardware itself. For example, error correction code (ECC) errors for memory accesses, communication transmission errors, cyclic redundancy check (CRC) errors, parity errors, and the like may cause machine check errors. If a machine check error is detected (and, in one embodiment, if the error is enabled in the machine check control unit **28**A), a machine check exception may be delivered to at least one of the processor cores **18**A-**18**B. The machine check exception may cause the receiving processor core **18**A-**18**B to suspend current execution and to fetch instructions from a predefined machine check exception vector for processing the machine check exception. As mentioned above, in the present embodiment, one of the machine check errors is an execution error detected by the redundancy check circuit **26**A in the redundant execution mode. If an execution error is signalled by the redundancy check circuit **26**A to the machine check control unit **28**A, the machine check control unit **28**A may cause a machine check exception to be delivered to one of the processor cores **18**A-**18**B. In one embodiment, the machine check exception may be delivered to the primary processor core. In other embodiments, the machine check exception may be delivered to all processor cores, or a subset of the processor cores, if desired.

The SRQ **30** may have any construction (e.g. registers, random access memory (RAM), etc.). In one embodiment, the SRQ **30** may comprise a shift queue storing various state information for each request, an address RAM storing an address provided in the communication, and a data RAM configured to store data associated with the communication (e.g. a cache block located by the address). Each shift queue entry may comprise a pointer to the address RAM and a pointer to the data RAM, locating the corresponding address and data for a communication (if any).

Turning now to FIG. **3**, a flowchart is shown illustrating operation of one embodiment of the node controller **20**A (including the redundancy check circuit **26**A) in response to receiving a communication from a processor core **18**A-**18**B. While the blocks are shown in a particular order in FIG. **3** for ease of understanding, any order may be used. Furthermore, blocks may be performed in parallel by combinatorial logic circuitry in the node controller **20**A. Blocks, portions of the flowchart, or the flowchart as a whole may also be pipelined over multiple clock cycles, if desired. The communication received from the processor core **18**A-**18**B may be a request, a response to a previous communication transmitted to the processor core **18**A-**18**B such as a probe or system management message, etc.

If redundant execution mode is not active (decision block **50**, "no" leg), the communication may be handled normally (block **52**). That is, no comparison with other communications need be performed. The communication may be handled independent of communications from other processor cores, in general, etc.

If redundant execution mode is active (decision block **50**, "yes" leg), the redundancy check circuit **26**A may check if a valid communication is received from each processor core (decision block **54**). Since the flowchart of FIG. **3** is performed in response to receiving a communication from at least one processor core, then a valid communication should have been received from each processor core. If a valid communication was not received from each processor core (decision block **54**, "no" leg), the redundancy check circuit **26**A may signal the machine check control unit **28**A to assert a machine check exception (block **56**). In one embodiment, the machine check exception may be delivered to the primary processor core. If a valid communication is received from each processor core (decision block **54**, "yes" leg), the redundancy check circuit **26**A may determine if the communications match (decision block **58**). For example, the comparator circuit **40** may be used to compare the communications. If the communications do not match (decision block **58**, "no" leg), the redundancy check circuit **26**A may signal the machine check control unit **28**A to assert the machine check exception (block **56**). If the communications match (decision block **58**, "yes" leg), the node controller **20**A may discard the communication from the auxiliary processor core and propagate the communication from the primary processor core (block **60**). Propagating the communication may generally include transmitting the communication to its destination. For example, in the embodiment of FIG. **2**, propagating the communication may include writing the communication to the SRQ **30**, scheduling the communication, and transmitting the communication through the crossbar **34**.

It is noted that checking for the concurrent reception of valid communications (decision block **54**) and checking for a mismatch in the communications (decision block **58**) may be combined. For example, the comparator circuit **40** may receive the valid bits along with the other communication bits and may compare the valid bits in parallel with the other communication bits, detecting a mismatch if at least one bit differs. In other embodiments, the redundancy check circuit **26**A may compare the state of the signals used by the processor core **18**A to communicate with the node controller **20**A to the state of the signals used by the processor core **18**B to communicate with the node controller **20**A on a clock cycle by clock cycle basis. Such embodiments may be used, for example, if the processor cores **18**A-**18**B are executing in lock step. In such embodiments, a communication as used in the present discussion for comparison may be the state of the signals on a given clock cycle.

In one embodiment, the communication from the primary processor core may be propagated even if the machine check exception is signalled, to ensure that the primary processor core may complete the underlying instruction that caused the communication and thus ensuring that the primary processor core may take the exception. For example, if the primary processor core is expecting a response to the communication, the primary processor core may delay the exception until the response is received. Additionally, if the machine check exception is delivered to the primary processor core and not the auxiliary processor core, the communication may be propagated to ensure that the auxiliary processor core may complete the underlying instruction (e.g. by receiving the response). In other embodiments, the communication from the primary processor core may not be propagated (e.g. if the exception cancels the underlying instruction and permits the exception to be taken).

Turning now to FIG. **4**, a flowchart is shown illustrating operation of one embodiment of the node controller **20**A (including the redundancy check circuit **26**A) in response to receiving a communication for a processor core **18**A-**18**B. That is, the communication may be a communication to be routed to the processor core **18**A-**18**B. The communication may be received from the memory controller **22**A or the HT circuits **24**A-**24**C, for example. Such communications may

include a response to a communication previously transmitted by the processor core (e.g. a read response with data, a done message for a write request, etc.) or may be another communication, such as a probe for maintaining cache coherency or a system management message. While the blocks are shown in a particular order in FIG. **4** for ease of understanding, any order may be used. Furthermore, blocks may be performed in parallel by combinatorial logic circuitry in the node controller **20**A. Blocks, portions of the flowchart, or the flowchart as a whole may also be pipelined over multiple clock cycles, if desired.

If redundant execution mode is not active (decision block **70**, "no" leg), the node controller **20**A may handle the communication normally (block **72**). That is, the communication may be scheduled for routing to the processor core **18**A-**18**B to which it is addressed. If the communication is to be routed to each processor core **18**A-**18**B (e.g. a probe), the communication may be independently routed to each processor core. If redundant execution mode is active (decision block **70**, "yes" leg), the node controller **20**A may route the communication concurrently to each processor core **18**A-**18**B (block **74**). For example, in the embodiment of FIG. **2**, the communication may be scheduled when it can be routed to each processor core **18**A-**18**B, and may be routed through the crossbar **34** to both processor cores **18**A-**18**B concurrently.

As mentioned previously, in other embodiments a node **12**A may include more than two processor cores **18**A-**18**B. Thus, in redundant execution mode, there may be more than one auxiliary core. In some embodiments, all processor cores in a node may be used in redundant execution mode unless an error has been detected (one processor core as the primary processor core and remaining processor cores as the auxiliary processor cores). If an error is detected, in some cases, a processor core may be disabled. In other embodiments, a fixed or programmable number of the processor cores (less than the total number in the node) may participate in redundant execution.

In such embodiments, additional information about the error may be presumed in some cases. For example, if a communication from the primary processor core miscompares with the corresponding communication from each of the auxiliary processor cores, it may be reasonable to conclude that the primary processor core experienced the error. If the primary processor core is programmably selected, a different processor core may be selected as the primary processor core and the previous primary processor core may be disabled. Similarly, if a communication from the primary processor core miscompares with the corresponding communication from only one of the auxiliary processor cores (i.e. the primary processor core's communication matches with the other auxiliary processor cores), it may be reasonable to conclude that the auxiliary processor core experienced the error and the erroring auxiliary processor may be disabled. If a miscompare of a communication from the primary processor core and the corresponding communication from at least two, but not all, of the auxiliary processor cores is detected, diagnostic software may be invoked to determine the error.

FIG. **5** is a block diagram of one embodiment of a portion of the redundancy check circuit **26**A that may be used in a node that includes N processor cores, where N is an integer greater than 2. The last processor core is labeled **18**N merely to provide a distinct reference numeral in FIG. **5**.

The portion of the redundancy check circuit **26**A shown in FIG. **5** may include the logic used to compare the communications when there is more than one auxiliary core. The redundancy check circuit **26**A receives the communication from each processor core **18**A, **18**B, to **18**N (e.g. from corresponding input flops similar to flops **36**A-**36**B, in one embodiment). The communications are provided to a mux **80** and to respective comparator circuits **82**A-**82**N. The output of the mux **80** is also coupled to the comparator circuits **82**A-**82**N. The outputs of the comparator circuits **82**A-**82**N are coupled to a control circuit **84**, which is coupled to the machine check control unit **28**A and to the configuration register **42**. The configuration register **42** includes a primary core (PC) field encoded to identify which of the processor cores **18**A-**18**N is designated as the primary core. The configuration register **42** may further include a set of core enables which indicate which cores are participating in redundant execution mode. The combination of the core enables may also serve as the redundant execution mode enable, or a separate enable bit may be provided.

In the illustrated embodiment, the primary processor core is designated by programming the configuration register **42**. Thus, the communication from the designated processor core is selected through the mux **80** (under control of the value in the PC field) for comparison with the communication from all processor cores. In other embodiments, if the primary processor core is fixedly assigned, the mux **80** may be eliminated in favor of hardwiring the primary processor core's communication lines to the comparator circuits **82**A-**82**N (and a comparator circuit **82**A-**82**N for the primary processor core itself may also be eliminated).

The control circuit **84** may receive the comparison results from the comparators **82**A-**82**N, and may optionally disable a processor core or cores from comparison in the redundant execution mode and/or signal a machine check exception dependent on the comparison results. Additional details are provided below with regard to the FIG. **6**.

Turning now to FIG. **6**, a flowchart is shown illustrating operation of one embodiment of the node controller **20**A (including the redundancy check circuit **26**A) in response to receiving a communication from a processor core **18**A-**18**N. While the blocks are shown in a particular order in FIG. **6** for ease of understanding, any order may be used. Furthermore, blocks may be performed in parallel by combinatorial logic circuitry in the node controller **20**A. Blocks, portions of the flowchart, or the flowchart as a whole may also be pipelined over multiple clock cycles, if desired. The communication received from the processor core **18**A-**18**N may be a request, a response to a previous communication transmitted to the processor core **18**A-**18**N such as a probe or system management message, etc. Handling of communications to be routed to the processor cores **18**A-**18**N may be similar to the flowchart of FIG. **4**.

If redundant execution mode is not active (decision block **90**, "no" leg), the communication may be handled normally (block **92**). That is, no comparison with other communications need be performed. The communication may be handled independent of communications from other processor cores, in general, etc.

If redundant execution mode is active (decision block **90**, "yes" leg), the control circuit **84** may mask the comparison results (from the comparator circuits **82**A-**82**N) according to which processor cores are currently enabled, as indicated in the configuration register **42** (block **94**). If the masked comparison results indicate that the communication from the primary processor core mismatches with a communication with a single auxiliary processor core (and matches with communications from other auxiliary processor cores—decision block **96**, "yes" leg), the control circuit **84** may disable the mismatching auxiliary processor core (block **98**).

In one embodiment, the control circuit **84** may merely disable further comparison with the mismatching auxiliary processor core by clearing the enable bit for the mismatching auxiliary processor core in the configuration register **42**. The mismatching auxiliary processor core may continue executing. In other embodiments, the control circuit **84** may indirectly or directly disable the mismatching auxiliary processor core from executing (e.g. by causing a machine check exception and having software disable the mismatching auxiliary processor core, by signaling hardware designed to disable the mismatching auxiliary processor core from executing, by holding the mismatching auxiliary processor core in reset, by putting the mismatching auxiliary processor core in a sleep mode, etc.).

If the masked comparison results indicate that the communication from the primary processor core mismatches with all auxiliary processor cores (that are enabled) (decision block **100**, "yes" leg), the control circuit **84** may disable the primary processor core and select one of the auxiliary processor cores to be the primary processor core (block **102**). As with the discussion above for disabling the auxiliary processor core, the control circuit **84** may simply disable the primary processor core from comparison by clearing the corresponding enable bit in the configuration register **42**, or may disable the primary processor core from executing at all. The control circuit **84** may also update the PC field of the configuration register **42** to reflect the newly selected primary processor core. The control circuit **84** may use any selection mechanism. For example, the processor cores may be numbered (e.g. 0 to N−1), and the control circuit **84** may increment the number of the presently selected primary processor core to select a new primary processor core. If one or more processor cores are disabled in the configuration register **42**, the control circuit **84** may increment until an enabled processor core is selected. Additionally, the control circuit **84** may signal the machine check control unit **28**A to assert a machine check exception (e.g. to the newly selected primary processor core—block **104**) to permit diagnostic software to perform any desired additional error handling. Alternatively, a machine check exception may not be signalled for this case in other embodiments.

If the masked comparison results indicate that the communication from the primary processor core miscompares with more than one auxiliary processor core, but not all auxiliary processor cores (decision block **106**, "yes" leg), the control circuit **84** may signal the machine check control unit **28**A to assert a machine check exception (e.g. to the primary processor core) to permit diagnostic software to attempt to locate the error (block **104**).

Generally, in each of the mismatch cases and the case of no mismatch, the node controller **20**A may discard the communications from the auxiliary processor cores and propagate the communication from the primary processor core (block **108**).

In another embodiment, a node **12**A may include N processor cores **18**A-**18**N, where N is an even integer (although **18**N is used merely as a unique reference character). The processor cores may be configured as N/2 pairs of redundant processor cores. Each pair may redundantly execute different software code from other pairs. In some embodiments, the processor cores may be preassigned into pairs for redundant execution purposes. For example, "odd" and "even" processor cores may be paired (numbering the processor cores from 0 to N−1 as mentioned above). Thus, processor cores **18**A-**18**B may be paired, as may other odd and even pairings up to processor cores **18**M and **18**N. Alternatively, processor core pairings may be programmable.

In a CMP embodiment of the node **12**A having N/2 redundant processor cores, N/2 separate code sequences may be redundantly executed concurrently within a single integrated circuit. That is, the CMP embodiment may provide a redundant, multiprocessing solution in a single chip implementation.

FIG. **7** is an example block diagram of a redundancy check circuit **26**A that may support both a first redundancy mode having a single primary processor core with multiple auxiliary processor cores and a paired redundancy mode having paired sets of redundant processor cores. In the embodiment of FIG. **7**, the mux **80** is provided to select communications from the various processor cores including the processor cores **18**A, **18**B, **18**M, and **18**N. The mux **80** is controlled by the PC field of the configuration register **42** as previously described. However, rather than being directly coupled to the comparators **82**A-**82**N, the output of mux **80** is provided as an input to a set of muxes such as muxes **120**A-**120**B. There may be a mux similar to muxes **120**A-**120**B for each pair of processor cores in the node **12**A. Each mux **120**A-**120**B may be further coupled to receive the communications from one pair of processor cores (e.g. mux **120**A is coupled to receive the communications from the processor cores **18**A-**18**B and the mux **120**B is coupled to receive the communications from the processor cores **18**M-**18**N). The output of each mux **120**A-**120**B is coupled as an input to a pair of comparators **82**A-**82**N (e.g. the output of the mux **120**A is coupled to the comparators **82**A-**82**B and the output of the mux **120**B is coupled to the comparators **82**M-**82**N). Each comparator **82**A-**82**N has another input coupled to receive a communication from a different one of the processor cores **18**A-**18**N and an output coupled to the control circuit **84**.

If paired redundancy mode is selected (via an OE field in the configuration register **26**A), the muxes **120**A-**120**B may select a communication from one of the odd or even processor cores to which it is coupled. If paired redundancy mode is not selected, the muxes **120**A-**120**B may select the output of the mux **80**. If paired redundancy mode is not selected, the operation of the redundancy check circuit **26**A may be similar to that described above with regard to FIGS. **5** and **6**. Operation of each pair of processor cores in the paired redundancy mode may be similar to that described above with regard to FIG. **3**.

For example, the OE field may comprise two bits, one for the odd processor core and one for the even processor core. Setting the bit for the odd processor core may cause selection of the odd processor core communications through the muxes **120**A-**120**B. Setting the bit for the even processor core may cause selection of the even processor core communications through the muxes **120**A-**120**B. If both bits are clear, paired redundancy mode is not selected. In other embodiments, the opposite meanings may be assigned to the set and clear states of the bits, or other encodings may be used for the OE field.

The odd or even processor core selected via the OE field may be viewed as the primary processor core of the pair. In other embodiments, the OE field may be used to select the paired redundancy mode and either the odd processor core or the even processor core may be preassigned as the primary processor core. In such embodiments, each mux **120**A-**120**B may be coupled to receive the output of the mux **80** and either the odd processor core communication or the even processor core communication, whichever is preas-

signed as the primary processor core. In still other embodiments, the redundancy check circuit may not support the first redundancy mode (having one primary processor core and multiple auxiliary processor cores) and the mux **80** may be eliminated. In other such embodiments, the muxes **120**A-**120**B may also be eliminated and pairs of processor cores may be hardwired to a single comparator circuit for that pair, similar to the embodiment of FIG. **2**. Additionally, the core enables in the configuration register **42** may be used to selectively enable various pairs for redundant execution. The comparison of communications in FIG. **7** may be based on receiving valid communications, or may be on a clock cycle by clock cycle basis (where a communication may be the state of the signals on a given clock cycle for comparison purposes).

Numerous variations and modifications will become apparent to those skilled in the art once the above disclosure is fully appreciated. It is intended that the following claims be interpreted to embrace all such variations and modifications.

What is claimed is:

1. A node comprising:

a plurality of processor cores; and

a node controller coupled to the plurality of processor cores and configured to route communications from the plurality of processor cores to other devices in a computer system, wherein the node controller comprises a circuit coupled to receive the communications from the plurality of processor cores and, in a redundant execution mode in which at least a first processor core of the plurality of processor cores is redundantly executing code that a second processor core of the plurality of processor cores is also executing, the circuit is configured to compare communications from the first processor core to communications from the second processor core to verify correct execution of the code, and wherein the node controller is configured to discard communications from the first processor core responsive to comparing the communications to corresponding communications from the first processor core and verifying that the communications and corresponding communications match.

2. The node as recited in claim **1** wherein the circuit is configured to compare a first communication from the first processor core to a second communication from the second processor core, wherein the first communication and the second communication are received concurrently by the circuit.

3. The node as recited in claim **2** wherein, if a valid communication is received from a first one of the first processor core and the second processor core and no valid communication is received from the other one of the first processor core and the second processor core, the circuit is configured to signal an execution error.

4. The node as recited in claim **2** wherein a separate communication path is provided between each of the plurality of processor cores and the node controller.

5. The node as recited in claim **1** wherein, if the circuit detects a mismatch between a first communication from the first processor core and a corresponding second communication from the second processor core, the circuit is configured to signal an execution error.

6. The node as recited in claim **5** wherein signalling the execution error comprises signalling a machine check exception.

7. The node as recited in claim **1** wherein the node controller is configured to propagate communications from the second processor core.

8. The node as recited in claim **7** wherein the node controller, in response to receiving a response to a communication propagated from the second processor core, is configured to route the response to both the first processor core and the second processor core.

9. The node as recited in claim **1** wherein a plurality of the plurality of processor cores are configured to redundantly execute the code in the redundant execution mode, wherein the plurality of the plurality of processor cores include the first processor core, the second processor core, and at least one other processor core.

10. The node as recited in claim **9** wherein one of the plurality of the plurality of processor cores is a primary processor core and remaining ones of the plurality of the plurality of processor cores are auxiliary processor cores, and wherein the circuit is configured to select communications received from the primary processor core for comparison to each of the plurality of the plurality of processor cores.

11. The node as recited in claim **10** wherein, if the circuit detects a mismatch between a communication from the primary processor core and a corresponding communication from only one auxiliary processor core, the circuit is configured to disable comparison of communications from that auxiliary processor core.

12. The node as recited in claim **10** wherein, if the circuit detects a mismatch between a communication from the primary processor core and a corresponding communication from each of the auxiliary processor cores, the circuit is configured to disable the primary processor core and select one of the auxiliary processor cores as the primary processor core.

13. The node as recited in claim **10** wherein, if the circuit detects a mismatch between a communication from the primary processor core and a corresponding communication from at least two of the auxiliary processor cores but not all of the auxiliary processor cores, the circuit is configured to signal an execution error.

14. The node as recited in claim **1** wherein the plurality of processor cores comprise a plurality of pairs of processor cores, and wherein each pair of the plurality of pairs of processor cores is configured to redundantly execute code independently of other pairs of the plurality of pairs of processor cores.

15. The node as recited in claim **1** wherein the plurality of processor cores and the node controller are integrated onto a single integrated circuit.

16. A method comprising:

running at least a first processor core of a plurality of processor cores and a second processor core of the plurality of processor cores in a redundant execution mode in which the first processor core redundantly executes the same code that the second processor core is executing;

verifying correct execution of the code in a node controller coupled to the plurality of processor cores, the verifying including comparing communications received in the node controller from the first processor core and the second processor core; and

discarding communications received from the first processor core and propagating corresponding communications received from the second processor core responsive to verifying that the communications from

the first processor core match the corresponding communications from the second processor core.

17. The method as recited in claim **16** wherein the comparing comprises comparing concurrently received communications.

18. The method as recited in claim **17** further comprising detecting a mismatch in the concurrently received communications and signalling an execution error.

19. The method as recited in claim **16** wherein the running at least the first processor core and the second processor core in the redundant execution mode comprises running each of the plurality of processor cores in the redundant execution mode, wherein one of the plurality of processor cores is a primary processor core and remaining ones of the plurality of processor cores are auxiliary processor cores, the method further comprising:

responsive to detecting a mismatch between a communication from the primary processor core and a corresponding communication from only one auxiliary processor core, disabling comparison of communications from that auxiliary processor core;

responsive to detecting a mismatch between a communication from the primary processor core and a corresponding communication from each of the auxiliary processor cores, disabling the primary processor core and selecting one of the auxiliary processor cores as the primary processor core; and

responsive to detecting a mismatch between a communication from the primary processor core and a corresponding communication from at least two of the auxiliary processor cores but not all of the auxiliary processor cores, signalling an execution error.

* * * * *

US005491788A

United States Patent [19]

Cepulis et al.

[11] Patent Number: 5,491,788

[45] Date of Patent: Feb. 13, 1996

[54] **METHOD OF BOOTING A MULTIPROCESSOR COMPUTER WHERE EXECUTION IS TRANSFERRING FROM A FIRST PROCESSOR TO A SECOND PROCESSOR BASED ON THE FIRST PROCESSOR HAVING HAD A CRITICAL ERROR**

[75] Inventors: **Darren J. Cepulis**, Houston; **Louis R. Gagliardi**, Tomball, both of Tex.

[73] Assignee: **Compaq Computer Corp.**, Houston, Tex.

[21] Appl. No.: **119,424**

[22] Filed: **Sep. 10, 1993**

[51] **Int. Cl.**[6] **G06F 11/00**

[52] **U.S. Cl.** **395/182.11**; 395/182.21; G06F/11/00

[58] **Field of Search** 395/575, 182.11, 395/182.21, 182.22; 371/11.3, 16.1, 67.1, 68.3

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,181,940	1/1980	Underwood et al.	364/200
4,634,110	1/1987	Julich	371/11.3
5,155,731	10/1992	Yamaguchi	371/16.5
5,247,659	9/1993	Curran et al.	395/575
5,408,647	4/1995	Landry	395/575
5,418,955	5/1995	Ikeda et al.	395/700

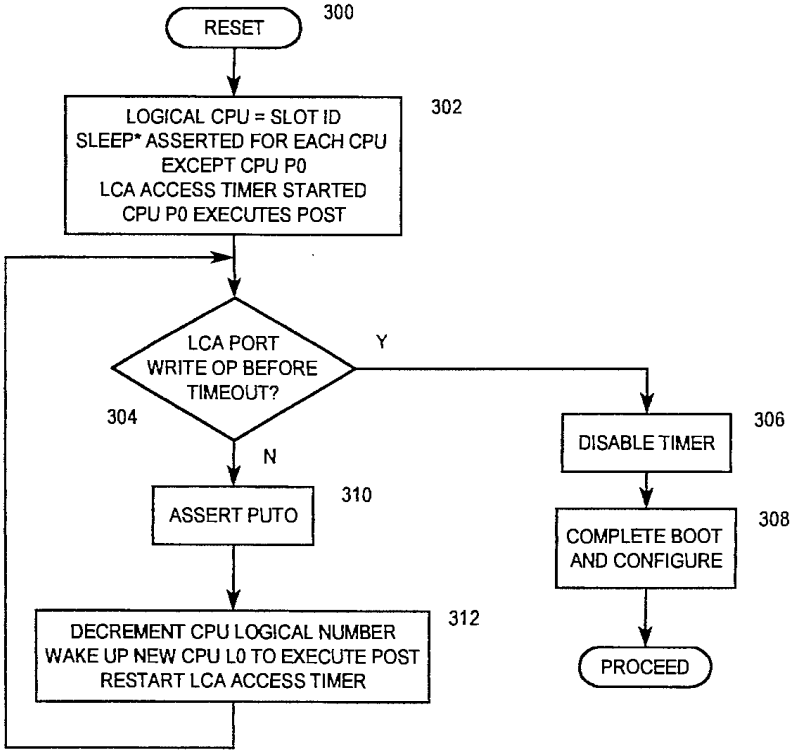
Primary Examiner—Hoa T. Nguyen

Attorney, Agent, or Firm—Pravel, Hewitt, Kimball & Krieger

[57] **ABSTRACT**

A multiprocessor computer system handles the failure of one or more of its processors without totally disabling the system. On power up, all of the CPUs are deactivated except for a CPU in a first physical slot. The power on self test routines review a log of errors and determine if certain critical errors have previously occurred. If so, the CPU in the first physical slot halts operation entirely. If the CPU in the first physical slot is not functioning properly or is halted, the hardware then awakens a CPU in a second physical slot, designates it as the first logical CPU, and the CPU then performs similar diagnostic checks. If it fails, the hardware again tries a third physical CPU and so on. When one CPU passes the initial error review, it proceeds with initialization of the computer system and performs further self testing. If it functions properly, it is designated as the first logical CPU, and retains its designation until the power is cycled. This first logical CPU then awakens the remaining CPUs and boots the rest of the system. If it fails this later self testing by having certain critical errors occur, the logical CPU **0** designation is transferred to another active CPU and the old CPU is halted. The new CPU commences operation effectively where the old CPU halted, so that system initialization is continued not restarted. The power on self test routines then further test the CPU in the first physical slot. Thus, if at least one CPU is operational, the computer system boots and operates.

8 Claims, 13 Drawing Sheets



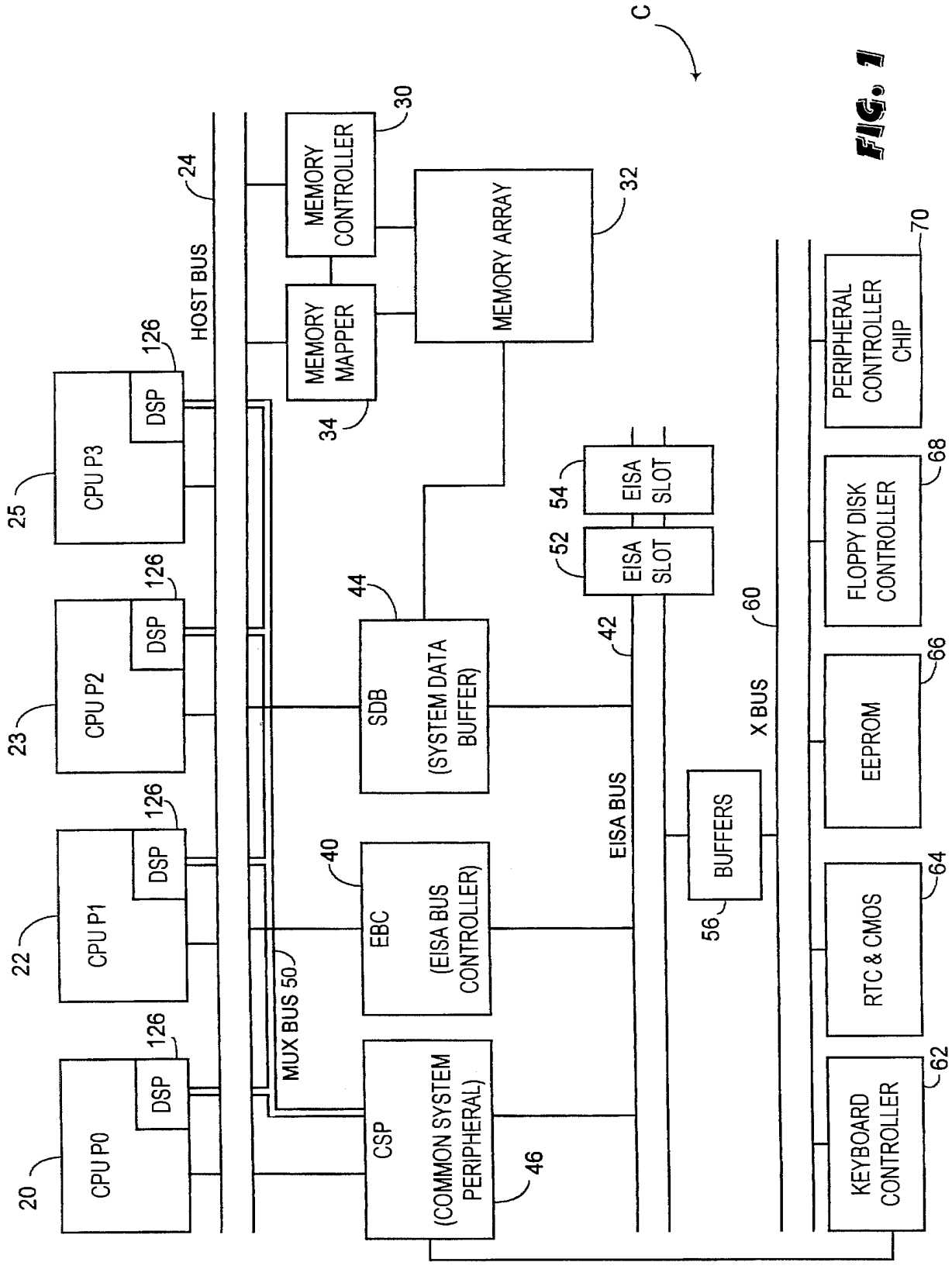

C
HOST BUS
24
CPU P0
20
CPU P1
22
CPU P2
23
CPU P3
25
DSP
126
MUX BUS 50
CSP
(COMMON SYSTEM PERIPHERAL)
46
EBC
(EISA BUS CONTROLLER)
40
SDB
(SYSTEM DATA BUFFER)
44
MEMORY CONTROLLER
30
MEMORY MAPPER
34
MEMORY ARRAY
32
EISA SLOT
52
54
EISA BUS
42
BUFFERS
56
X BUS
60
PERIPHERAL CONTROLLER CHIP
70
FLOPPY DISK CONTROLLER
68
EEPROM
66
RTC & CMOS
64
KEYBOARD CONTROLLER
62

FIG. 1

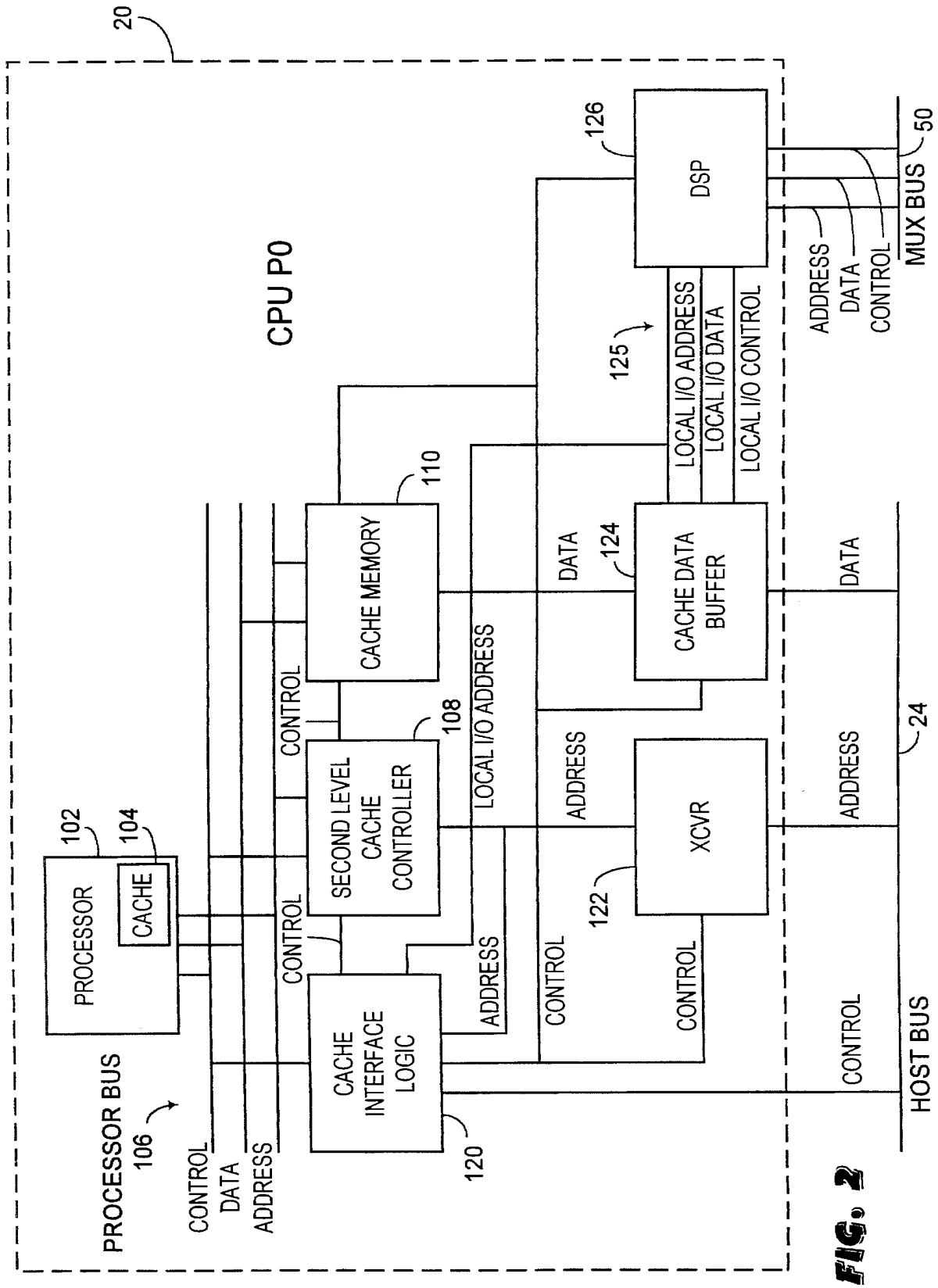

20
CPU P0
PROCESSOR
102
CACHE
104
PROCESSOR BUS
106
CONTROL
DATA
ADDRESS
CONTROL
SECOND LEVEL CACHE CONTROLLER
108
CACHE MEMORY
110
CACHE INTERFACE LOGIC
120
LOCAL I/O ADDRESS
ADDRESS
CONTROL
CONTROL
ADDRESS
DATA
122
XCVR
CONTROL
DATA
124
CACHE DATA BUFFER
125
LOCAL I/O ADDRESS
LOCAL I/O DATA
LOCAL I/O CONTROL
126
DSP
ADDRESS
DATA
CONTROL
MUX BUS
50
ADDRESS
24
DATA
CONTROL
HOST BUS

FIG. 2

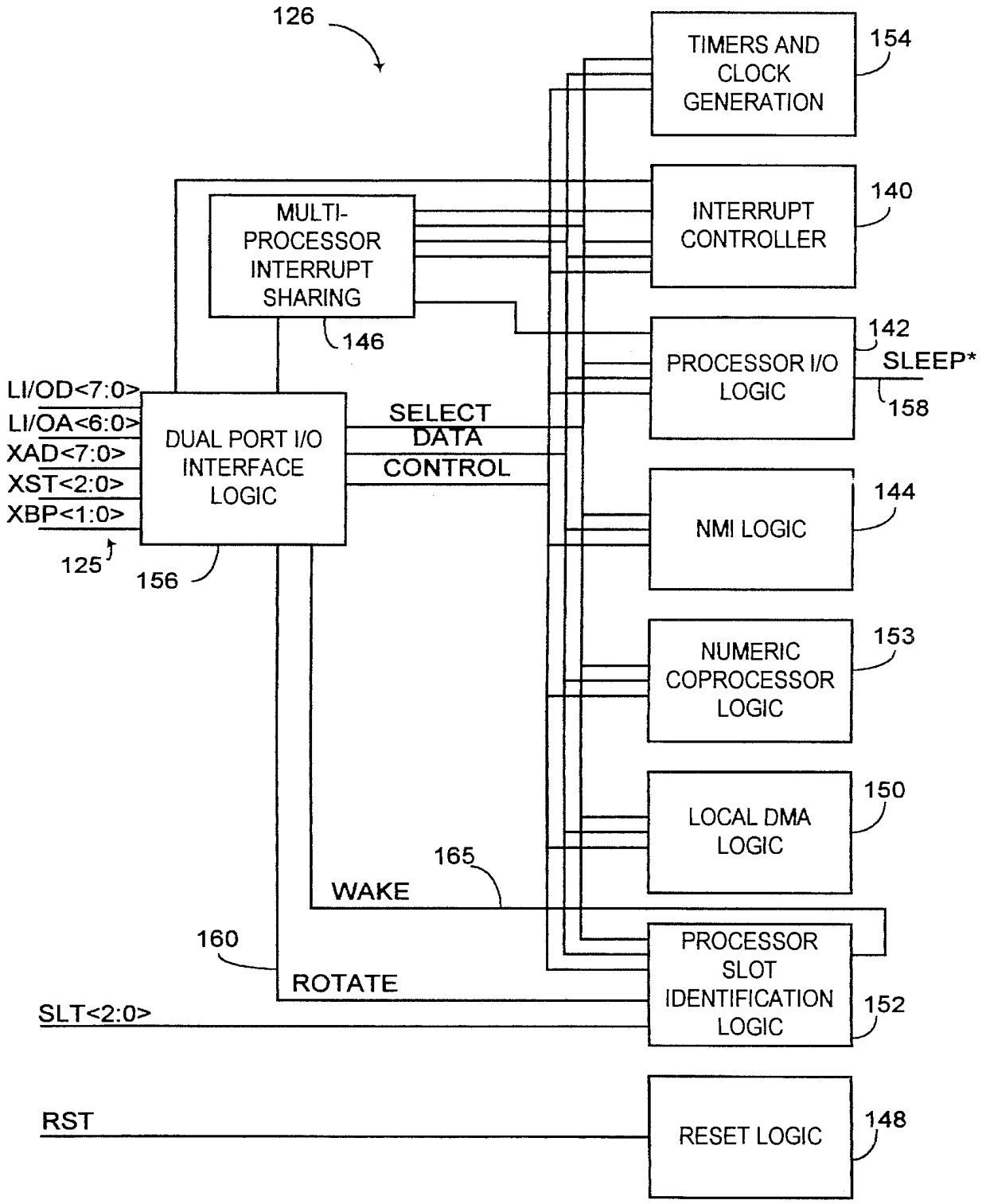

126
TIMERS AND CLOCK GENERATION
154
INTERRUPT CONTROLLER
140
MULTI-PROCESSOR INTERRUPT SHARING
146
PROCESSOR I/O LOGIC
142
SLEEP*
158
LI/OD<7:0>
LI/OA<6:0>
XAD<7:0>
XST<2:0>
XBP<1:0>
DUAL PORT I/O INTERFACE LOGIC
SELECT
DATA
CONTROL
125
156
NMI LOGIC
144
NUMERIC COPROCESSOR LOGIC
153
LOCAL DMA LOGIC
150
165
WAKE
160
ROTATE
PROCESSOR SLOT IDENTIFICATION LOGIC
152
SLT<2:0>
RST
RESET LOGIC
148

FIG. 3

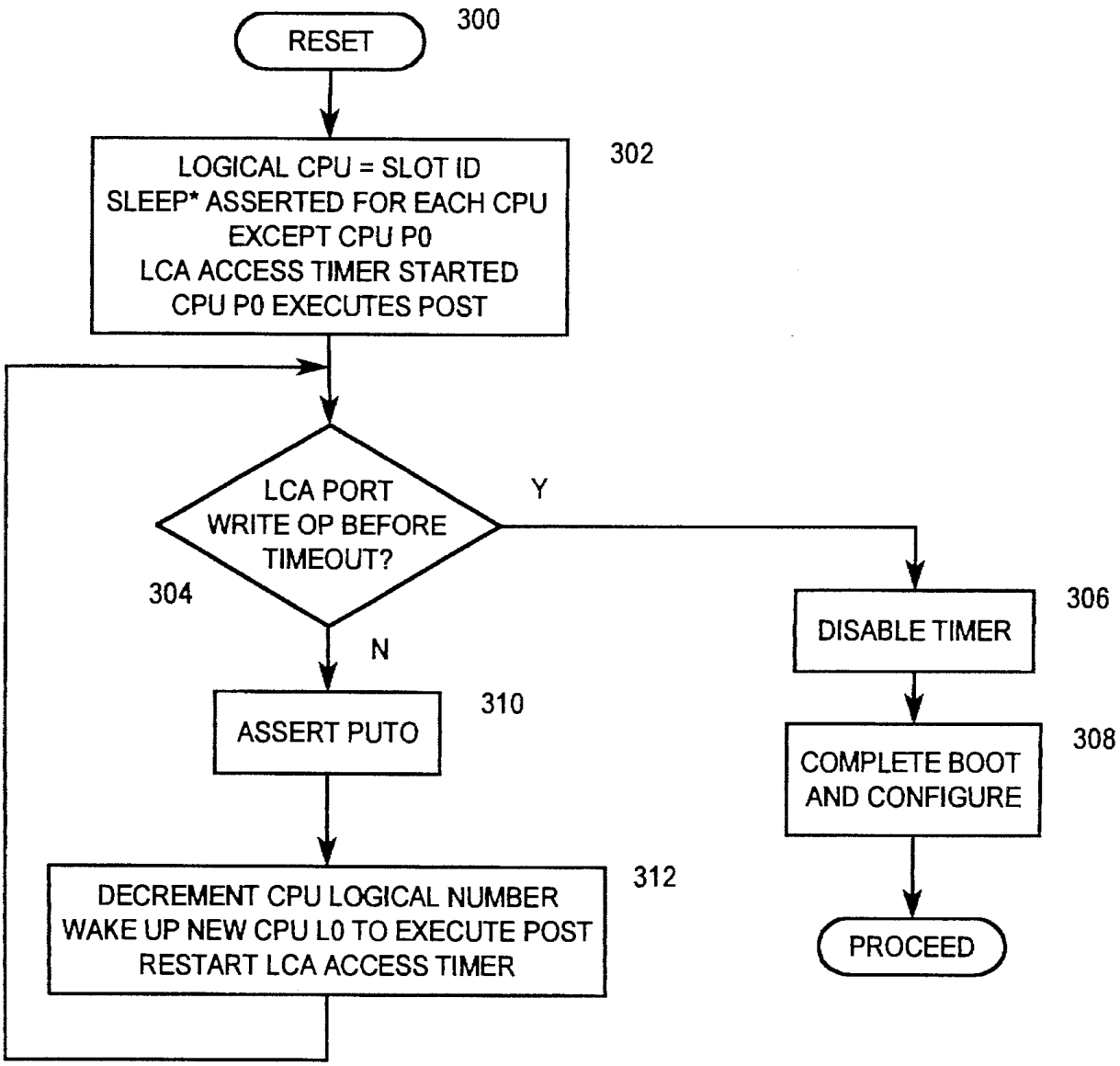

RESET
300
LOGICAL CPU = SLOT ID
SLEEP* ASSERTED FOR EACH CPU
EXCEPT CPU P0
LCA ACCESS TIMER STARTED
CPU P0 EXECUTES POST
302
LCA PORT
WRITE OP BEFORE
TIMEOUT?
304
Y
N
DISABLE TIMER
306
COMPLETE BOOT
AND CONFIGURE
308
PROCEED
ASSERT PUTO
310
DECREMENT CPU LOGICAL NUMBER
WAKE UP NEW CPU L0 TO EXECUTE POST
RESTART LCA ACCESS TIMER
312

FIG. 4

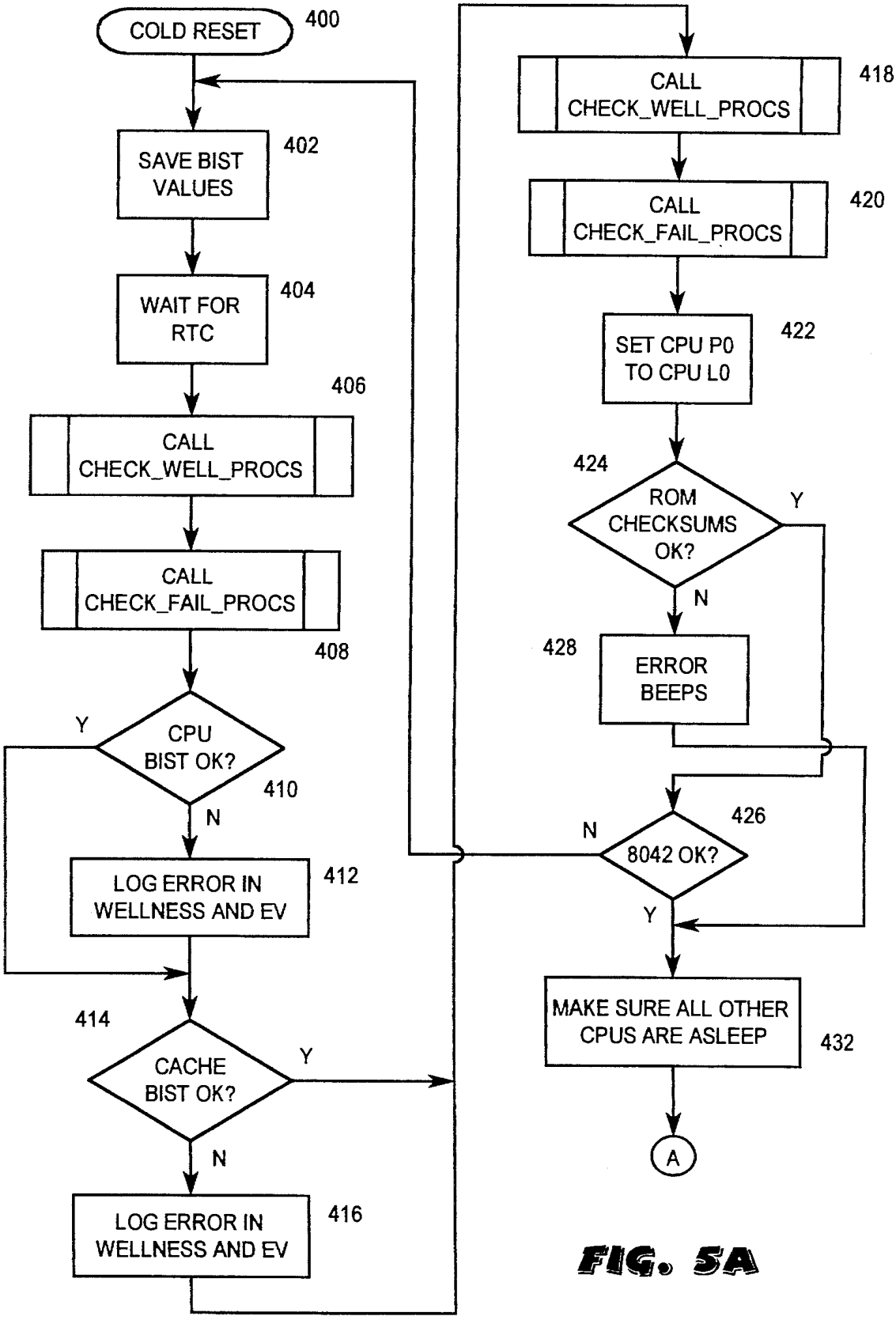

COLD RESET
400
SAVE BIST VALUES
402
WAIT FOR RTC
404
406
CALL CHECK_WELL_PROCS
CALL CHECK_FAIL_PROCS
408
CPU BIST OK?
Y
N
410
LOG ERROR IN WELLNESS AND EV
412
414
CACHE BIST OK?
Y
N
LOG ERROR IN WELLNESS AND EV
416
CALL CHECK_WELL_PROCS
418
CALL CHECK_FAIL_PROCS
420
SET CPU P0 TO CPU L0
422
424
ROM CHECKSUMS OK?
Y
N
428
ERROR BEEPS
426
8042 OK?
N
Y
MAKE SURE ALL OTHER CPUS ARE ASLEEP
432
A

FIG. 5A

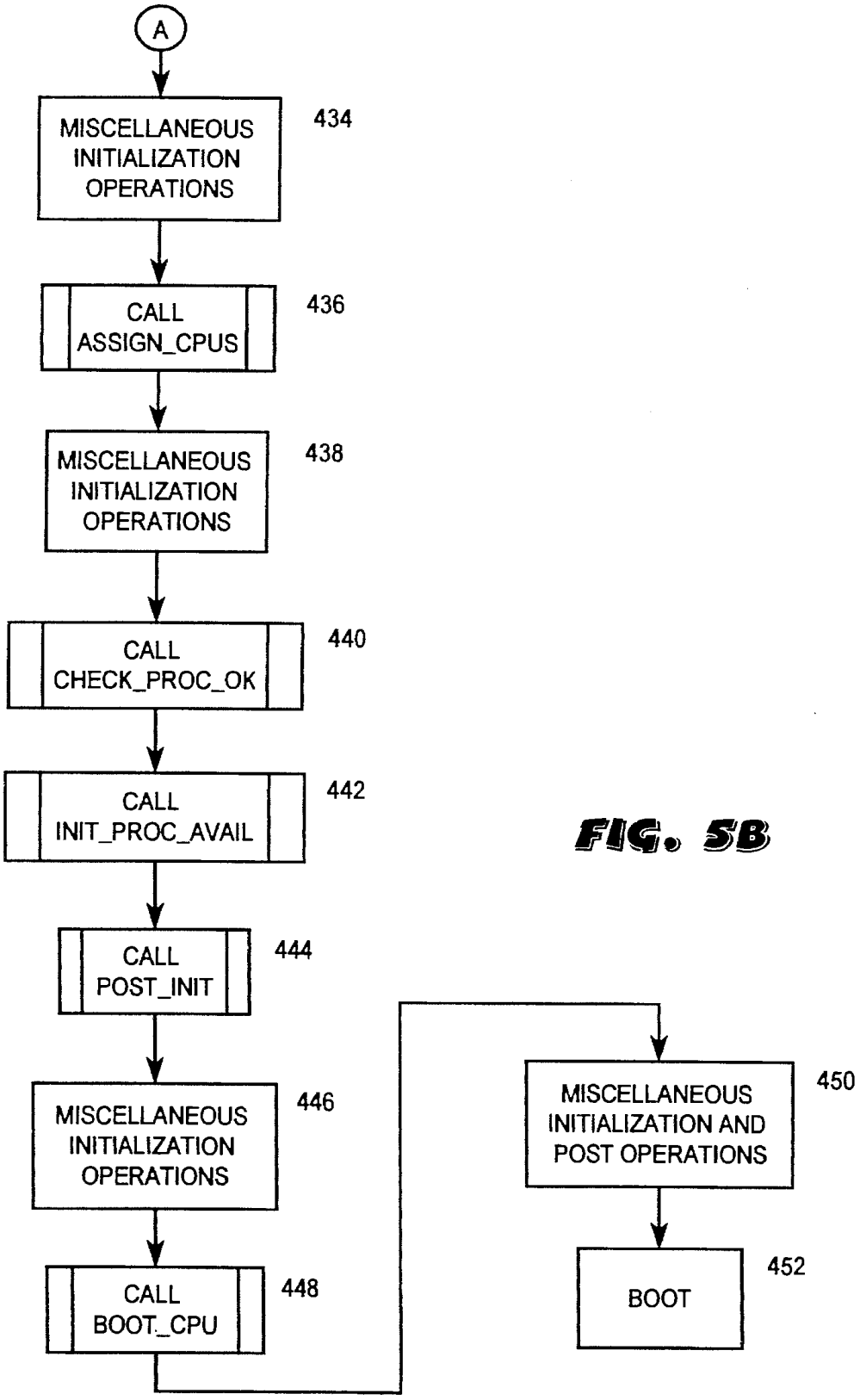

A
MISCELLANEOUS INITIALIZATION OPERATIONS
434
CALL ASSIGN_CPUS
436
MISCELLANEOUS INITIALIZATION OPERATIONS
438
CALL CHECK_PROC_OK
440
CALL INIT_PROC_AVAIL
442
CALL POST_INIT
444
MISCELLANEOUS INITIALIZATION OPERATIONS
446
CALL BOOT_CPU
448
MISCELLANEOUS INITIALIZATION AND POST OPERATIONS
450
BOOT
452

FIG. 5B

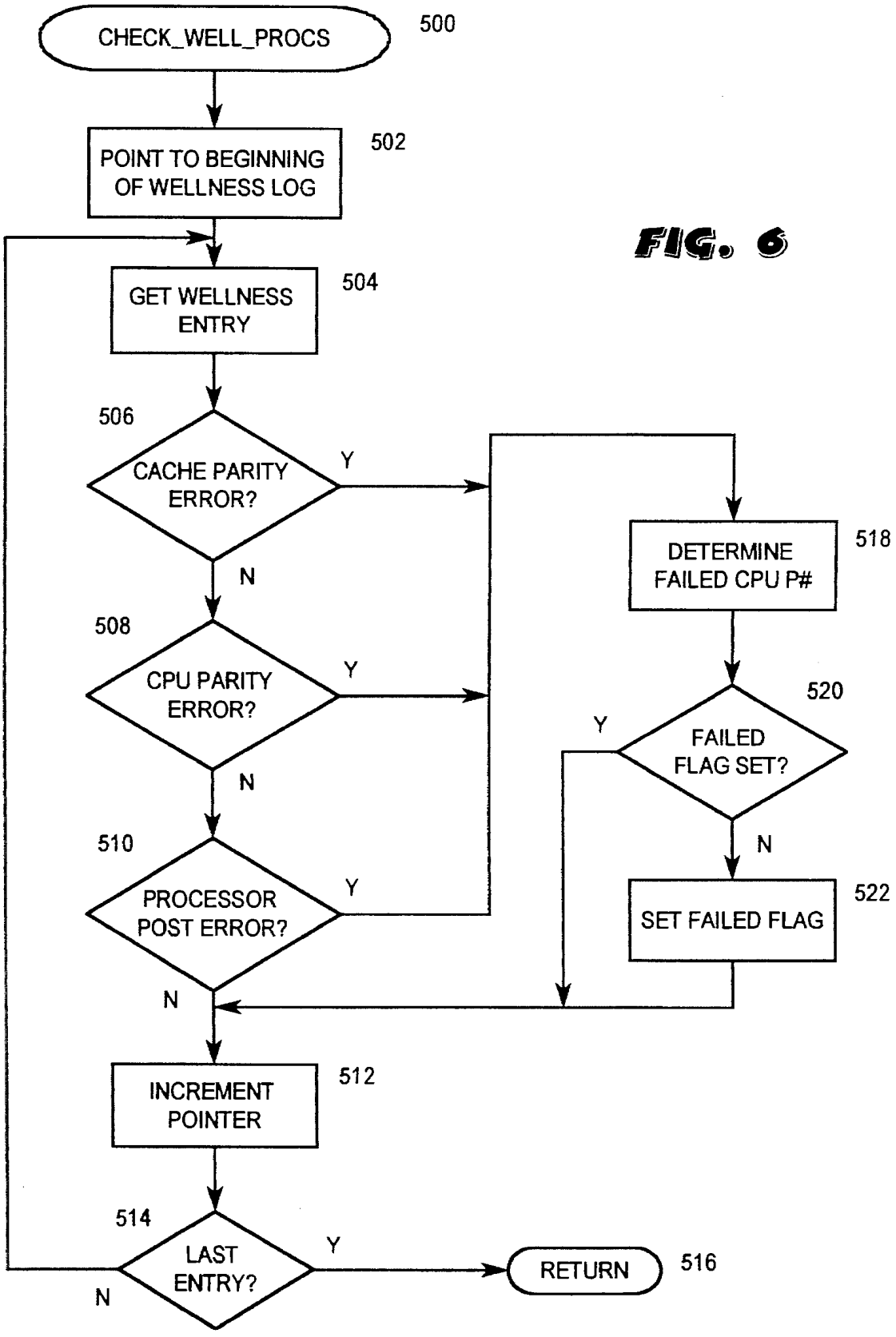

CHECK_WELL_PROCS
500
POINT TO BEGINNING OF WELLNESS LOG
502
GET WELLNESS ENTRY
504
506
CACHE PARITY ERROR?
Y
N
508
CPU PARITY ERROR?
Y
N
510
PROCESSOR POST ERROR?
Y
N
DETERMINE FAILED CPU P#
518
520
FAILED FLAG SET?
Y
N
SET FAILED FLAG
522
INCREMENT POINTER
512
514
LAST ENTRY?
Y
N
RETURN
516

FIG. 6

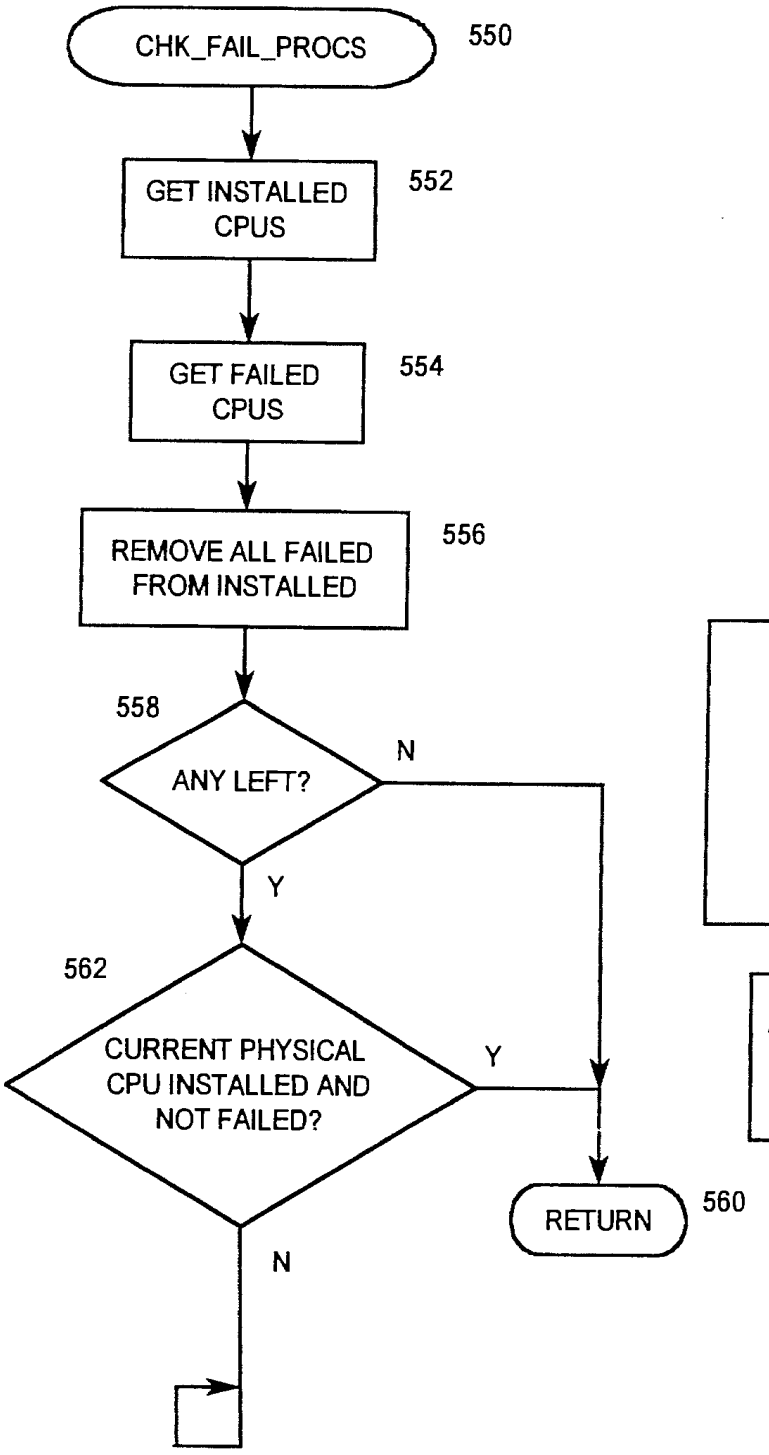

CHK_FAIL_PROCS
550
GET INSTALLED CPUS
552
GET FAILED CPUS
554
REMOVE ALL FAILED FROM INSTALLED
556
558
ANY LEFT?
N
Y
562
CURRENT PHYSICAL CPU INSTALLED AND NOT FAILED?
Y
N
RETURN
560

FIG. 7

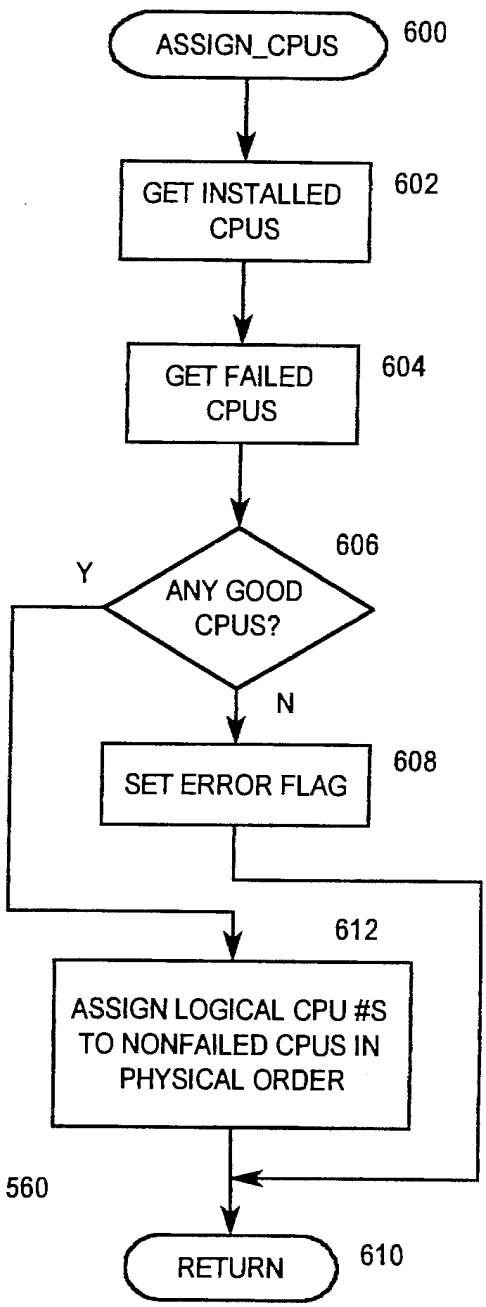

ASSIGN_CPUS
600
GET INSTALLED CPUS
602
GET FAILED CPUS
604
606
Y
ANY GOOD CPUS?
N
SET ERROR FLAG
608
612
ASSIGN LOGICAL CPU #S TO NONFAILED CPUS IN PHYSICAL ORDER
RETURN
610

FIG. 8

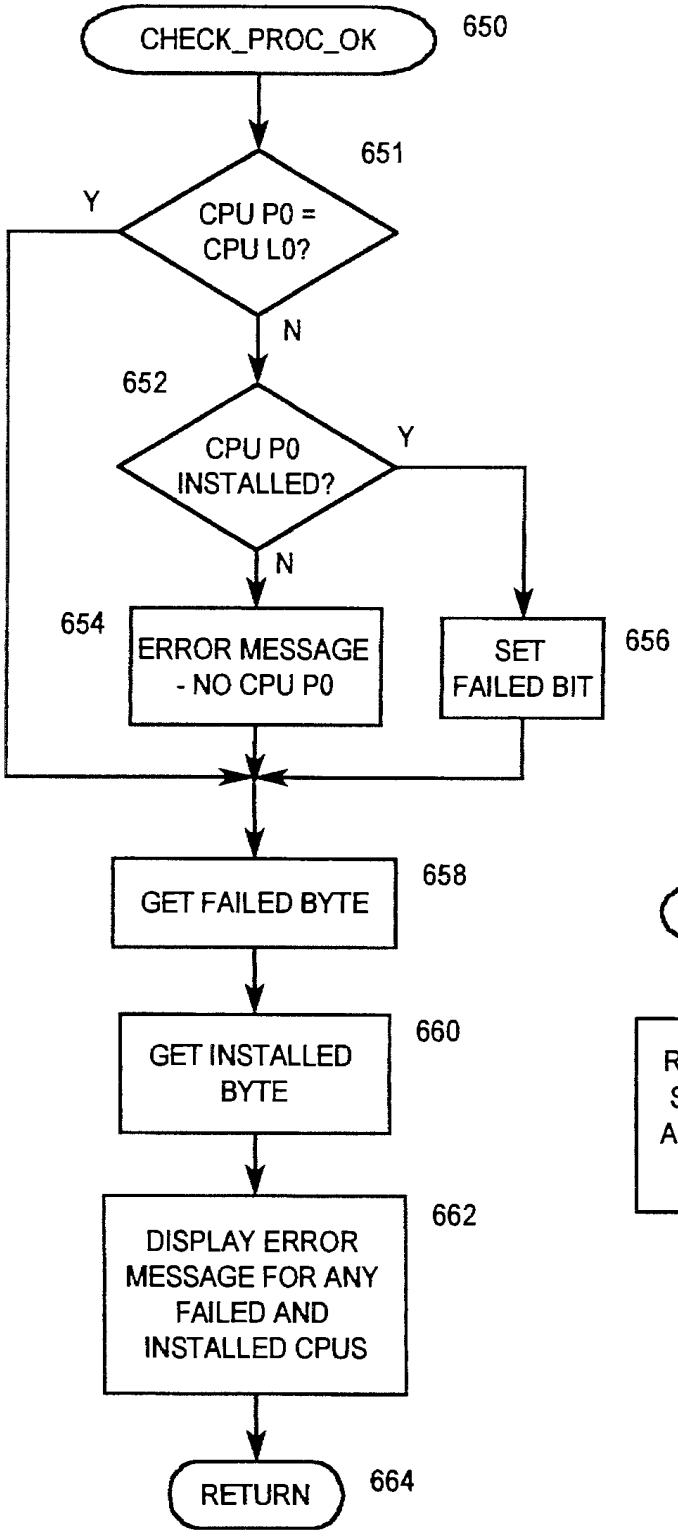

CHECK_PROC_OK
650
651
CPU P0 = CPU L0?
Y
N
652
CPU P0 INSTALLED?
Y
N
654
ERROR MESSAGE - NO CPU P0
SET FAILED BIT
656
GET FAILED BYTE
658
GET INSTALLED BYTE
660
DISPLAY ERROR MESSAGE FOR ANY FAILED AND INSTALLED CPUS
662
RETURN
664

FIG. 9

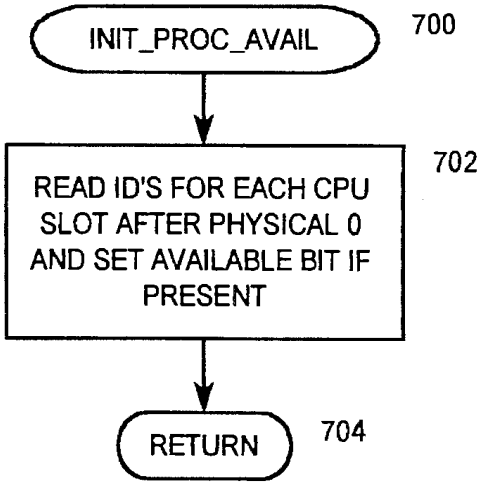

INIT_PROC_AVAIL
700
READ ID'S FOR EACH CPU SLOT AFTER PHYSICAL 0 AND SET AVAILABLE BIT IF PRESENT
702
RETURN
704

FIG. 10

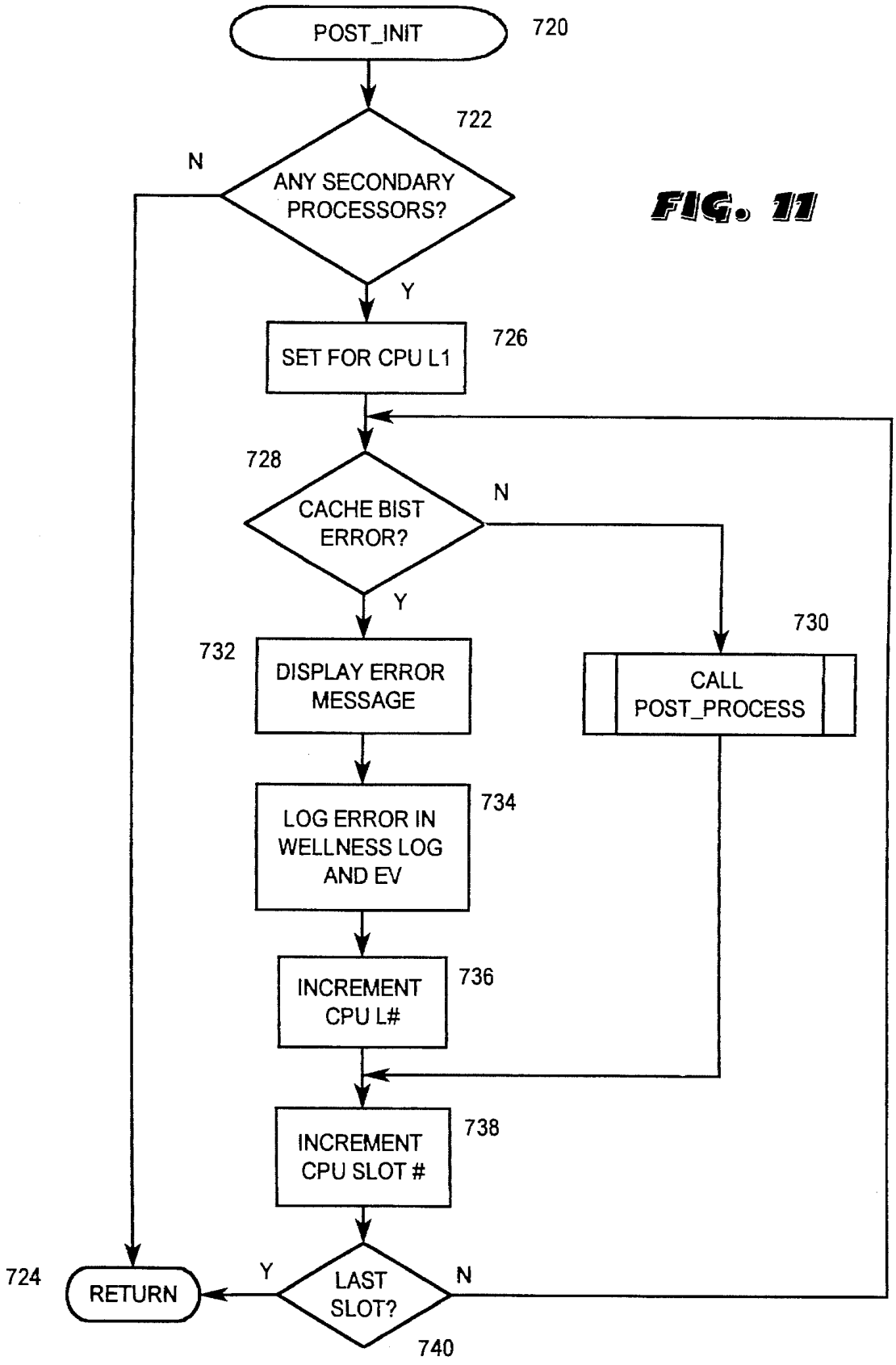

POST_INIT
720
722
ANY SECONDARY PROCESSORS?
N
Y
SET FOR CPU L1
726
728
CACHE BIST ERROR?
N
Y
730
732
DISPLAY ERROR MESSAGE
CALL POST_PROCESS
LOG ERROR IN WELLNESS LOG AND EV
734
INCREMENT CPU L#
736
INCREMENT CPU SLOT #
738
724
RETURN
Y
LAST SLOT?
N
740

FIG. 11

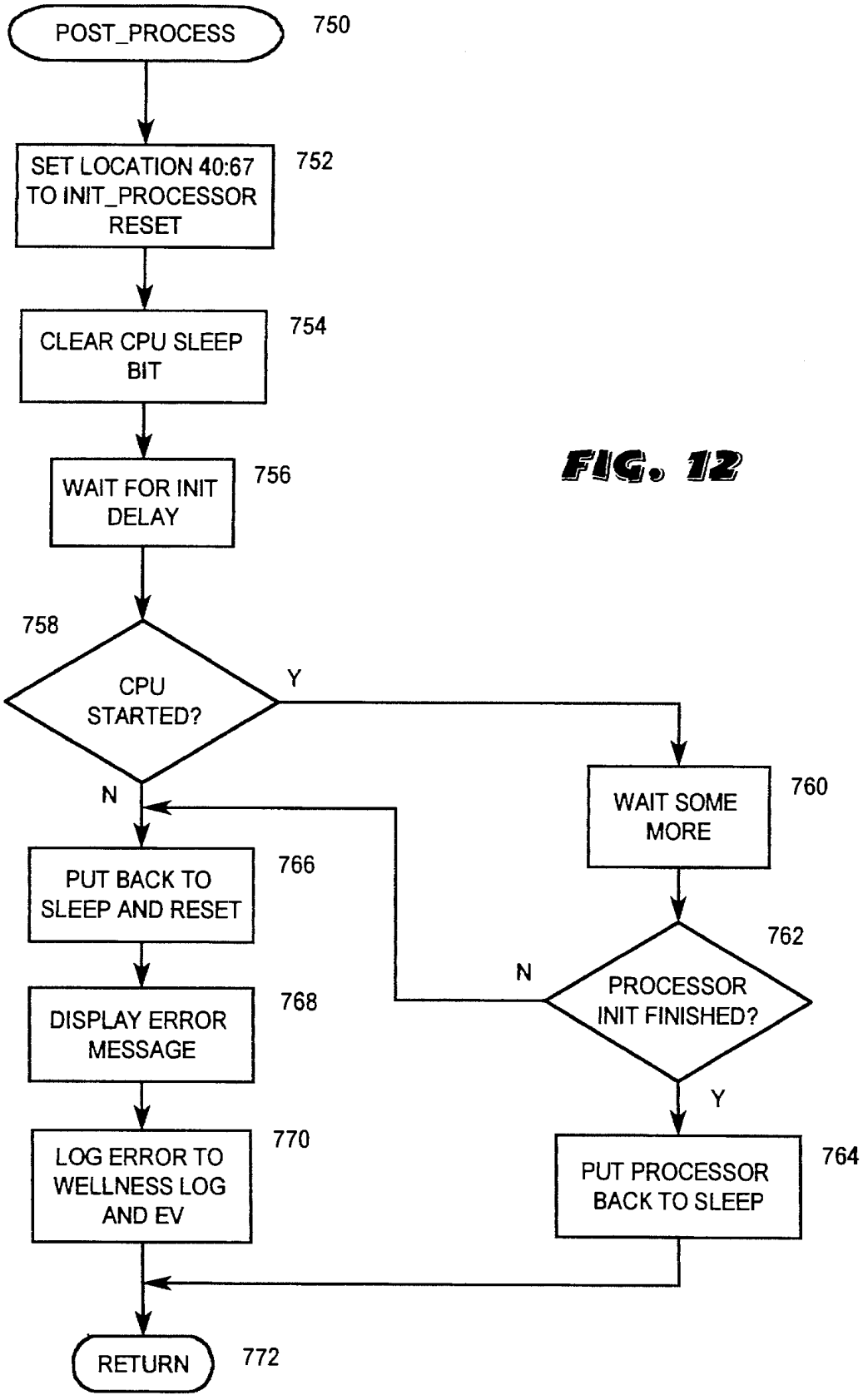

POST_PROCESS
750
SET LOCATION 40:67 TO INIT_PROCESSOR RESET
752
CLEAR CPU SLEEP BIT
754
WAIT FOR INIT DELAY
756
758
CPU STARTED?
Y
N
WAIT SOME MORE
760
762
PROCESSOR INIT FINISHED?
N
Y
PUT BACK TO SLEEP AND RESET
766
DISPLAY ERROR MESSAGE
768
LOG ERROR TO WELLNESS LOG AND EV
770
PUT PROCESSOR BACK TO SLEEP
764
RETURN
772

FIG. 12

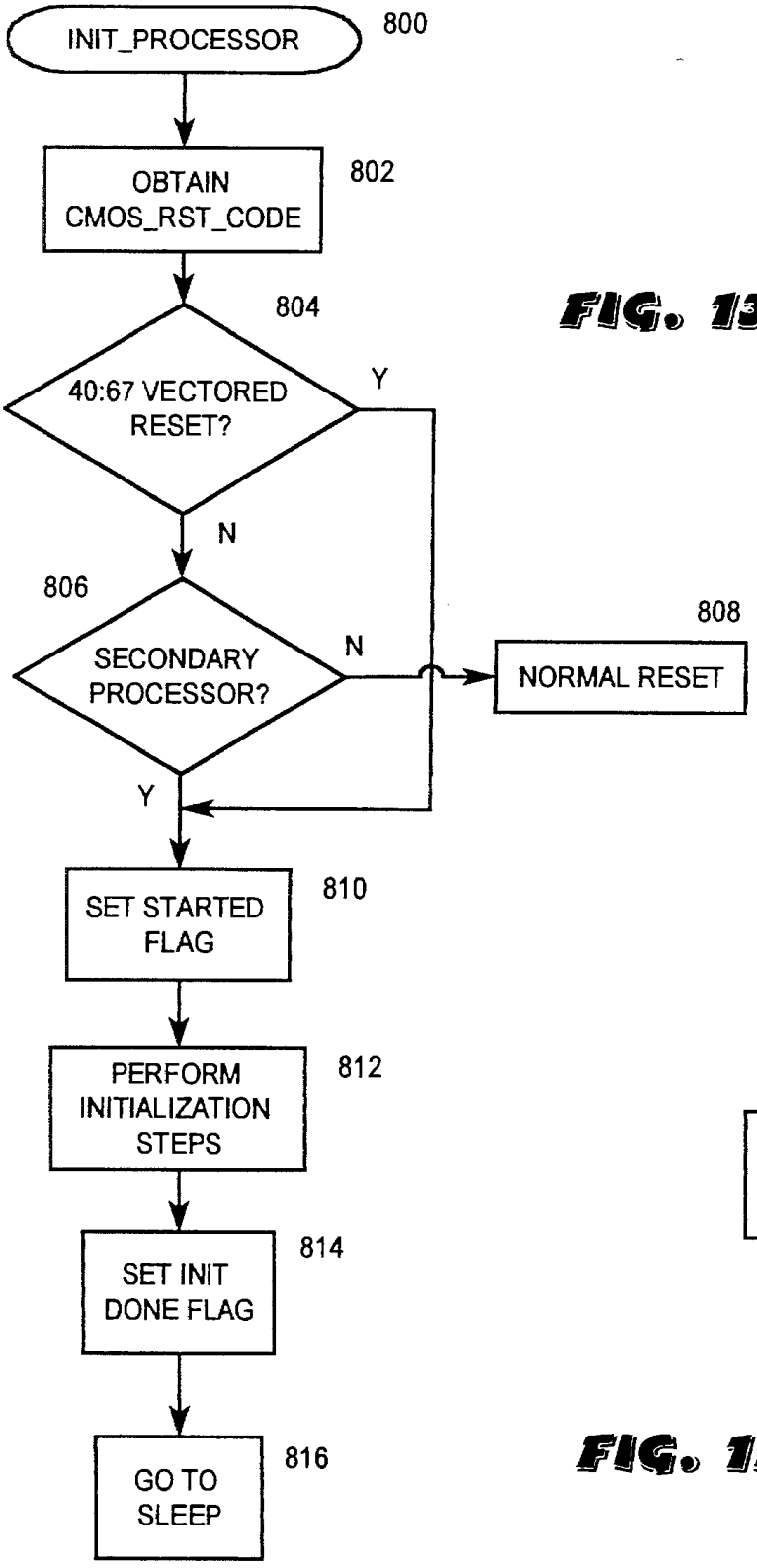

INIT_PROCESSOR
800
OBTAIN CMOS_RST_CODE
802
804
40:67 VECTORED RESET?
Y
N
806
SECONDARY PROCESSOR?
N
808
NORMAL RESET
Y
SET STARTED FLAG
810
PERFORM INITIALIZATION STEPS
812
SET INIT DONE FLAG
814
GO TO SLEEP
816

FIG. 13

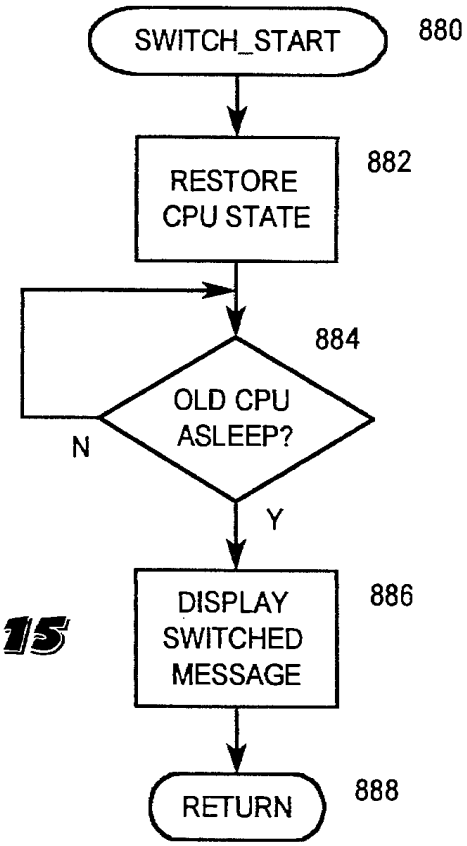

SWITCH_START
880
RESTORE CPU STATE
882
884
OLD CPU ASLEEP?
N
Y
DISPLAY SWITCHED MESSAGE
886
RETURN
888

FIG. 15

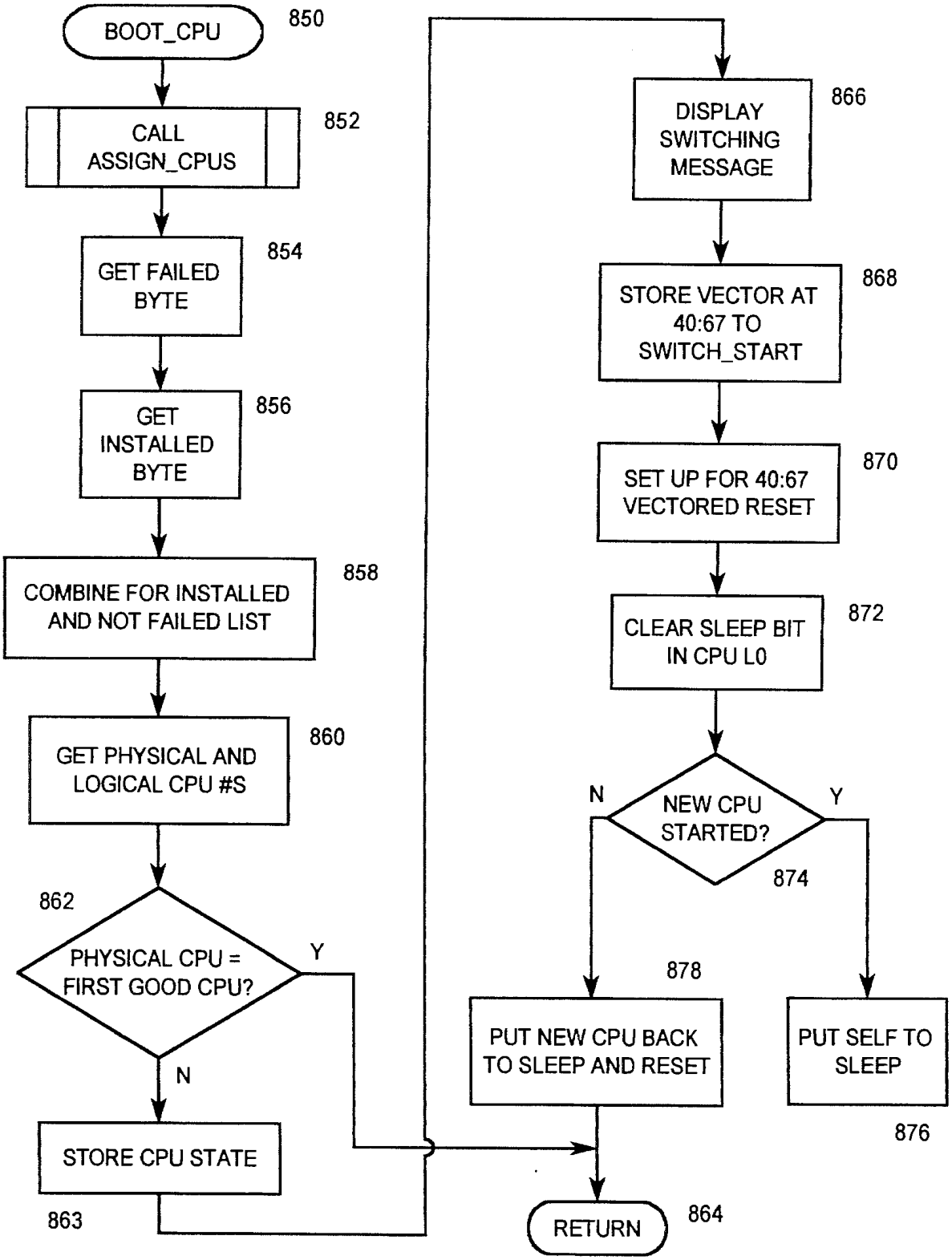

BOOT_CPU
850
CALL ASSIGN_CPUS
852
GET FAILED BYTE
854
GET INSTALLED BYTE
856
COMBINE FOR INSTALLED AND NOT FAILED LIST
858
GET PHYSICAL AND LOGICAL CPU #S
860
862
PHYSICAL CPU = FIRST GOOD CPU?
Y
N
STORE CPU STATE
863
DISPLAY SWITCHING MESSAGE
866
STORE VECTOR AT 40:67 TO SWITCH_START
868
SET UP FOR 40:67 VECTORED RESET
870
CLEAR SLEEP BIT IN CPU L0
872
N
NEW CPU STARTED?
Y
874
878
PUT NEW CPU BACK TO SLEEP AND RESET
PUT SELF TO SLEEP
876
RETURN
864

FIG. 14

METHOD OF BOOTING A MULTIPROCESSOR COMPUTER WHERE EXECUTION IS TRANSFERRING FROM A FIRST PROCESSOR TO A SECOND PROCESSOR BASED ON THE FIRST PROCESSOR HAVING HAD A CRITICAL ERROR

BACKGROUND OF THE INVENTION

1. Field of the Invention

The invention relates to multiprocessor computer systems, and more particularly, to start-up logic for assigning logical central processing unit (CPU) designations among multiple CPUs, with the booting CPU being reassigned based on the existence of certain prior errors on that CPU.

2. Description of the Related Art

Advancements in computer technology proceed at a tremendous rate. Modern microprocessors operate at frequencies so high that processors only a few years old seem sluggish and lethargic in comparison. On the other hand, modern applications have become so complex and versatile that even the high performance delivered by modern systems is taxed to the limit. Consequently, the effort to develop yet more powerful and effective computer systems continues.

One well known method of improving computer performance is to provide multiple processors in a single system. Asymmetrical multiprocessor systems, in which one microprocessor is the master and another microprocessor performs specific functions as a slave of the master microprocessor, are common and well known. Although the master/slave relationship improves computer performance due to the division of tasks, the computer does not operate at maximum capability. This is because the slave processor performs only particularly designated operations, and thus remains idle when a task not designated for the slave processor is performed. While these operations are executed, the computer system is no more efficient than a single processor system.

The computer system's efficiency may be further enhanced by making the multiple processors symmetrical. In a symmetrical system, any processor can perform any required function. Thus, all microprocessors operate simultaneously, spending little or no idle time, and the computer system operates near its maximum efficiency. In addition, the system may be further improved by adding supplementary microprocessors as the workload increases. Adding microprocessors is particularly effective in file server systems having an array of independent functions to be performed simultaneously.

Although symmetrical multiprocessor systems are efficient, they are difficult to design. One of the many obstacles to overcome in designing a symmetrical multiprocessor system is the potential presence of a non-functional processor. A simple method of booting up a multiprocessor system is to power up one of the central processing units (CPUs), generally designated CPU**0**, and ignore the others. When the first CPU has booted up, the first CPU then turns on and tests the remaining CPUs and the various components of the system. If the first microprocessor does not function properly, however, it cannot turn on the remaining processors and the entire system is left incapacitated. Consequently, the computer owner or operator has a computer system with one or more operational CPUs, but ironically, the system is useless until the repairman arrives.

In addition, for many DOS based applications and for booting purposes, one of the CPUs must be designated as CPU**0**. CPU**0** commonly performs various functions for the system, like DRAM refresh operations, which make CPU**0** unique even in a symmetrical multiprocessor system. Thus, most multiprocessor systems require one of the microprocessors to be designated as CPU**0**. In many systems, a CPU residing in a particular physical location is always designated as CPU**0**. If one of the CPU locations is always designated as containing CPU**0**, however, the slot may be empty, or the CPU may fail, crippling the entire system.

One system, described in more detail below, addressed the problem of the CPU in the first physical slot failing by automatically rotating the CPU**0** designation to the CPU in the next physical slot if the first CPU did not perform a selected operation within a given time period. This solved the problem of the total failure of the CPU, as it was marked bad and the CPU**0** designation was rotated. However, it was not helpful in the cases of the more marginal failures, such as parity errors, which are relatively infrequent but are critical enough that the CPU cannot be considered dependable. The prior system would have simply continued until the CPU failed again, with a resulting loss of time and productivity and potential loss of information. Thus, it is desirable to have CPU**0** rotation on causes other than total CPU failure.

SUMMARY OF THE PRESENT INVENTION

The present multiprocessor computer system includes power up logic for finding a fully functioning CPU to designate as CPU**0**, particularly a CPU which has not previously had certain severe or critical errors which limit system dependability. Each microprocessor has a physical location designation which remains constant. Because the CPU designated as CPU**0** may not occupy physical location **0** in the computer, the designated CPU is referred to as logical CPU**0**, or CPU L**0**. When the system is powered up, all of the CPUs except the CPU in physical slot **0** (CPU P**0**) are initially placed in a SLEEP mode. The microprocessor in physical location **0** performs its power on self test (POST). An early part of this POST routine involves reviewing a log of errors and determining if a critical error has occurred. If so, the CPU halts operation. If there were no prior critical errors, the POST routine continues. Included in the POST routine are other tests of the CPU, both the processor itself and the associated cache memory system. If the CPU functions properly during all of these tests, the CPU is designated as CPU L**0**, and the CPU retains this designation until the system is power cycled. The CPU designated as CPU L**0** then awakens the remaining CPUs and boots up the rest of the computer system.

If CPU P**0** is not functioning properly and cannot perform even the most basic functions, it is designated as inoperative. The hardware system then awakens CPU P**1** and repeats the process of testing the CPU. If CPU P**1** is operational, then it is designated as CPU L**0**, and it boots the remainder of the system. On the other hand, if CPU P**1** also fails, it is also given an inoperative designation. The computer system then turns on CPU P**2**, and repeats the process. The process repeats until an operational microprocessor is found to perform the CPU L**0** functions. Consequently, if at least one of the microprocessors in the system is functioning, the computer system boots and operates.

If the early portion of the POST routine determines that there have been critical errors, then the halting of the CPU occurs sufficiently quickly that the rotation hardware per-

forms as described. If, however, the errors are not determined until the later tests of the CPU in the POST routine, the next operational CPU is activated, the CPU **L0** designation passed to that CPU and the original CPU is noted as failed and is halted. The new CPU commences operation at approximately the point where the original CPU halted, so that completed operations of the POST routine are not repeated.

Thus, the multiprocessor system is not crippled by the failure or probable failure of a single processor, based on either total failure or certain critical intermittent failures, enhancing the dependability of the computer system.

BRIEF DESCRIPTION OF THE DRAWINGS

A better understanding of the present invention can be obtained when the following detailed description of the preferred embodiment is considered in conjunction with the following drawings, in which:

FIG. **1** is a block diagram of a computer system incorporating the present invention;

FIG. **2** is a block diagram of the CPU board of FIG. **1**;

FIG. **3** is a more detailed block diagram of the distributed system peripheral (DSP) of FIG. **2**;

FIG. **4** is a simplified flow chart of the power up sequence for finding an operational CPU and FIG. **5A**, **5B**, **6**, **7**, **8**, **9**, **10**, **11**, **12**, **13**, **14** and **15** are flowcharts illustrating transferring control of the computer system to an operational CPU based on the occurrence of critical errors.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Referring now to FIG. **1**, a computer system C is shown. The computer system C includes four CPU boards **20**, **22**, **23**, **25** connected to a host bus **24** and a memory controller **30**, which is also coupled to the host bus **24**. Although four CPUs **20**, **22**, **23**, **25** are used in the preferred computer system C, the present system functions for any multiprocessor system. Each CPU occupies a physical slot in the computer system by which the CPUs **20**, **22**, **23**, **25** are designated. Thus, CPU **P0 20** is the CPU that occupies physical slot **0** in the system. The CPUs are designated CPU **P0 20**, CPU **P1 22**, CPU **P2 23**, and CPU **P3 25**, in reference to each CPU's physical location.

A main memory array **32**, preferably comprised of dynamic random access memory (DRAM) modules, is coupled to the memory controller **30**. Memory mapper logic **34** is coupled to the host bus **24**, the memory controller **30**, and the memory array **32**. The memory mapper logic **34** provides memory mapping functions to facilitate memory accesses to the memory array **32**.

A bus controller **40**, preferably an EISA bus controller (EBC), is coupled between the host bus **24** and an expansion bus **42**, preferably an Extended Industry Standard Architecture (EISA) bus. The EBC **40** provides various bus cycle translation and conversion functions to facilitate transfers between the host bus **24** and the EISA bus **42**. A system data buffer (SDB) **44** is also coupled between the host bus **24**, the memory array **32** and the EISA bus **42**. A logic block referred to as the common system peripheral (CSP) **46** is coupled between the host bus **24** and the EISA bus **42**. The CSP **46** is also coupled through a MUX bus **50** to logic blocks referred to as distributed system peripherals (DSPs) **126** in each of the CPUs **20**, **22**, **23**, **25**. The CSP **46** includes various system components including a direct memory access (DMA) controller, an EISA arbitration controller, and numerous system board logic functions like memory refresh control and interrupt logic.

The EISA bus **42** includes a plurality of EISA slots **52** and **54** for receiving EISA expansion cards, like a network interface card or a hard disk interface card. The EISA bus **42** is coupled through buffers **56** to a bus referred to as the X bus **60**. A number of peripheral devices are coupled to the X bus **60** including the keyboard controller **62**, a real time clock (RTC) and CMOS memory **64**, an electrically erasable programmable read only memory (EEPROM) **66** used to store the BIOS and system functions, a floppy disk controller **68**, and a peripheral controller chip **70**, which includes numerous ports and UARTs (universally asynchronous receiver/transmitters).

The MUX bus **50** between the CSP **46** and the various DSPs **126** associated with the CPUs **20**, **22**, **23**, **25** includes an XAD bus. The XAD bus is an 8-bit bi-directional data bus on which interrupt, I/O, and special cycle data is transferred. The MUX bus **50** further includes a cycle status bus called the XST bus. This three-bit bus is controlled by the CSP **46** and defines the cycle currently being transferred on the 8-bit MUX bus **50**. The MUX bus **50** further includes a two-bit XBP bus, which is an input bus on which DMA byte pointer information is received from the DSPs **126**. Using the XAD, XST, and XBP buses, the CSP **46** interfaces with the DSPs **126** to provide interaction between the general computer system C and the CPUs **20**, **22**, **23**, **25**. The MUX bus **50** is a broadcast bus that transmits all signals to all of the CPUs **20**, **22**, **23**, **25**, and each CPU **20**, **22**, **23**, **25** determines whether it is the CPU being accessed.

For more information on the operation of the MUX bus **50**, please see related copending application Ser. No. 07/955,482 entitled MULTIPLEXED COMMUNICATION PROTOCOL BETWEEN CENTRAL AND DISTRIBUTED PERIPHERALS IN MULTIPROCESSOR COMPUTER SYSTEMS, filed on Oct. 2, 1992, which is hereby incorporated by reference. For more information regarding the CSP **46** and the DSPs **126**, please see related copending application Ser. No. 07/955,683, entitled ARRANGEMENT OF DMA, INTERRUPT AND TIMER FUNCTIONS TO IMPLEMENT SYMMETRICAL PROCESSING IN A MULTIPROCESSOR COMPUTER SYSTEM, filed on Oct. 2, 1992, which is hereby incorporated by reference.

The CSP **46** provides an interface between the MUX bus **50** and the EISA bus **42** and X bus **60** of the computer system C. Various transfers and operations are defined on the MUX bus **50** using the XST bus to define the transfer and the XAD bus to identify addresses, data, and interrupts. The CSP **46** implements a MUX bus master interface to assert interrupts and transfer data to and from the DSPs **126**. The CSP **46** in the present system also includes power up timer logic. When the system is powered up, a power up timer begins counting. If CPU **20** in physical location zero does not write to a particular DSP port accessed via the CSP **46** within a specific interval, which is two seconds in the preferred embodiment, the CSP **46** asserts a power up timeout (PUTO) signal to all of the DSPs **126**. The PUTO signal is generated as a miscellaneous data transfer on the MUX bus **50**. As described below, the PUTO signal causes the current CPU to be designated as inoperative and initiates testing of the next CPU.

Referring now to FIG. **2**, a block diagram of CPU **P0 20** is shown. For the purposes of this application, all of the CPUs **20**, **22**, **23**, **25** are substantially identical. In the following description, CPU **P0 20** is described for simplic-

ity, and the following description applies equally well to the other CPUs **22**, **23**, **25**. CPU P**0** **20** includes a processor **102** which preferably includes an internal cache **104**. The processor **102** is preferably the Intel i**486** processor preferably running at 33/66 MHz or Pentium processor preferably running at 66 MHz. However, the use of other types of processors is also contemplated. The processor **102** is coupled to a processor bus **106** including control, data and address portions, as shown.

The CPU P**0** **20** further has a cache memory system which includes an array of cache RAMs **110**, a cache controller **108**, and various cache interface logic **120** and buffers **124** to interface between the processor bus **106** and the CPU components. In the preferred embodiment, the cache RAMs **110** are C8 or 82490 cache RAMs produced by Intel Corporation, and the cache controller **108** is a C5 or 82495 cache controller, also produced by Intel. The cache controller is also connected to an address transceiver **122** for transmitting addresses to and from the host bus **24**. The transceiver **122** receives control signals from the cache interface logic **120**.

The cache data buffer is linked not only to the host bus, but is also connected by a local I/O bus **125** to the DSP **126**. The DSP **126** implements various logic functions that are closely related to the processor/cache system. Referring now to FIG. **3**, the DSP **126** includes interrupt controller logic **140**, preferably comprising two cascaded Intel 8259 interrupt controllers which provide 15 levels of interrupts. The DSP **126** also includes non-maskable interrupt (NMI) logic **144** and multiprocessor interrupt sharing logic **146**. The multiprocessor interrupt sharing logic **146** allows a CPU to interrupt other CPUs at any interrupt level in one I/O access, thus providing an efficient multiprocessor communication capability. Also included are reset logic **148**, DMA (direct memory access) logic **150**, numeric coprocessor logic **153**, timer clock generation logic **154**, and dual port I/O interface logic **156**. The dual port I/O interface logic **156** interfaces the various registers and logic in the DSP **126** with the local I/O bus **125** and the MUX bus **50**.

CPU processor I/O logic **142** on the DSP includes a processor control port, and processor slot identification logic **152** includes a CPU Identification (WHOAMI) port, an Index CPU port, and a Logical CPU Assignment (LCA) port. The processor control port controls miscellaneous processor functions, like cache enabling and flushing and processor interrupt signals. In particular, the processor control port enables and disables the SLEEP, signal **158** to the microprocessor **102**. When the system is power cycled, the SLEEP* signal **158** is disabled until the processor control port is written and the SLEEP, signal **158** is enabled. Any device, including the microprocessor **102** itself, may write to a CPU's processor control port to change the sleep bit in the port, which causes the SLEEP* signal **158** to be driven low by the DSP **126**. The SLEEP* signal **158** is asserted low as long as the value of the sleep bit in the processor control port remains **1**.

The SLEEP* signal **158** is provided to the cache interface logic **120**. In the preferred embodiment, the SLEEP* signal **158** causes a hold request to be asserted to the microprocessor **102**. When the hold request is acknowledged, the internal **486** first level cache **104** and the external second level C**5** cache **110** are flushed. After the flush is completed, the hold request signal remains asserted so that the microprocessor **102** stays inactive. The hold request signal is maintained as long as the SLEEP* signal **158** is asserted. Thus, to activate the microprocessor **102**, the processor control port of the DSP **126** is again written to disable the sleep state, which causes the SLEEP* and the microprocessor hold request signals to be negated, and the microprocessor **102** resumes operation.

Processor slot identification logic **152** consists of the logic required for physical CPU identification and logical CPU assignment. The logical CPU assignment is an arbitrary designation of a particular CPU, and each CPU may be addressed by its physical location or logical assignment using the MUX bus **50**. One of the ports in the processor slot identification logic **152** is the Index CPU port, which is used to select the physical CPU to access when the LCA port is accessed, or to select the logical CPU to access when other local ports are accessed from the EISA bus. Local ports can be accessed directly or via an Index Address port in the CSP. The Index CPU port must be written before being used to assure its contents. If the indicated CPU matches the assignment indicated by the DSP's **126** WHOAMI port, the CPU performs the local operation. Otherwise, the operation from the MUX bus **50** is ignored.

Each DSP **126** includes a Logical CPU Assignment (LCA) port. The LCA port is used by the software, as discussed below, to assign logical CPU designations to physical CPU locations. It is also used to determine the physical location of a logical CPU. Each CPU's LCA port is accessible to all elements of the system C, but is not locally accessible but is accessible only over the EISA bus **42** through the CSP **46**. The port contains CPU assignment bits used to assign the logical CPU number to the physical CPU location. The LCA port works in conjunction with the Index CPU port to access a particular physical CPU. Before the LCA port is written, the Index CPU port must be written with the desired CPU's physical location. When the LCA port is written on the next operation, the CPU having the indicated physical designation, instead of logical assignment, is accessed.

The WHOAMI port stores the logical CPU designation for the CPU and may only be read via the local bus **125**. When accessed, the WHOAMI port responds with a number indicating the individual CPU's logical designation. The CPU's logical designation is determined when the system C is powered up, as discussed below. When the system C is power cycled, the WHOAMI port initially receives a value corresponding to the CPU's physical location. By accessing this port, each CPU determines its own logical CPU assignment, and therefore determines whether it is the particular CPU being addressed on the MUX bus **50**.

Each CPU also receives a separate set of three SLT signals **161**. The SLT signals **161** are permanently connected to particular signals to correspond to individual slots. For example, the SLT signals **161** for physical CPU slot **0** always assert **000**, while those for slots **1** and **2** always assert **001** and **010**, respectively. The SLT signals **161** are provided to the processor slot identification logic **152** on the DSP **126** so that each CPU may determine its physical slot location.

The processor slot identification logic **152** further receives a ROTATE signal **160** generated by the dual port I/O interface logic **156**. The interface logic **156** asserts the ROTATE signal **160** when the PUTO signal is asserted on the MUX bus **50**. When the ROTATE signal **160** is asserted, the processor slot identification logic **152** decrements the CPU's logical assignment in the WHOAMI port. If the CPU's logical CPU assignment is **0** when the ROTATE signal is asserted, the WHOAMI port value is set to FFh, indicating a non-functioning CPU. If the CPU's logical assignment is FFh when the ROTATE signal **160** is asserted, the ROTATE signal is ignored.

If the CPU's logical assignment rotates to **0**, the processor slot identification logic **152** asserts a WAKE signal **165** to the interface logic **120**. The WAKE signal **165** causes the interface logic **120** to change the processor control port's sleep bit in the processor I/O logic **142**. As a result, the SLEEP* signal **158** is negated, and the CPU begins operation.

Referring now to FIG. **4**, the logical configuration of the CPUs is established after the computer system is power cycled. When the system is powered up, the computer system C resets in step **300** and, in step **302**, the DSP inputs the SLT signals to the WHOAMI port, so that each CPU receives an initial logical designation corresponding to its physical location. At the same time, each DSP asserts the SLEEP* signal on its CPU except for CPU **P0**, which remains awake and begins its power-on self test (POST) routine. In an alternative embodiment, the DSP for CPU **P0 20** may also enable its sleep bit, but CPU **P0 20** is then immediately awakened. When the SLEEP* signal is asserted for each CPU, all of the CPUs **22**, **23**, **25**, except for CPU **P0 20**, become inactive. Meanwhile, the power up timer in the CSP **46** loads a preset interval on reset and begins counting when the reset signal is negated. In the preferred embodiment, the preset interval is approximately **2** seconds. The power up timer counts until it is disabled or times out.

The timer may be disabled by a write operation to the LCA port. In step **304**, the CSP **46** determines whether a write operation to the LCA port has occurred before the timer times out. If CPU PO **20** is operating properly, the POST program directs the CPU **P0 20** to execute a write operation to the LCA port. The write operation to the LCA port is detected by the CSP **46**, which disables the timer in step **306**.

Next, in step **308**, the operational CPU, now designated CPU **L0**, completes its POST program and boots up the rest of the system C using standard boot software. In particular, CPU **L0** wakes up and boots the remaining CPUs **22**, **23**, **25**, with each CPU executing software to test itself and to assign an appropriate logical CPU designation. This process can conveniently be done by placing a special vector, which is interrogated early in the POST program, to special wake up code, which includes a handshake to CPU **L0**, so that each CPU can be brought up individually and indicate its status. A more detailed description is provided below. For more details on the basic operation of the startup and handshake procedure, please refer to application Ser. No. 07/431,743, originally entitled "COMMON RESET ROM", filed on Nov. 3, 1989, and continued as Ser. No. 08/051,601, filed on Apr. 22, 1993 and retitled "METHOD AND APPARATUS FOR RESETTING MULTIPLE PROCESSORS USING A COMMON ROM" which is hereby incorporated by reference. If one of the CPUs **22**, **23**, **25** is not functional as indicated by a failure to handshake within a given time, it is so designated in its DSP and the next consecutive functioning CPU is designated with the next logical CPU assignment. If the slot is empty, the logical CPU assignment sequence continues at the next slot occupied by an operational CPU. In addition to booting up the other CPUs, the designated CPU **L0** boots up the rest of the computer system C, and operates as CPU **L0** for all unique CPU **L0** functions. As noted, this sequence is described in more detail below, particularly when intermittent critical errors have occurred on the first CPU.

If the write operation to the LCA port does not occur before the timer times out, the CSP **46** generates the PUTO signal in step **310** and it is transferred onto the MUX bus **50**. When the PUTO signal is received by the DSPs **126**, each DSP asserts the ROTATE signal **160** in step **312** and the logical assignment in the WHOAMI port of each CPU is decremented by 1. The current CPU **L0** that failed to write the LCA port within the timer interval is assigned a value in its DSP **126** of FFh, indicating that the CPU is inoperative. A CPU having a logical assignment of FFh does not have its slot identification decremented when the power up timeout signal is asserted.

After the logical CPU assignments have been rotated, the new CPU **L0** is awakened by the WAKE signal and begins the same POST code described before, which includes the write to the LCA port. The power up timer is restarted, and the process returns to step **304**. The procedure repeats until an operational CPU writes to the LCA port before the timer times out and boots the rest of the system. If none of the CPUs are operational, the system will not boot up.

Referring now to FIG. **5A**, a more detailed version of the RESET sequence is shown. In this case the COLD RESET sequence **400** commences at step **402** where the BIST or basic internal self test value provided by the processor of CPU **P0 20** is stored for later examination. Control proceeds to step **404** which is a wait period until the RTC **64** becomes fully active. Control then proceeds to step **406**, which is a call to the CHECK_WELL_PROCS sequence **500** (FIG. **6**) to determine and update the failed flags. Preferably a series of flag bytes is maintained in the CMOS portion of the RTC **64** to store certain variables. One byte is used to indicate which processors are present, another to indicate which have failed. Control then proceeds to step **408**, which is a call to the CHK_FAIL_PROCS sequence **550** (FIG. **7**) which actually checks to see if any failed processors are present and if the current processor is a failed processor. Assuming that the current processor is not a failed processor, control proceeds to step **410** to determine if the CPU BIST value as stored in step **402** indicates that there are no problems. If there are problems, control proceeds to step **412** where the specific error is indicated, is logged in the wellness log and in environment variables. The wellness log is a running log contained preferably in the extended CMOS are contained in the RTC **64** or in another nonvolatile RAM source in the computer system C which keeps track of all of the errors which have occurred in the computer system C.

In most cases during run time operations, these errors generate an NMI or nonmaskable interrupt, with the NMI sequence then logging the error in the wellness log so that a maintenance record can be maintained. The environment variables are also areas in the extended CMOS area and which are used to set flags and pass information as to the state of the computer or for configuration values. In this case the error is just logged in an environment variable that indicates an error has occurred. The wellness log is used for more complete information. For more details, please refer to Ser. No. 07/955,849, entitled "COMPUTER FAILURE RECOVERY AND ALERT SYSTEM", filed on Oct. 2, 1992 and hereby incorporated by reference.

Control proceeds to step **414** after step **412** or if the CPU was okay in step **410**. In step **414** the cache controller system **104** present on the CPU is checked to determine if its basic internal self test value was okay. If not, control proceeds to step **416** where the error is logged in the wellness log and in the environment variable. If it was okay in step **414** or after **416**, control proceeds to step **418** where the CHECK_WELL_PROCS sequence **500** is again called. This call is to reupdate the failed status based on the errors which may have been determined in steps **410** and **414**. Control then proceeds to step **420**, where the CHK_FAIL_PROCS sequence **550** is called to shutdown a processor if necessary.

Control proceeds to step **422** where physical CPU**0** is set to indicate to be a logical CPU**0**. This is done by a write operation to the LCA port as described above. This operation will have the effect of disabling the timer so that automatic, hardware-based CPU L**0** rotation does not occur.

Control then proceeds to step **424** to determine if the checksum in the EEPROM **66** is okay. This is a check of the integrity of the EEPROM **66**. If it checks okay, control proceeds to step **426** to determine if the 8042 keyboard controller **62** is okay. If not, control returns to step **402** and operations repeat. If the checksum did not validate in step **424**, control proceeds to step **428** where an error is indicated by the appropriate number of beeps. If the keyboard controller **62** was okay in step **426** or after step **428**, control proceeds to step **432** where values are written to ensure that all of the other CPUs in the system are asleep so that they do not interfere with the POST and booting process. Control then proceeds to step **434** (FIG. 5B) where miscellaneous initialization operations occur. After a number of these operations, control proceeds to step **436** where the call ASSIGN_CPUs sequence **600** (FIG. **8**) is called. Control then proceeds to step **438**, where more miscellaneous initialization operations occur. Control proceeds to step **440** where the CHECK_PROC_OK sequence **650** (FIG. **9**) is called, which checks to see if any CPUs have failed and if so provides a message. Control then proceeds to step **442** where the INIT_PROC_AVAIL sequence **700** (FIG. **10**) is called. This sequence **700** sets available bits in a stored flag byte for each of the CPUs after physical CPU**0**, as that is the one assumed to be operating. Control then proceeds to step **444**, where the POST_INIT sequence **720** (FIG. **11**) is called. This sequence **720** is used to start and initialize any secondary processors, which are processors other than CPU L**0**, in the system. Control then proceeds to step **446** where more miscellaneous initialization operations occur. Control proceeds to step **448** where the BOOT_CPU sequence **850** (FIG. **14**) is called. This sequence **850** is used to check if the CPU currently executing has developed errors during the initialization operations and thus is sufficiently dependable to continue the boot process. Control proceeds to step **450** where miscellaneous initialization and POST or power-on self test operations occur. Ultimately control proceeds to step **452**, which is the booting process of the computer system C.

It is noted that on the preferred microprocessors, step **422** will be executed in well less than the hardware CPU rotation time, but that any steps after step **422** are assumed to have occurred well after the rotation time. Particularly, step **448** will have occurred well after the rotation time.

The CHECK_WELL_PROCS sequence **500** (FIG. **6**) commences operation at step **502** where a pointer is set to the beginning of the wellness log. Control proceeds to step **504** to get the wellness entry indicated by the pointer. In step **506** a determination is made whether the wellness entry indicates a cache parity error. If not, control proceeds to step **508** to determine if it indicates a CPU parity error. If not, control proceeds to step **510** to determine if the entry indicates that a processor POST error has occurred, that is an error during the POST procedure described below. If not, then none of these critical errors have occurred, and the particular CPU is considered sufficiently reliable to continue processing. Control proceeds to step **512**, where the pointer value is incremented. At step **514** a determination is made whether the last entry in the wellness log has been checked. If not, control returns to step **504** and the next entry is obtained. If so, control proceeds to step **516**, which is a return from the sequence **500**.

If errors have been indicated in steps **506**, **508** or **510**, control proceeds to step **518** to determine the physical CPU number of the CPU indicating the failure. Control then proceeds to step **518** to determine if a failed flag is set for that particular CPU. Preferably this would have been set when the error occurred during normal operations when the wellness entries were actually made, but it could have occurred at other times as seen below. If the flag is not set, control proceeds to step **522** where it is set for that particular processor and control then proceeds to step **512**. If the flag was already set, control proceeds from step **520** to step **512**. Thus the sequence **500** scans the wellness log and indicates CPU failure for certain critical errors which have previously occurred.

The CHK_FAIL_PROCS sequence **550** commences at step **552** where a value indicating which CPUs are installed is obtained. This can be obtained by referencing a value contained in the CMOS memory or can be done by checking each processor slot. Control proceeds to step **554** where a value indicating the failed CPUs is obtained, such as the value developed in the CHECK_WELL_PROCS sequence **500**. Preferably the installed and failed CPU values are contained in the environment variable area, with particular bits indicating particular processors. Control proceeds to step **556**, where all of the failed CPUs are removed from the installed value. Control then proceeds to step **558** to determine if any CPUS are indicated as remaining. If not, control proceeds to step **560**, which is a return from the sequence **550**. If so, control proceeds to step **562** to determine if the current physical CPU is installed and has not failed. The current physical CPU is determined by doing a call to the appropriate location as described above and the failed flag is then compared. If it is installed and has not failed, this indicates that the CPU is considered sufficiently reliable and control proceeds to step **560**.

If the failed flag is set, then the control essentially proceeds to a closed loop, commonly referred to as JMP $ instruction. This is quite proper in that the CHK_FAIL_PROCS sequence **550** is called from the COLD RESET sequence **400** well within the **2** second period used by the CPU rotation timer to indicate the need for the hardware to rotate the logical CPU value. Thus in this manner if the computer C goes through a cold reset, i.e. a power up condition, then its failed status of the booting processor is quickly determined and if certain critical errors have previously occurred, the CPU simply shuts down and the hardware proceeds to rotate the logical CPU number so that the booting process can continue. As discussed above, the reason for this rotation of the CPUs is that in many cases it may be possible for a processor to actually complete the entire POST and boot process before it fails. This is particularly true where parity error occurred as they are relatively infrequent and if they do occur, the consequences are relatively extreme. Thus it is considered that a CPU which has previously had a parity error in the processor itself or in the cache system is not sufficiently dependable over the long term to allow it to fully operate the computer C.

The ASSIGN_CPUS sequence **600** (FIG. **8**) commences at step **602** where the installed CPUs are obtained. Control proceeds to step **604** where the list of failed CPUs is obtained. Control proceeds to step **606** to determine if there are any good CPUs. If not, an error flag is set in step **608** and control proceeds to step **610**, which is a return from the sequence **600**. If there are good CPUs, control proceeds from step **606** to step **612**, where logical CPU numbers are assigned to nonfailed physical CPUs in physical order. Control then proceeds to step **610**.

The CHK_PROC_OK sequence **650** (FIG. **9**) commences at step **652** to determine if CPU P**0** is equal to CPU L**0**. That is, is the physical **0** CPU also the logical CPU **0**, indicating that it is apparently properly booting the system and has not previously failed. If not, control proceeds to step **651** to determine if CPU P**0** is actually installed. If not, control proceeds to step **654** where a no CPU P**0** error message is provided to the user. If it is installed, control proceeds to step **656** where the failed bit is set in the failed byte to indicate the CPU P**0** has failed. If the CPU P**0** is equal to CPU L**0** in step **650** or after steps **654** and **656**, control proceeds to step **658**, where the failed byte is obtained. Control then proceeds to step **660** where the installed byte value is obtained. Control proceeds to step **662** where an error message is provided to the user to indicate any failed and installed CPUs. Control then proceeds to step **664**, which is a return.

The INT_PROC_AVAIL sequence **700** (FIG. **10**) has a step **702**. In step **702**, the computer system C reads the identifications for each CPU slot after physical **0**, as physical **0** is assumed to be the one booting, and if there is a value, then a CPU is present and the available bit is set in the available byte. Control then proceeds to step **704** which is a return from the sequence.

The POST_INT sequence **720** commences at step **722** to determine if there are any secondary processors. If not, control proceeds directly to step **724**, which is a return, as there is nothing to initialize. If a secondary processor is present, control proceeds to step **726**, where a pointer is set to indicate CPU L**1**. Control proceeds to step **728** to determine if the cache for that particular CPU had a BIST error. If not, control proceeds to step **730** where the POST_PROCESS sequence **750** (FIG. **12**) is called. This sequence **750** initializes operation of the secondary processor. After step **730**, control proceeds to step **738**. If there was an error, control proceeds to step **732** where an error message is displayed. In step **734** the error is logged in both the wellness log and the environment variable. Control proceeds to step **736** where the logical CPU number is incremented. Control proceeds to step **738** where the physical CPU slot number is incremented. Control proceeds to step **740** to determine if this was the last slot. If not, control returns to step **728** where the process is repeated. If it was the last slot, control proceeds to step **724**.

The POST_PROCESS sequence **750** commences at step **752**, where a vector value is provided at the address 40:67h to a sequence called INIT_PROCESS or RESET **800** (FIG. **13**). The 40:67h vector sequence is described in the referenced applications, but briefly is a location previously defined in IBM PC compatibles to receive a vector location used under certain conditions after a reset is issued to the processor. The original reason for this vector was to allow an 80286 processor to return from protected mode to real mode, but the vector location is now utilized for other operations. Control then proceeds to step **754** where the particular CPUs SLEEP bit is cleared. This will allow the CPU to begin operations. Control proceeds to step **756**, where a period of time equal to an initialization delay is passed. This time period is provided to allow a certain period to have passed for the particular CPU to begin operations. Control proceeds to step **758** to determine if the CPU has indicated that it has started the initialization procedure. If so, control proceeds to step **760**, where a further period of time elapses. Control proceeds to step **762** to determine if a flag has been set by the CPU which is initializing to indicate that the CPU in initialization operation has finished. If so, this is an indication the CPU has fully initialized, so control proceeds to step **764** where it is put back to sleep pending execution for further tasks.

If the CPU did not indicate START in the allotted time or it did not indicate finished in the allotted time, control proceeds from steps **758** and **762** to step **768**, where the CPU is put back to sleep and a reset condition is indicated. Control proceeds to step **768**, where an error message is displayed and then to step **770** where the error is logged in the wellness log and environment variable. Control then proceeds to step **772**, which is a return from the sequence **750**, which is also where execution proceeds after step **764**.

The INIT_PROCESSOR_RESET sequence **800** commences at step **802**, where the CMOS_RST_CODE value is obtained. As noted above, 40:67h is utilized for certain reset operations as a vector pointer. The way this is indicated is that a value contained in the CMOS memory portion of the RTC **64** is utilized to indicate the particular type of reset operation occurring. If a certain type is occurring, then a vector from 40:67h is executed, so that a more complete initialization and reset sequence is not executed. Control proceeds to step **804** to determine if the 40:67h vector reset is in effect. If not, control proceeds to step **806** to determine if this sequence **800** is being executed on a secondary processor. If not, control proceeds to step **808** which is the normal reset sequence. Effectively step **808** causes an entry into step **432** of the COLD RESET sequence **400**. If vectored reset was indicated in step **804** or if this was a secondary processor in step **806**, control proceeds to step **810**, where the STARTED flag is set to inform the other CPU in step **758** that the CPU initialization operation has started. Control proceeds to step **812** where initialization steps occur. Control then proceeds to step **814** where the INIT_DONE flag is set to indicate to the other CPU that initialization has completed. Control then proceeds to step **816**, where the CPU puts itself to sleep.

The BOOT_CPU sequence **850** commences at step **852**, where the ASSIGN_CPUS sequence **300** is called. Control then proceeds to step **854** where the failed byte is obtained and to step **856**, where the installed byte is obtained. Control proceeds to step **858**, where the two values are combined to develop an installed and not failed list. Control then proceeds to step **860** where the physical and logical CPU numbers for the particular processor on which the sequence is operating are obtained. Control then proceeds to step **862** to determine if the physical CPU as indicated by step **860** is equal to the first good CPU as indicated in the combined installed and not failed value provided in step **858**. If so, this is an indication that everything is satisfactory and control proceeds to step **864**, which is a return from the sequence. If the physical CPU is not the first good CPU, this is an indication that an error has previously occurred on CPU P**0** such that it can no longer be considered reliable. Then in step **863** the state of all the registers in the particular CPU are stored for later recall. Control then proceeds to step **866**, where a message is displayed indicating control of the system is switching from one CPU to another active CPU in the system. Control then proceeds to step **868**, where the vector pointing to the SWITCH_START sequence **880** (FIG. **15**) is stored at 40:67h. Control proceeds to step **870** where operation is set up to perform a vectored reset from 40:67. This would include probably setting the CMOS values. Control proceeds to step **872**, where the SLEEP bit in CPU L**0** is cleared. By definition, this sequence can not be operating on CPU L**0** as the ASSIGN_CPUS sequence **300** executed in step **852** would not have assigned a logical value to this particular CPU as it has been indicated as being failed. Thus by clearing the SLEEP bit in logic in CPU L**0**,

the first active processor will actually have been activated. Control then proceeds to step **874** to determine if the new CPU has started operations by looking for the START flag. If so, control proceeds to step **876** where the CPU puts itself to sleep. If it has not started after a proper time, control proceeds to step **878** where the new CPU is put back to sleep and reset. Control then proceeds to step **864**.

The SWITCH_START sequence **880** commences at step **882**, where the CPU state which had been saved by the prior processor in step **864** is restored, so that effectively the new processor has the same state as the prior processor so operation can commence at effectively same point. Control proceeds to step **884** to determine if the old CPU is asleep yet. Control remains at step **884** until it has. Once it has gone to sleep, control proceeds to step **886**, where a message is provided to indicate that the switch has occurred. Control then returns in step **888**. A return is proper because by the state being restored, operation is able to be continued as though there was no processor switch. Effectively the return step **888** is the same as the return step **864** and returns operation to proceed at step **450**. Thus in the BOOT_CPU sequence **850** the processors are rotated automatically based on the critical error log for errors which occurred during later initialization operations and had not occurred by the initial operations. The switch is done using the BOOT_CPU sequence **850** because step **448** occurs well after the preferred **2** second time for the hardware CPU rotation.

Therefore the computer system according to the preferred embodiment not only automatically rotates CPUs to the first fully operational CPU in case of major hardware failure of the boot processor but also transfers control should the processor have developed certain critical errors during its later operation such that it would be considered unreliable and undependable for long term operation. By this further improvement it is clear that the system will boot completely on only fully functional and dependable CPUs, thus reducing user aggravation and avoiding the potential loss of data.

The foregoing disclosure and description of the invention are illustrative and explanatory thereof, and various changes in the size, shape, materials, components, circuit elements, wiring connections and contacts, as well as in the details of the illustrated circuitry and construction and method of operation may be made without departing from the spirit of the invention.

We claim:

1. A method for booting a multiprocessor computer system with a processor, comprising the steps of:

maintaining a log of predetermined critical errors which have occurred on the computer system;

resetting the computer system and beginning execution of initialization operations on a first processor of the multiprocessor computer system;

determining if said first processor has previously had one of said predetermined critical errors; and

if one of said predetermined critical errors had occurred, disabling said first processor and transferring execution to a second processor of the multiprocessor computer system to perform initialization operations.

2. The method of claim **1**, further comprising the step of:

transferring execution from the first processor to the second processor if the first processor has not performed a predetermined one of said initialization operations in a predetermined time, and

wherein said step of determining if said first processor has previously had one of said predetermined critical errors occurs before said predetermined time, and

wherein said step of disabling said first processor occurs before said predetermined time and includes halting operation of said first processor before performing said predetermined operation so that said step of transferring execution if the first processor has not performed said predetermined operation causes transfer of execution.

3. The method of claim **2**, further comprising the steps of:

if none of said predetermined critical errors had occurred, performing said predetermined operation;

performing tests on said first processor after performing said predetermined operation and logging a critical error if one occurs;

determining if one of said tests indicates one of said predetermined critical errors has occurred;

if one of said predetermined critical errors has occurred, storing the processor state, activating said second processor and halting operation of said first processor; and

restoring said stored processor state to said second processor and performing the remaining initialization operations on said second processor.

4. The method of claim **1**, wherein said step of determining if said first processor has previously had one of said critical errors includes the steps of:

performing tests on said first processor and logging a critical error if one occurs; and

determining if one of said tests indicates one of said predetermined critical errors has occurred, and

wherein said step of disabling said first processor includes the steps of:

if one of said predetermined critical errors has occurred, storing the processor state, activating said second processor and halting operation of said first processor; and

restoring said stored processor state to said second processor and performing the remaining initialization operations on said second processor.

5. A system for booting a multiprocessor computer system with a reliable processor, the system comprising:

means for detecting and maintaining a log of predetermined critical errors which have occurred on the computer system; and

means for initializing the computer system including:

means for resetting the computer system and beginning execution of initialization operations on a first processor of the multiprocessor computer system;

means for determining if said first processor has previously had one of said predetermined critical errors; and

means for disabling said first processor and transferring execution to a second processor of the multiprocessor computer system to perform initialization operations if one of said predetermined critical errors had occurred to said first processor.

6. The system of claim **5**, wherein said means for initializing the computer system further includes:

means for transferring execution from the first designated processor to the second designated processor if the first designated processor has not performed a predetermined one of said initialization operations in a predetermined time; and

wherein said means for determining operates before said predetermined operation is performed, and

wherein said means for disabling halts operation of said first processor before said predetermined operation is performed.

7. The system of claim **6**, wherein said means for initializing the computer system further includes:

means for performing tests on said first processor and logging a critical error in said log if one occurs;

means for determining if one of said tests indicates one of said predetermined critical errors has occurred;

means for storing the processor state of said first processor and halting operation of said first processor and for activating said second processor if one of said predetermined critical errors has occurred; and

means for restoring said stored processor state to said second processor and performing the remaining initialization operations on said second processor.

8. The system of claim **5**, wherein said means for determining further includes:

means for performing tests on said first processor and logging a critical error in said log if one occurs; and

means for determining if one of said tests indicates one of said predetermined critical errors has occurred; and

wherein said means for disabling further includes:

means for storing the processor state of said first processor, activating said second processor and halting operation of said first processor if one of said predetermined critical errors has occurred; and

means for restoring said stored processor state to said second processor and performing the remaining initialization operations on said second processor.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 5,491,788
DATED : February 13, 1996
INVENTOR(S) : Darren J. Cepulis et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

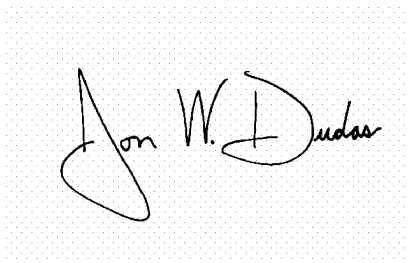
Title page,
Item [54], Title, delete "**TRANSFERRING**" and insert therefor -- **TRANSFERRED** --.

Column 3,
Line 27, delete "CPU" and insert therefor -- CPU; --.
Line 27, the sentence starting with "FIG. 5A," should start a new paragraph.

Column 14,
Line 39, before "processor," delete "reliable".

Signed and Sealed this

Twenty-ninth Day of November, 2005



JON W. DUDAS
Director of the United States Patent and Trademark Office

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re patent of Michaelis *et al.*	§	Attorney Docket No.: 185945.012600
	§	
U.S. Patent No. 7,624,302	§	Customer No.: 67395
	§	
Issue Date: November 24, 2009	§	
	§	
Filing Date: October 25, 2004	§	
	§	
For: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR	§	

REQUESTER'S POWER OF ATTORNEY
PURSUANT TO 37 C.F.R. § 1.510(f)

Pursuant to 37 CFR 1.510(f) Requester, **Unified Patents, LLC**, hereby appoints the Practitioner(s) associated with **Customer Number 67395**, as its attorney(s) to prosecute and to transact all business in the United States Patent and Trademark Office in connection with the above-identified request for *Ex Parte* Reexamination. Please recognize the correspondence address (including any electronic mail address) associated with **Customer Number 67395** for the above-identified *Ex Parte* Reexamination proceeding number to be the address associated with the appointed counsel(s).

The undersigned is authorized to sign this Power of Attorney on behalf of the Requester.

Executed on 8/28/2023

Unified Patents, LLC
By: Jonathan Stroud
Printed Name: Jonathan Stroud
Title: General Counsel

2. No Interrupt Assignment Entries are declared for any of the bus source interrupts, and the operating system uses some other bus-specific knowledge of bus interrupt schemes in order to support the bus. This operating system bus-specific knowledge is beyond the scope of this specification.

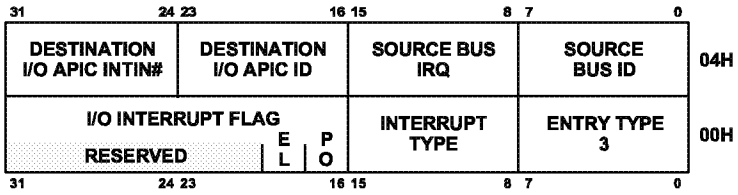


Figure 4-7. I/O Interrupt Entry

Table 4-10. I/O Interrupt Entry Fields

Field	Offset (in bytes:bits)	Length (in bits)	Description
ENTRY TYPE	0	8	Entry type 3 identifies an I/O interrupt entry.
INTERRUPT TYPE	1	8	See Table 4-11 for values.
PO	2:0	2	Polarity of APIC I/O input signals: 00 = Conforms to specifications of bus (for example, EISA is active-low for level-triggered interrupts) 01 = Active high 10 = Reserved 11 = Active low Must be 00 if the 82489DX is used.
EL	2:2	2	Trigger mode of APIC I/O input signals: 00 = Conforms to specifications of bus (for example, ISA is edge-triggered) 01 = Edge-triggered 10 = Reserved 11 = Level-triggered
SOURCE BUS ID	4	8	Identifies the bus from which the interrupt signal comes.
SOURCE BUS IRQ	5	8	Identifies the interrupt signal from the source bus. Values are mapped onto source bus signals, starting from zero. A value of 0, for example, would indicate IRQ0 of an ISA bus. See Section D.3 for PCI bus semantics.
DESTINATION I/O APIC ID	6	8	Identifies the I/O APIC to which the signal is connected. If the ID is 0FFh, the signal is connected to all I/O APICs.
DESTINATION I/O APIC INTIN#	7	8	Identifies the INTIN*n* pin to which the signal is connected.

Table 4-11. Interrupt Type Values

Interrupt Type*	Description	Comments
0	INT	Signal is a vectored interrupt; vector is supplied by APIC redirection table.
1	NMI	Signal is a nonmaskable interrupt.
2	SMI	Signal is a system management interrupt.
3	ExtINT	Signal is a vectored interrupt; vector is supplied by external PIC. For example, if an 8259 is used as the external PIC, the source is the 8259 INTR output line, and the vector is supplied by the 8259.

* All other values are reserved.

4.3.5 Local Interrupt Assignment Entries

These configuration table entries tell what interrupt source is connected to each local interrupt input of each local APIC. Figure 4-8 shows the format of each entry, and Table 4-12 explains each field.

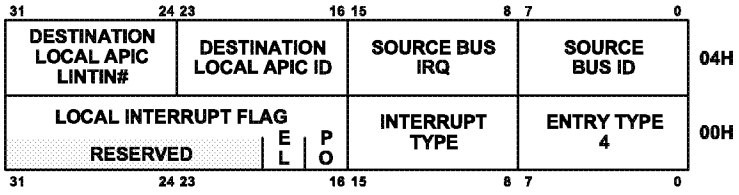


Figure 4-8. Local Interrupt Entry

Table 4-12. Local Interrupt Entry Fields

Field	Offset (in bytes:bits)	Length (in bits)	Description
ENTRY TYPE	0	8	Entry type 4 identifies a local interrupt entry.
INTERRUPT TYPE	1	8	See Table 4-11 for values
PO	2:0	2	Polarity of APIC local input signals: 00 = Conforms to specifications of bus (for example, EISA is active-low for level triggered interrupts) 01 = Active high 10 = Reserved 11 = Active low Must be 00 if the 82489DX is used.
EL	2:2	2	Trigger mode of APIC local input signals: 00 = Conforms to specifications of bus (for example, ISA is edge triggered) 01 = Edge-triggered 10 = Reserved 11 = Level-triggered
SOURCE BUS ID	4	8	Identifies the bus from which the interrupt signal came.
SOURCE BUS IRQ	5	8	Identifies the interrupt signal from the source bus. Values are mapped onto source bus signals, starting from zero. A value of 0, for example, would indicate IRQ0 of an ISA bus. See Section D.3 for PCI bus semantics.
DESTINATION LOCAL APIC ID	6	8	Identifies the local APIC to which the signal is connected. If the ID is 0FFh, the signal is connected to all local APICs.
DESTINATION LOCAL APIC LINTIN#	7	8	Identifies the LINTIN*n* pin to which the signal is connected, where n = 0 or 1.

4.4 Extended MP Configuration Table Entries

A variable number of variable-length entries are located in memory, immediately following entries in the base section of the MP configuration table described in Section 4.3. These entries compose the extended section of the MP configuration table. Each entry in the extended section of the table has three elements:

- ENTRY TYPE
- ENTRY LENGTH
- DATA

ENTRY TYPE and ENTRY LENGTH make up a header that is present in all extended configuration table entries. The ENTRY TYPE and ENTRY LENGTH fields are eight bit unsigned values. ENTRY LENGTH specifies the total length of the given configuration entry. The length of the DATA field is two bytes less than the value specified by ENTRY LENGTH.

This scheme ensures that operating systems can parse all the entries in the extended MP configuration table area, even if an entry type is unrecognized. Operating systems that find an unknown entry type when parsing this section of the table should ignore the content and move on to the next entry, until the offset from the end of the base table reaches the length that is specified in the EXTENDED TABLE LENGTH field of the configuration table header. The ability to skip entries with unrecognized type codes beyond those listed in Table 4-13 is essential since it is anticipated that more types of entries will be added to this list over time.

The total length of the Extended MP configuration table depends upon the configuration of the system. The entries are sorted on ENTRY TYPE in ascending order. Table 4-13 gives the meaning of each value of ENTRY TYPE.

Table 4-13. Extended MP Configuration Table Entry Types

Entry Description	Entry Type Code*	Length (in bytes)	Comments
SYSTEM ADDRESS SPACE MAPPING	128	20	Entry to declare system visible memory or I/O space on a bus.
BUS HIERARCHY DESCRIPTOR	129	8	Entry to describe I/O bus interconnection.
COMPATIBILITY BUS ADDRESS SPACE MODIFIER	130	8	Entry to describe predefined address ranges that modify the memory or I/O space visible on a bus in order to support ISA compatibility.

*All other type codes are reserved.

4.4.1 System Address Space Mapping Entries

System Address Space Mapping entries define the system addresses that are visible on a particular bus. Each bus defined in the Base Table can have any number of System Address Space Mapping entries included in the Extended Table. Thus, individual buses can be configured to support different address ranges, thereby decreasing the amount of bus traffic on a given bus and increasing the overall system performance. Each consecutive address range that is usable by the operating system to access devices on a given bus has a System Address Space Mapping entry. Figure 4-9 shows the format of each entry, and Table 4-14 explains each field. See also Appendix E, for more information.

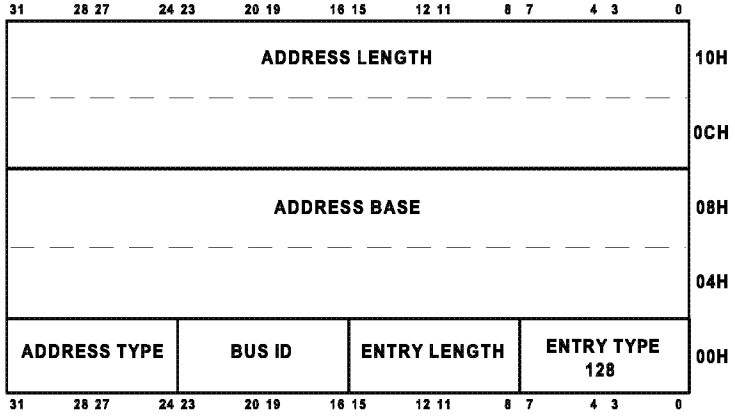


Figure 4-9. System Address Space Entry

Table 4-14. System Address Space Mapping Entry Fields

Field	Offset (in bytes:bits)	Length (in bits)	Description
ENTRY TYPE	0	8	Entry type 128 identifies a System Address Space Mapping Entry.
ENTRY LENGTH	1	8	A value of 20 indicates that an entry of this type is twenty bytes long.
BUS ID	2	8	The BUS ID for the bus where the system address space is mapped. This number corresponds to the BUS ID as defined in the base table bus entry for this bus.
ADDRESS TYPE	3	8	System address type used to access bus addresses must be: 0 = I/O address 1 = Memory address 2 = Prefetch address All other numbers are reserved.
ADDRESS BASE	4	64	Starting address
LENGTH	12	64	Number of addresses which are visible to the bus

If any main memory address is mapped to a software visible bus, such as PCI, it must be explicitly declared using a System Address Space Mapping entry.

In the case of a bus that is directly connected to the main system bus, system address space records and compatibility base address modifiers must be provided as needed to fully describe the complete set of addresses that are mapped to that bus. For example, in Figure 4-10, complete explicit descriptions must be provided for PCI BUS 0 and PCI BUS 1 even if one of the buses is programmed for subtractive decode.

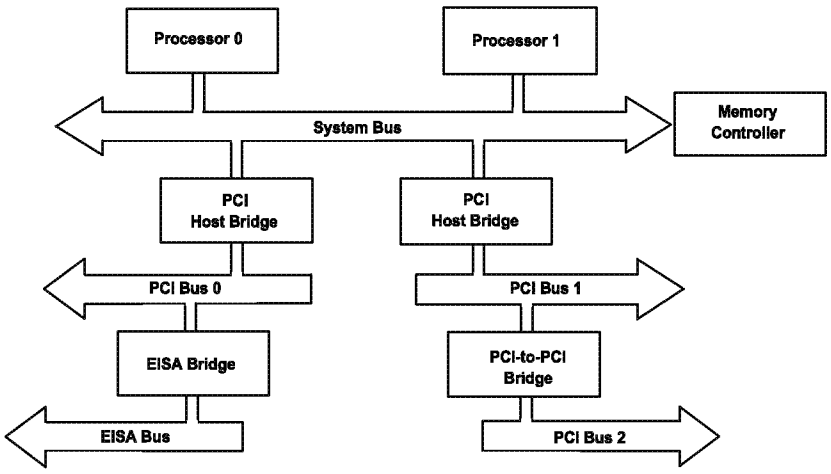


Figure 4-10. Example System with Multiple Bus Types and Bridge Types

Since all device settings must fall within supported System Address Space mapping for a given bus in order to be usable by the operating system, buses that do not support dynamically configurable devices (i.e., ISA, EISA) should support all possible addresses to that bus.

In general, the MP configuration table must provide entries to describe system address space mappings for all I/O buses present in the system. There are two exceptions to this rule:

1. For buses that are connected via PCI-to-PCI-bridge-specification-compliant bridges: in this case, the system address space mappings for such buses may be omitted from the configuration table. In such cases, the operating system is expected to discover the address space mapping by querying the PCI-to-PCI-bridge-specification-compliant bridge directly.
2. For buses that are connected via a parent I/O bus and for which the subtractive decode bit is set (refer to Section 4.4.2 for details).

Typically, this would mean that a minimal description of resources allocated to PCI buses in a system need only include System Address Space Mapping entries for PCI bus zero and any additional peer PCI buses (if present) where these buses are connected by PCI bridges that are specific to the chipset or host bus.

Note

System Address Mappings are unidirectional in nature. They only describe system addresses that propagate to the target bus from any given processor. For DMA support on the target bus, all memory addresses that contain real memory should be accessible directly by either the bus or by a bus-specific DMA controller. For buses with fewer than 32-bit address lines, all real memory at addresses that the bus can generate must be accessible for DMA.

4.4.2 Bus Hierarchy Descriptor Entry

If present, Bus Hierarchy Descriptor entries define how I/O buses are connected relative to each other in a system with more than one I/O bus. Bus Hierarchy Descriptors are used to supplement System Address Mapping entries to describe how addresses propagate to particular buses in systems where address decoding cannot be completely described by System Address Space Mapping entries alone. Entries of this type are required for each bus that is connected to the system hierarchically below another I/O bus. For example, given the system described in Figure 4-10, bus hierarchy entries are required for the EISA bus and the PCI BUS 2 since both have parent buses that are themselves I/O buses.

The Bus Hierarchy entry provides information about where in a hierarchical connection scheme a given bus is connected and the type of address decoding performed for that bus. Figure 4-11 shows the format of each entry, and Table 4-15 explains each field. See also Appendix E, for more information.

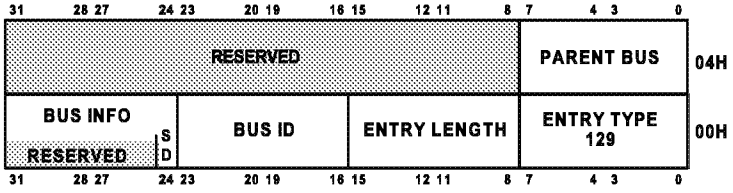


Figure 4-11. Bus Hierarchy Descriptor Entry

Table 4-15 Bus Hierarchy Descriptor Entry Fields

Field	Offset (in bytes:bits)	Length (in bits)	Description
ENTRY TYPE	0	8	Entry type 129 identifies a Bus Hierarchy Descriptor Entry.
ENTRY LENGTH	1	8	A value of 8 indicates that this entry type is eight bytes long.
BUS ID	2	8	The BUS ID identify of this bus. This number corresponds to the BUS ID as defined in the base table bus entry for this bus.
BUS INFORMATION:SD	3:0	1	Subtractive Decode Bus. If set, all addresses visible on the parent bus but not claimed by another device on the parent bus (including bridges to other buses) are useable on this bus.
PARENT BUS	4	8	Parent Bus. This number corresponds to the BUS ID as defined in the base table bus entry for the parent bus of this bus

For buses where the BUS INFORMATION:SD bit is set, System Address Mappings may not be needed. Since the bus is defined as being subtractive decode, the range of addresses that appear on the bus can be derived from address decoding information for parent and peer buses.

4.4.3 Compatibility Bus Address Space Modifier Entry

The Compatibility Bus Address Space Modifier defines a set of predefined address ranges that should either be added or removed from the supported address map ranges for a given bus. This entry type is used in combination with System Address Space Mapping entries to complete the description of memory and I/O ranges that are visible on a bus that incorporates support for ISA device compatibility.

For example, a host bus bridge for a PCI bus that provides ISA compatibility may decode a predefined range of addresses used for ISA device support in addition to the addresses used for PCI devices on that bus. A Compatibility Bus Address Space Modifier can be used in this case to add these predefined address ranges to the list specified by System Address Space Mapping entries for that PCI bus. As a corollary, in a system where two peer PCI buses are included, one of which provides ISA compatibility, a Compatibility Bus Address Space Modifier can be used to subtract these predefined ranges from the address space assigned to the PCI bus that does not support ISA devices to avoid any potential conflict.

The same effect can be achieved by using System Address Space Mapping entries to completely describe the address ranges supported on a bus, including those ranges that might otherwise be described by a Compatibility Bus Address Space Modifier entry. However, given the number of discrete address ranges that are used for ISA device compatibility, using an approach based solely on System Address Space Mapping entries may result in a significantly larger number of configuration table entries and a corresponding increase in table size.

Figure 4-12 shows the format of each entry, and Table 4-16 explains each field.

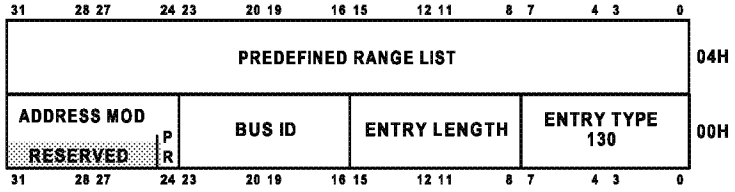


Figure 4-12. Compatibility Bus Address Space Modifier Entry

Table 4-16. Compatibility Bus Address Space Modifier Entry Fields

Field	Offset (in bytes:bits)	Length (in bits)	Description
ENTRY TYPE	0	8	Entry type 130 identifies a Compatibility Bus Address Space Modifier Entry.
ENTRY LENGTH	1	8	A value of 8 indicates that an entry of this type is eight bytes long.
BUS ID	2	8	Bus for address space mappings are to be modified. This number corresponds to the BUS ID as defined in the base table entry for this bus.
ADDRESS MODIFIER:PR	3:0	1	If this bit is set to one, the address ranges specified by PREDEFINED RANGE LIST are to be subtracted from the address space associated with the bus. If this bit is set to zero, the specified address ranges are to be added to the address space associated with the bus.
PREDEFINED RANGE LIST	4	32	A number that indicates the list of predefined address space ranges that this record will modify for the bus.

PREDEFINED RANGE LIST may take one of the values from Table 4-17. The value of PREDEFINED RANGE LIST indicates the set of address ranges that are to be either added to or subtracted from the address range associated with the BUS ID.

Table 4-17. Predefined Range Lists

List	Value	Address Ranges
ISA Compatible I/O Range	0	X100-X3FF X500-X7FF X900-XBFF XD00-XFFF
VGA Compatible I/O Range	1	X3B0 - X3BB X3C0 - X3DF X7B0 - X7BB X7C0 - X7DF XBB0 - XBBB XBC0 - XBDF XFB0 - XFBB XFC0 - XFDF

All addresses in Table 4-17 are in hexadecimal notation. In each case the X represents any hexadecimal digit, 0-F. As a result, the ISA Compatible I/O Range describes 64 distinct ranges.

5
Default Configurations

The MP specification defines several default MP system configurations. The purpose of these defaults is to simplify BIOS design. If a system conforms to one of the default configurations, the BIOS will not need to provide the MP configuration table. The operating system will have the default MP configuration table predefined internally.

Default system configuration types are defined by MP feature information byte 1, which is part of the MP floating pointer structure. The physical address pointer field of the MP floating pointer structure must be zero if one of the default configurations is selected. (Refer to Chapter 4 for a detailed description of the MP floating pointer structure.)

To use a default configuration, a system must meet the following basic criteria:

1. The system supports two processors.
2. Both processors must execute the common Intel architecture instruction set.
3. The local APICs are located at base memory address 0FEE0_0000h.
4. The local APIC IDs are assigned consecutively by hardware starting from zero.
5. An I/O APIC is present at base memory address 0FEC0_0000h.
6. Either PIC Mode or Virtual Wire Mode is implemented as the power-on default interrupt mode.

The default system configurations include configurations that use the discrete APIC, such as the Intel 82489DX or its equivalent, and configurations that use the integrated APIC, such as the Pentium processors (735\90, 815\100).

Each default configuration has a unique code. Table 5-1 specifies the configuration associated with each code.

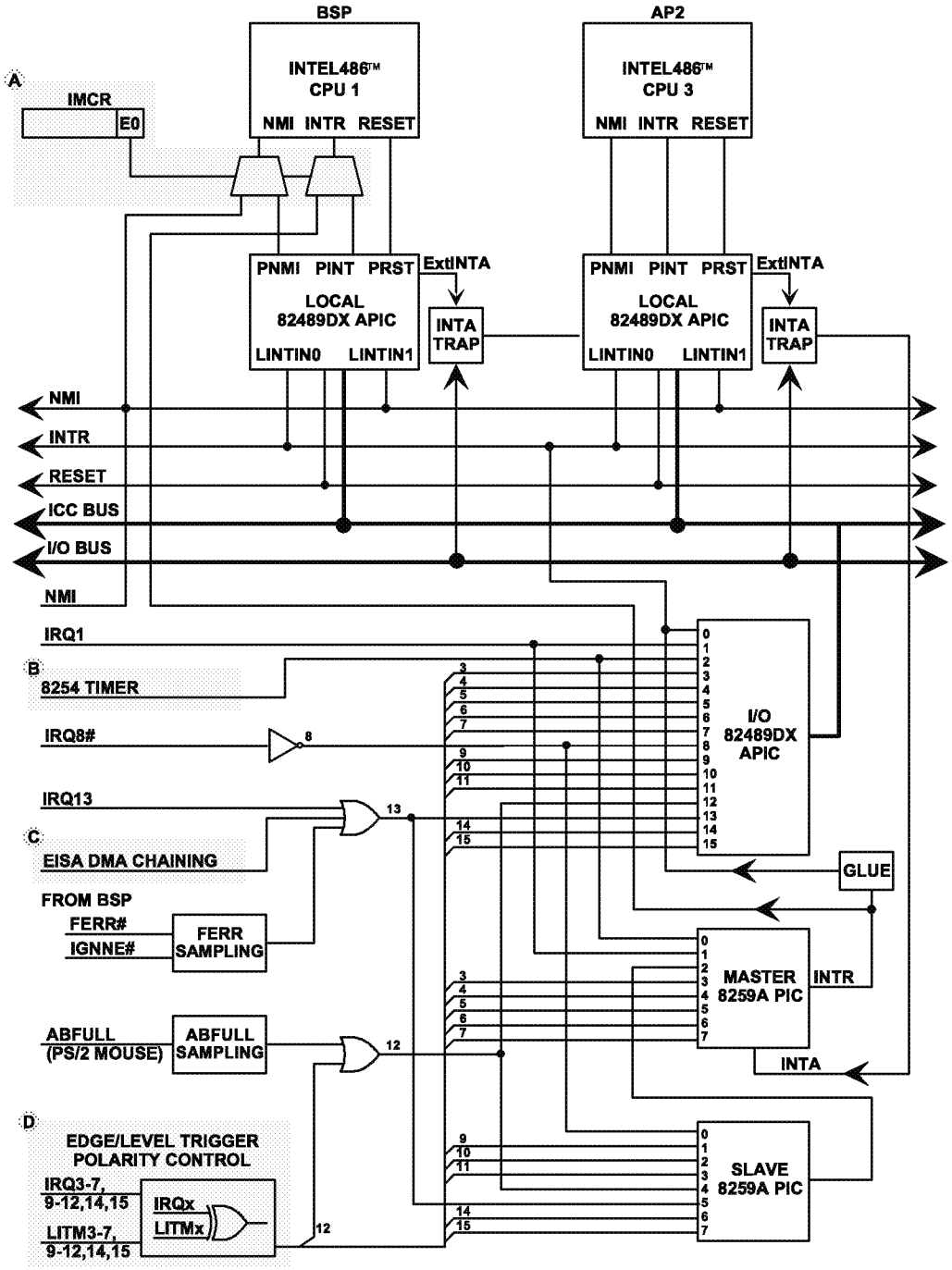
Table 5-1. Default Configurations

Default Config Code	Number of CPUs	Bus Type	APIC Type	Variant	Schematic
1	2	ISA	82489DX		As in Figure 5-1, but without EISA logic.
2	2	EISA	82489DX	Neither timer IRQ0 nor DMA chaining	As in Figure 5-1, but without IRQ0 and IRQ13 connection to the I/O APIC.
3	2	EISA	82489DX		As in Figure 5-1.
4	2	MCA	82489DX		As in Figure 5-1, but without EISA bus logic, with inverters before I/O APIC inputs 1-15.
5	2	ISA + PCI	Integrated		As in Figure 5-2, but without EISA logic.
6	2	EISA + PCI	Integrated		As in Figure 5-2.
7	2	MCA + PCI	Integrated		As in Figure 5-2, but without EISA bus logic, with inverters before I/O APIC inputs 1-15.
8-255	Reserved for MP future use.				

The default system configurations are designed to support dual-processor systems with fixed configurations. Systems with dynamically configurable components, for example, a uniprocessor system with an upgrade socket for the second processor, must always generate the MP configuration table. Failure to do so may cause the operating system to install the wrong modules due to erroneous configuration information.

5.1 Discrete APIC Configurations

Figure 5-1 shows the default configuration for systems that use the discrete 82489 APIC. The Intel486 processor is shown as an example; however, this configuration can also employ Pentium processors. In Pentium processor systems, PRST is connected to INIT instead of to RESET.



SHADED AREAS:

A: OPTIONAL IF VIRTUAL WIRE MODE IS IMPLEMENTED

B,C: MAY NOT BE EXTERNALIZED WITH SOME EISA CHIPSETS

C,D: EISA BUS SPECIFIC

Figure 5-1. Default Configuration for Discrete APIC

The INTA TRAP and GLUE in the figure are the additional hardware interface logic needed for the 82489DX APIC. INTA TRAP conditions all interrupt acknowledge cycles with ExtINTA to steer the vector either from the 8259A PIC or the APIC. INTA TRAP is also responsible for preventing the interrupt acknowledge cycle from reaching the 8259A PIC, in case ExtINTA is negated when PINT is activated. During an interrupt acknowledge cycle with ExtINTA active, the APIC does not return RDY#. Therefore, the ready generation logic should also take into consideration the status of ExtINTA to steer the ready signal either from the APIC or from external bus logic, depending upon the source of the interrupt vector.

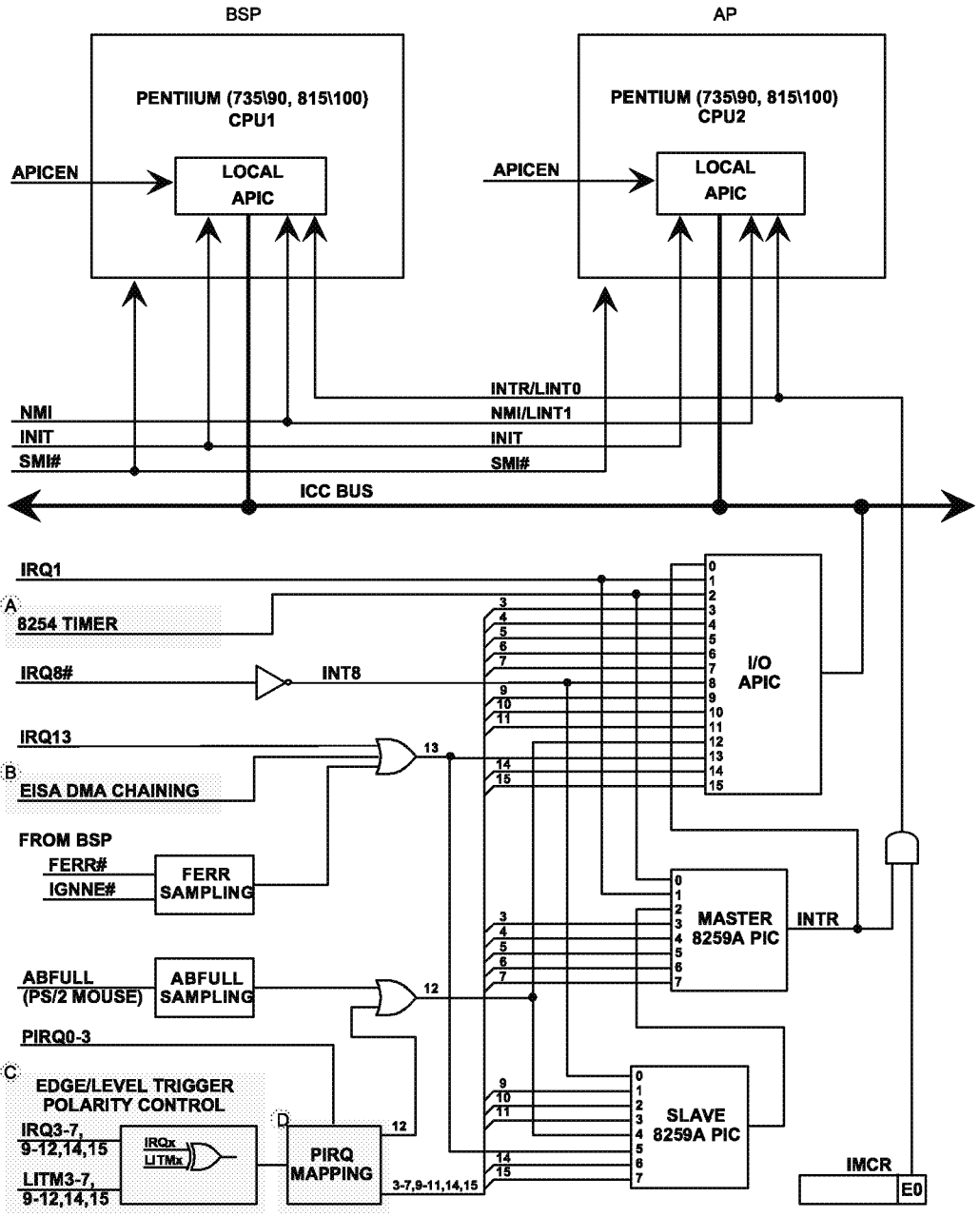
The GLUE logic converts the INTR level-triggered interrupt output signal to a signal that is acceptable to edge-triggered input pins, such as INTIN0 or LINTIN0.

If the AP used in these configurations does not automatically HALT after RESET or INIT, the AP must be prevented from executing the BIOS by external hardware or by the BIOS itself.

5.2 Integrated APIC Configurations

Figure 5-2 shows the default configuration for systems that use processors with the integrated APIC, such as Pentium processors (735\90 or 815\100). The local APIC is configured as part of the processor unit. The APICEN input is used to enable or disable the internal local APIC. If the internal local APIC is used, the BIOS must initialize the APIC to Virtual Wire Mode during system initialization.

Both Pentium processors used in the dual-processor (DP) system design are identical. The AP functions exactly the same as the BSP processor, except that the AP will go to HALT after the assertion of the RESET or INIT signals. It will remain halted until the MP operating system sends a STARTUP IPI to bring it on line.



SHADED AREAS:
A,B: MAY NOT BE EXTERNALIZED WITH SOME EISA CHIPSETS
B,C: EISA BUS SPECIFIC
D: PCI BUS SPECIFIC

Figure 5-2. Default Configuration for Integrated APIC

Two local interrupt input pins, LINT0 and LINT1, are shared with the INTR and NMI pins, respectively. The LINT0, LINT1, SMI# and INIT signals are switched by APICEN, and they

should be cross-connected between the BSP and AP processors. Although the INIT pin is cross-connected between BSP and AP, a targeted INIT IPI initializes only the targeted processor, because the INIT IPI does not cause the INIT pin to change state.

The interconnection of I/O APIC interrupt lines is the same as for the 82489DX APIC configuration. However, for PCI system implementations based on the Intel PCI chipset, the PCI PIRQx lines are mapped to the ISA IRQx via a mapping register. This type of implementation makes PCI interrupt lines appear as ISA interrupt lines, which are transparent to the operating system. All PCI systems defined in the default configurations are of this type. No I/O interrupt assignment entries are declared for PCI interrupts, as described in Section 4.3.4.

5.3 Assignment of I/O Interrupts to the APIC I/O Unit

The typical APIC I/O unit has 16 general-purpose interrupt inputs. Table 5-2 shows how the interrupt request line (IRQ) assignments are connected to the I/O APIC in each of the default configurations.

Table 5-2. Default Configuration Interrupt Assignments

First I/O APIC INTINx	Config 1	Config 2	Config 3	Config 4	Config 5	Config 6	Config 7	Comments
INTIN0	8259A INTR	8259A INTR	8259A INTR	8259A INTR	8259A INTR	8259A INTR	N/C	INTR output from master 8259A or equivalent
INTIN1	IRQ1	IRQ1	IRQ1	IRQ1	IRQ1	IRQ1	IRQ1	Keyboard controller buffer full
INTIN2	IRQ0	N/C	IRQ0	IRQ0	IRQ0	IRQ0	IRQ0	8254 Timer
INTIN3	IRQ3	IRQ3	IRQ3	IRQ3	IRQ3	IRQ3	IRQ3	
INTIN4	IRQ4	IRQ4	IRQ4	IRQ4	IRQ4	IRQ4	IRQ4	
INTIN5	IRQ5	IRQ5	IRQ5	IRQ5	IRQ5	IRQ5	IRQ5	
INTIN6	IRQ6	IRQ6	IRQ6	IRQ6	IRQ6	IRQ6	IRQ6	
INTIN7	IRQ7	IRQ7	IRQ7	IRQ7	IRQ7	IRQ7	IRQ7	
INTIN8	IRQ8	IRQ8	IRQ8	IRQ8	IRQ8	IRQ8	IRQ8	Real time clock
INTIN9	IRQ9	IRQ9	IRQ9	IRQ9	IRQ9	IRQ9	IRQ9	
INTIN10	IRQ10	IRQ10	IRQ10	IRQ10	IRQ10	IRQ10	IRQ10	
INTIN11	IRQ11	IRQ11	IRQ11	IRQ11	IRQ11	IRQ11	IRQ11	
INTIN12	IRQ12	IRQ12	IRQ12	IRQ12	IRQ12	IRQ12	IRQ12	
INTIN13	IRQ13	N/C	IRQ13	IRQ13	IRQ13	IRQ13	IRQ13	Floating point exception and DMA chaining
INTIN14	IRQ14	IRQ14	IRQ14	IRQ14	IRQ14	IRQ14	IRQ14	
INTIN15	IRQ15	IRQ15	IRQ15	IRQ15	IRQ15	IRQ15	IRQ15	

NOTE:
N/C designates not connected.

Certain EISA chipsets do not bring out the IRQ0, 8254 timer interrupt, and IRQ13 EISA DMA chaining interrupt signals. If these signals are not directly available, INTIN2 and INTIN13 should be disabled. Refer to Section 5.3.1 for more details.

5.3.1 EISA and IRQ13

IRQ13 is a shared interrupt as defined in the EISA bus specification. Because a compliant system supports only the on-chip floating point unit, IRQ13 carries only the EISA chaining interrupt.

If IRQ13 is not connected to the I/O APIC, the EISA chaining interrupt may be handled as a mixed-mode operation. Mixed mode means that the APIC and 8259A-equivalent PIC are connected in a cascading manner via INTIN0, and INTIN0 is programmed for ExtINT and edge-triggered mode. If all other interrupts are masked off in the PIC, INTIN0 only receives the DMA chaining interrupt.

An MP operating system should disable the I/O APIC INTIN13 and configure the I/O APIC to mixed mode if the EISA DMA chaining signal is not available at the I/O APIC.

5.3.2 Level-triggered Interrupt Support

Several AT-compatible buses, such as EISA and MCA, support active-low, level-triggered interrupts. If these types of buses are to be incorporated in a compliant system, external inverters must be implemented to ensure that signals presented to the 82489DX APIC are active-high and level-triggered. See Section 4.3.4 on I/O Interrupt Assignment Flags.

For EISA implementations, the external interrupt polarity control inverters must be controlled by the EISA edge/level-triggered polarity control registers (4D0h-4D1h). MCA does not have this register. To convert an active-high trigger to an active-low trigger, an inverter for each interrupt line must be implemented.

5.4 Assignment of System Interrupts to the APIC Local Unit

The APIC local unit has two general-purpose interrupt inputs that are reserved for system interrupts. Table 5-3 shows how the interrupt request line (IRQ) assignments are connected to the local APIC in each of the default configurations.

Table 5-3 Assignment of System Interrupts to APIC Local Unit

All Local APICs LINTINx	Config 1	Config 2	Config 3	Config 4	Config 5	Config 6	Config 7	Comments
LINTIN0	8259A INTR	8259A INTR	8259A INTR	8259A INTR	8259A INTR	8259A INTR	8259A INTR	INTR output from master 8259A or equivalent
LINTIN1	NMI	NMI	NMI	NMI	NMI	NMI	NMI	Nonmaskable interrupt

The 8259A INTR output signal is connected to the LINTIN0 of all local APICs, which makes INTR dynamically routable via software. NMI is connected to the LINTIN1 of all local APICs, which makes NMI dynamically routable via software.

In PIC-Mode configurations, the NMI signal is delivered to the local interrupt input 1 (LINTIN1) of all local APICs and the input of a 2-to-1 MUX. When the system is operated in PIC Mode, the NMI is sent to the BSP directly via the MUX. The BIOS and the operating system must leave the LINTIN1 of all local APICs disabled to ensure that the BSP is the only processor that receives the NMI.

In PIC-Mode configurations, the 8259 INTR signal also follows the same convention, connecting to the local interrupt line 0 (LINTIN0) of all local APICs and the input of the second 2-to-1 MUX. When the system is operated in PIC Mode, the 8259 INTR is sent to the BSP directly via the MUX. The BIOS and the operating system must leave the LINTIN0 of all local APICs disabled to ensure that the BSP is the only processor that receives the 8259 INTR signal.

When the system is operated in Symmetric I/O Mode, the operating system may enable the LINTIN0 and LINTIN1 of any or all local APICs as necessary.

A
System BIOS Programming Guidelines

Depending on the MP components in a multiprocessor system, the system BIOS may have the following additional responsibilities:

1. Put the APs to sleep, so that they do not all try to execute the same BIOS code as the BSP. This is necessary, because BIOS code is not typically multithreaded for multiprocessing.
2. Initialize the APICs and other MP components (if any).
3. Build the MP configuration table to communicate information to the operating system about the APICs and APs.

Note that the above activities can be implemented by the hardware. The BIOS is not required to perform these activities if the hardware makes them unnecessary. For example, the system BIOS for one of the default configurations defined in Chapter 5 needs to ensure only that the MP feature information bytes identify the configuration. In all other respects, the BIOS can be the same as a standard PC/AT BIOS.

Support for the shutdown status byte (0Fh) of the PC/AT CMOS RAM is required. The startup of APs by the operating system depends on the jump to warm reset vector (40:67h) capability, as defined by the shutdown status byte. Appendix B explains this in more detail.

A.1 BIOS Post Initialization

Once system power is applied or the reset button is pressed (if the system is so equipped), a hardware circuit generates a system RESET sequence to put all the system hardware into an initial state. All active processors start to execute instructions and enter the POST (power-on self test) procedure of the BIOS, which is responsible for initializing all components in a system to a known state and for constructing various system tables in the BIOS data area (400h-4FFh) for the operating system to use.

For compliant systems that match one of the default configurations listed in Chapter 5, the work performed by the BIOS POST is minimal. The MP feature information bytes must identify the default configuration type and determine whether PIC Mode or Virtual Wire Mode is implemented.

During the system INIT or soft reset cycle, both local and I/O APICs must be reinitialized by the INIT signal and by the BIOS. This is required because the operating system will always assume that all components in the system are initialized to a known state. For the APIC, this means that all APIC registers are cleared and the local APIC ID register is initialized by the BIOS or the hardware.

Upon warm reset, the BIOS must initialize all APICs to the power on state if the warm reset signal does not physically reset the APICs.

A.2 Controlling the Application Processors

Provision must be made to prevent all processors from executing the BIOS after a power-on RESET. System developers may choose to do this by the hardware alone or by cooperation between hardware and the BIOS. In the latter case, the BIOS may be used for selecting the BSP and placing all APs to sleep after POST. The BIOS may use the APIC ID as a means by which to identify each processor and select the proper code sequence to execute. Only the selected BSP continues to load the operating system after the POST routine.

A.3 Programming the APIC for Virtual Wire Mode

The APICs do not require BIOS programming if the default interrupt mode at start-up is PIC Mode. Special programming is needed only if the startup interrupt mode is Virtual Wire Mode.

Because Virtual Wire Mode must run all existing uniprocessor software, the system BIOS must initialize and enable the BSP's APIC first. The local unit must be programmed to function as a "virtual wire," which delivers the CPU interrupt from the 8259A-equivalent PIC to the BSP via its local APIC.

The External Interrupt (ExtINT) delivery mode must be used so that the APICs and 8259A-equivalent PICs can function together in the same system. For interrupts that are programmed for ExtINT delivery mode, there is no need to issue an EOI to the APIC; only the 8259A PIC requires an EOI as usual. Also, because the 8259A delivers the vector to the processor for ExtINT delivery mode, the interrupt vector in the APIC's redirection table is ignored.

To program the APIC to Virtual Wire Mode, the system BIOS must program the APIC to enable the LINT0 of the BSP's local APIC for edge-triggered ExtINT delivery mode, and LINT1 for level-triggered NMI delivery mode. There is no need to program the I/O APIC if it is not used in Virtual Wire Mode.

Example A-1 is an example of programming the LINTIN0 and LINTIN1 to support Virtual Wire Mode.

```
;-----------------------------------------------------------------------;
; InitLocalAPIC( )                                                      ;
;-----------------------------------------------------------------------;
;                                                                       ;
;            Initialize the local APIC to virtual wire mode.            ;
;                                                                       ;
;-----------------------------------------------------------------------;

SVR             equ     0FEE000F0H
LVT1            equ     0FEE00350H
LVT2            equ     0FEE00360H

APIC_ENABLED    equ     000000100H

        public  InitLocalAPIC
InitLocalAPIC   proc    near

        push    ds                      ; save regs used for APIC init
        push    es
        push    esi

        mov     al,080h                 ; ensure NMI disabled
        out     070h,al

        in      al,021h                 ; read primary imr
        push    ax                      ; save settings
        mov     al,0ffh                 ; mask all off
        out     021h,al
        in      al,0a1h                 ; read secondary imr
        push    ax                      ; save settings
        mov     al,0ffh                 ; mask all off
        out     0a1h,al

        extrn   pmode_on : near
        call    pmode_on                ; switch into real big mode

;
; The APIC spurious interrupt must point to a vector whose lower
; nibble is 0F, that is 0xF, where x is 0 - F. Here we use Int 00FH,
; which handles spurious interrupts and supplies the necessary IRET.
; This vector is assumed to have already been initialized in memory.
;
; Enable the APIC via SVR and set the spurious interrupt to use Int 00F
;
        mov     esi,SVR
        mov     eax,[esi]               ; read SVR
        and     eax,0FFFFFF0FH          ; clear spurious vector (use vector
                                          00FH)
        or      eax,APIC_ENABLED        ; bit 8 = 1
        mov     [esi],eax               ; write SVR

;
; Program LVT1 as ExtInt, which delivers the signal to the INTR signal of all
; processors' cores listed in the destination as an interrupt that originated
; in an externally-connected interrupt controller.
;
```

Example A-1. Programming Local APIC for Virtual Wire Mode

```
        mov     esi,LVT1
        mov     eax,[esi]               ; read LVT1
        and     eax,0FFFE00FFH          ; not masked, edge, active high
        or      eax,000005700H          ; ExtInt
        mov     [esi],eax               ; write LVT1

;
; Program LVT2 as NMI, which delivers the signal on the NMI signal of all
; processors' cores listed in the destination.
;
        mov     esi,LVT2
        mov     eax,[esi]               ; read LVT2
        and     eax,0FFFE00FFH          ; not masked, edge, active high
        or      eax,000005400H          ; NMI
        mov     [esi],eax               ; write LVT2

        extrn   pmode_off : near
        call    pmode_off               ; switch back to real mode

        pop     ax                      ; restore imr settings
        out     0a1h,al                 ; restore secondary imr
        pop     ax
        out     021h,al                 ; restore primary imr
                                        ; this routine leaves NMI disabled
        pop     esi                     ; restore regs used in APIC init
        pop     es                      ; (unless also saved for CPUID)
        pop     ds
        ret

InitLocalAPIC   endp
```

Example A-1. Programming Local APIC for Virtual Wire Mode (continued)

A.4 Constructing the MP Configuration Table

For a compliant system, one of the main functions of the system BIOS is to construct the MP floating pointer structure and the MP configuration table. Because the MP configuration table is optional, the BIOS must set the MP feature information bytes in the MP floating pointer structure to indicate whether an MP configuration table is present.

If the MP configuration table is required, the BIOS constructs it in conjunction with the BSP and APs. The BIOS is responsible for synchronizing the activities of the APs during the construction of the table. The BIOS may need some synchronization during processor initialization so that each processor may be brought up in the proper order. The mechanism for synchronization is not specified; however, the procedure described in the following paragraphs of this section uses AP status flags as an example of a synchronization mechanism. This procedure also initializes the APs serially. System developers may employ other mechanisms and may initialize all processors in parallel to minimize the system start-up time.

The BIOS maintains an initialized AP status flag for each AP. Each AP will begin executing the same BIOS code as the BSP, but will eventually be put in a HALT state or held in a loop until the BSP enables its AP status flag.

The BSP is responsible for positioning the MP configuration table. The table can be located within any unreported, hidden system memory space or within the BIOS ROM region. The BIOS can select any unused space in those regions. For example, some PC/AT systems implement the Extended BIOS Data Segment, a 1-Kbyte block usually positioned at the top of the PC's 640K base memory. The configuration table may be placed in this portion of the memory map if enough free space is available.

The BIOS initializes the floating table pointer with the configuration table's memory address. Then the BSP constructs the MP configuration table based on the total number of processors, buses, I/O APICs, and interrupt sources in the system. It sets all AP processor entry bits to zero.

Next, the BSP enables each AP in turn by setting its AP status flag. Each AP follows these steps:

1. It executes a CPU identification procedure, reads its local APIC ID from the Local Unit ID Register, and uses this information to complete its entry in the MP configuration table.
2. Just prior to exiting the BIOS, the AP clears its status flag to signal the BSP to enable the next AP initialization process.
3. The AP, which is in Real Mode with interrupts disabled, executes a HLT instruction, and enters the HALT state.

After the APs complete the initialization process, the BSP continues filling in the rest of the MP configuration table entries. For I/O interrupt assignment entries, each line should have its trigger mode and polarity configured according to the device installed.

Once the MP table configuration process is complete, the BSP must calculate the checksum for the MP configuration table. The MP configuration table should be fully populated at this point. A disabled processor entry indicates that an AP failed in the initialization self-test. System developers may choose either to suspend system start up or to continue with the partially functional system.

The BSP is the only processor that is responsible for loading the operating system. It is up to the MP operating system to decide when to bring the APs on-line.

B
Operating System Programming Guidelines

The goal of the MP specification is to transfer enough information about the hardware environment to the operating system that a single, shrink-wrapped, operating-system binary can boot-up and fully utilize a wide variety of multiprocessor systems. The following sections explain how the operating system can take advantage of this specification to handle these operations:

1. Operating-system boot-up.
2. Self configuration.
3. Interrupt mode initialization.
4. Application processor startup.
5. Application processor shutdown.
6. Dynamic interrupt masking.
7. Support for unequal processors.

B.1 Operating System Boot-up

While all processors in an MP-compliant system are functionally identical, one of the processors will be designated as the boot processor (BSP) at system initialization by the system hardware or by the system hardware in conjunction with the BIOS. The rest of the processors are designated as the application processors (APs). The BSP is responsible for booting the operating system. Once the MP operating system is up and running, the BSP functions as an AP.

Usually a processor is designated as the BSP because it is capable of controlling all system hardware, including AP startup and shutdown. The operating system must determine and remember the APIC ID of the designated BSP, so it can keep the BSP operating as the last running processor during system shutdown. The BSP is not necessarily the first processor, especially in fault-tolerant MP systems in which any available processor can be designated as the BSP.

At the time that the first instruction of the operating system is executed, the APs are in the following state:

- The APs have been restrained (either by the BIOS or by the hardware) from executing operating system code.
- The APs are in a halted condition with interrupts disabled. This means that the AP's local APICs are passively monitoring the APIC bus and will react only to INIT or STARTUP interprocessor interrupts (IPIs).

The operating system's first task is to determine whether the system conforms to the MP specification. This is done by searching for the MP floating pointer structure. If a valid floating pointer structure is detected, it indicates that the system is MP-compliant, and the operating system should continue to look for the MP configuration table. If the system is not MP-compliant, the operating system may attempt other means of MP system detection, if it is capable of doing so, or treat the system as a uniprocessor system.

B.2 Operating System Booting and Self-configuration

An MP configuration table is required by the MP specification, with the exception of the default system configurations defined in Chapter 5. The table should be treated as read-only by the operating system. If the MP configuration table exists, the BSP should access the processor entries in the table to configure the operating system.

The BSP should later configure the operating system based on the bus, I/O APIC, IRQs, and system interrupt assignment entries of the configuration table. Note that certain types of buses are mutually exclusive, such as EISA with MCA, or ISA with MCA. The operating system may report such errors in the configuration table, if they occur.

If the MP configuration table does not exist, the BSP configures the operating system for the default system configuration indicated by the default configuration bits of the MP feature information bytes. In this case, only two processors and one I/O APIC exist in the system; both processors have the same type and features.

The operating system contains a set of predefined internal configuration tables that represent the default configurations described in Chapter 5. For MP-compliant systems that use one of the default configurations, the operating system derives the required configuration information from the corresponding predefined table.

To wake up the AP, the BSP should use the universal algorithm defined in Section B.4.

B.3 Interrupt Mode Initialization and Handling

At the time the operating system boots, the interrupt structure is configured for DOS compatibility. The system may be running either in PIC Mode or in Virtual Wire Mode. If it is in PIC Mode, the NMI and INTR will bypass the BSP's local APIC when the Interrupt Mode Configuration Register (IMCR) has a value of zero. The operating system should not try to read the IMCR because it may not exist.

The operating system should switch over to Symmetric I/O Mode to start multiprocessor operation. If the IMCRP bit of the MP feature information bytes is set, the operating system must set the IMCR to APIC mode. The operating system should not write to the IMCR unless the IMCRP bit is set.

Then the operating system should enable its own local APIC, thereby allowing IPI communications with other APIC-based processors. At this time, the APs' local APICs have interrupts disabled. Interrupts must remain disabled at the APs' local APICs while the BSP is enabling the I/O APIC and bringing the system to the normal operating state. Otherwise, an I/O interrupt may be delivered to the uninitialized AP, resulting in the loss of the interrupt.

It is the responsibility of the operating system to assign unique IDs to I/O APIC units.

B.4 Application Processor Startup

An AP may be started either by the BSP or by another active AP. The operating system causes application processors to start executing their initial tasks in the operating system code by using the following universal algorithm. The algorithm detailed below consists of a sequence of interprocessor interrupts and short programmatic delays to allow the APs to respond to the wakeup commands. The algorithm shown here in pseudo-code assumes that the BSP is starting an AP for documentation convenience. The BSP must initialize BIOS shutdown code to 0AH and the warm reset vector (DWORD based at 40:67) to point to the AP startup code prior to executing the following sequence:

```
BSP sends AP an INIT IPI
BSP DELAYs (10mSec)
If (APIC_VERSION is not an 82489DX) {
    BSP sends AP a STARTUP IPI
    BSP DELAYs (200uSEC)
    BSP sends AP a STARTUP IPI
    BSP DELAYs (200uSEC)
}
BSP verifies synchronization with executing AP
```

Example B-1. Universal Start-up Algorithm

If the MP configuration table exists, it provides the IDs of the application processor local APICs. These IDs should be used as the destination addresses in targeted IPIs.

If the MP configuration table does not exist on an MP-compliant system, the system must be of default configuration type. The MP specification requires local APIC IDs to be numbered sequentially, starting at zero for all default configurations. As a result, the BSP can determine the AP's local APIC ID in default, two-processor configurations by reading its own local APIC ID. Since there are only two possible local APIC IDs in this case, zero and one, when the APIC ID of the BSP is one, the APIC ID of the AP is zero, and vice versa. This is important, because a BSP cannot start up an AP unless it already knows the local APIC ID.

Both INIT IPI and STARTUP IPI are open-ended commands. The operating system is responsible for determining whether its local APIC unit successfully dispatches one of these commands. The operating system must do so for an INIT and STARTUP IPI because the APIC either does not automatically retry or guarantee delivery for one of these special messages.

A local APIC unit indicates successful dispatch of an IPI by resetting the Delivery Status bit in the Interrupt Command Register (ICR). The operating system polls the delivery status bit after sending an INIT or STARTUP IPI until the command has been dispatched.

A period of 20 microseconds should be sufficient for IPI dispatch to complete under normal operating conditions. If the IPI is not successfully dispatched, the operating system can abort the command. Alternatively, the operating system can retry the IPI by writing the lower 32-bit double word of the ICR. This "time-out" mechanism can be implemented through an external interrupt, if interrupts are enabled on the processor, or through execution of an instruction or time-stamp counter spin loop.

Optionally, the operating system can make use of the information provided by the integrated APIC error register. Integrated local APIC units on Intel Architecture processors provide an APIC error register that indicates the reason for non-delivery of an APIC message. Reasons for non-delivery include SEND_ACCEPT and RECEIVE_ACCEPT errors that are generated when no processor responds to a message on the APIC bus. An example of this type of failure includes sending a STARTUP IPI to a destination APIC ID for a non-existent processor. Other errors are usually indicative of hardware problems and include SEND and RECEIVE CHECKSUM errors.

The operating system is responsible for determining whether the IPI was received by the targeted AP and executed successfully. For example, the operating system can define a status flag for each processor. After being awakened, a processor sets its status flag, indicating to the operating system that it is present and running. If the status flag does not change value after a certain time, the operating system should treat the processor as not present or not functional.

The following two sections describe specifics relating to the use of INIT and STARTUP IPIs that can be used to guide the implementation of the universal algorithm presented here.

B.4.1 USING INIT IPI

INIT IPIs can be used with systems based on the 82489DX APIC, or on systems that are based on multiple Pentium (735/90, 815/100) processors. INIT IPI is an Interprocessor Interrupt with trigger mode set to level and delivery mode set to "101" (bits 8 to 10 of the ICR). INIT IPIs should always be programmed as level triggered; the operating system must perform two writes to the ICR to assert and then deassert this delivery mode.

An INIT IPI is an IPI that has its delivery mode set to RESET. Upon receiving an INIT IPI, a local APIC causes an INIT at its processor. The processor resets its state, except that caches, floating point unit, and write buffers are not cleared. Then the processor starts executing from a fixed location, which is the reset vector location. To cause the processor to jump to a different location, the INIT IPI must be used as part of a warm-reset.

The warm reset is a feature of every standard PC/AT BIOS. It allows the INIT signal to be asserted without actually causing the processor to run through its entire BIOS initialization procedure (POST). This feature is used, for example, to return an 80286 processor to Real Mode. The key components of warm reset are the following:

- Shutdown code. One of the first actions of the BIOS POST procedure is to read the shutdown code from location 0Fh of the CMOS RAM. This code can have any of several values that indicate the reason that an INIT was performed. A value of 0Ah indicates a warm reset.
- Warm-reset vector. When POST finds a shutdown code of 0Ah, it executes an indirect jump via the warm-reset vector, which is a doubleword pointer in system RAM location 40:67h.

By putting an appropriate pointer in the warm-reset vector, setting the shutdown code to 0Ah, then causing an INIT, the BIOS (or the operating system) can cause the current processor to jump immediately to any location. Because all processors in an MP system share the same system memory, and because the INIT IPI gives one processor the power to cause an INIT at another, the operating system can cause any processor to jump immediately to any location.

B.4.2 USING STARTUP IPI

STARTUP IPIs are used with systems based on Intel processors with local APIC versions of 1.x or higher. These local APICs recognize the STARTUP IPI, which is an APIC Interprocessor Interrupt with trigger mode set to edge and delivery mode set to "110" (bits 8 through 10 of the ICR).

The STARTUP IPI causes the target processor to start executing in Real Mode from address 000VV000h, where VV is an 8-bit vector that is part of the IPI message. Startup vectors are limited to a 4-kilobyte page boundary in the first megabyte of the address space. Vectors A0-BF are reserved; do not use vectors in this range. STARTUP IPIs are not maskable, do not cause any change of state in the target processor (except for the change to the instruction pointer), and can be issued only one time after RESET or after an INIT IPI reception or pin assertion. A STARTUP IPI neither requires the targeted APIC to be enabled nor the interrupt table to be programmed. If the target processor is in the halted state immediately after RESET or INIT, a STARTUP IPI causes it to leave that state and start executing. The effect is to set CS:IP to VV00:0000h.

For an operating system to use a STARTUP IPI to wake up an AP, the address of the AP initialization routine (or of a branch to that routine) must be in the form of 000VV000h. Sending a STARTUP IPI with VV as its vector causes the AP to jump immediately to and begin executing the operating system's AP initialization routine.

The operating system should not issue a STARTUP IPI to an 82489DX since the STARTUP IPI will be ignored instead of forcing the targeted processor to execute from the given address.

B.5 AP Shutdown Handling

An AP may be shut down by itself, by the BSP, or by another active AP. Shutting down an AP with an active task or a bound device driver is not permitted. Only the BSP may shut itself down, and it must be the last processor to shut down.

There are several possible alternatives for shutdown handling. Two of the possibilities are presented here by way of example.

In the simplest case, the operating system can define an IPI for shutting down and taking APs off-line. The exact usage and behavior of an IPI that is defined for this purpose is operating system specific and is not defined in this specification.

Should the BSP need to take an AP off-line or to place the AP back into a HALT state, the BSP can also use the INIT IPI with Warm Restart. The operating system places the address of a HLT instruction in the warm-reset vector (40:67), sets the CMOS shut-down code to 0Ah, then sends an INIT IPI to the AP. The INIT IPI causes the AP to enter the BIOS POST routine, where it immediately jumps to the warm-reset vector and executes the operating system's HLT instruction. Only one processor can execute the shutdown routine at any given time, due to the use of the shutdown code. The operating system must not rely on any code being executed after the delivery

of an INIT IPI used to shut down an AP. As a result, the operating system must ensure that any required state information is captured and that caches are flushed as necessary before sending the INIT IPI.

In order to do a complete system shutdown, followed by a warm restart if necessary, the operating system should return the system to a state similar to that at power-on. This includes disabling the Local APIC interrupts (LINT0/LINT1/Local APIC Timer/Error interrupt) on all processors, disabling the Local APIC on all APs and disabling all interrupts at all the I/O APICs in the system. The operating system can use an IPI or an NMI to signal to all APs for per-processor shutdown handling. The operating system may then set the CMOS shutdown code to 0Ah and perform a keyboard controller reset.

B.6 Other IPI Applications

The operating system may use IPIs for other run-time duties, such as handling the various processor caches.

B.6.1 Handling Cache Flush

The MP specification requires that hardware maintain cache coherency. Cache flushing by the operating system should not be required under normal circumstances. The only need for cache flushing by the operating system is prior to powering down a processor.

Should a system-wide cache flush be necessary, the operating system should use the broadcast IPI mechanism to request that each of the processors write back and invalidate its own cache subsystem and then synchronize upon the completion of that activity.

B.6.2 Handling TLB Invalidation

The operating system should use the IPI mechanism to request that each of the processors invalidate its TLBs. The operating system may use a broadcast IPI for this purpose. The BSP and APs should synchronize the completion of their actions either via memory-based semaphores or via targeted return IPIs. The actual IPI vector is operating system dependent.

B.6.3 Handling PTE Invalidation

The operating system should use the IPI mechanism to request that each processor invalidate a specific page-table entry (PTE) if it is cached in that processor's TLBs. The operating system may use a broadcast IPI for this purpose. The BSP and APs should synchronize the completion of their actions either via memory-based semaphores or via targeted return IPIs. The actual IPI vector is operating system dependent.

B.7 Spurious APIC Interrupts

For the 8259, there is a time window in which a spurious interrupt may be misinterpreted as a genuine interrupt. For example, if an interrupt goes inactive just after the first INTA cycle but before the second INTA cycle, the 8259 will also signal this spurious interrupt as a genuine

interrupt. The distributed APIC architecture, by its nature, is more vulnerable to spurious interrupt, because the device interrupt may be latched and recognized without the INTA cycle.

To ensure that spurious interrupts are handled properly, it is strongly recommended that the device drivers must read the status register before servicing the device. In cases where spurious interrupts do occur, the device drivers may simply ignore them.

Note that this spurious interrupt is not related in any way to the Local APIC Spurious Interrupt generated by the Local APIC Error Register (0xFEE00370)

B.8 Supporting Unequal Processors

Some MP operating systems that exist today do not support processors of different types, speeds, or capabilities. However, as processor lifetimes increase and new generations of processors arrive, the potential for dissimilarity among processors increases. The MP specification addresses this potential by providing an MP configuration table to help the operating system configure itself. Operating system writers should factor in processor variations, such as processor type, family, model, and features, to arrive at a configuration that maximizes overall system performance. At a minimum, the MP operating system should remain operational and should support the common features of unequal processors.

C
System Compliance Checklist

Any "NO" answer indicates non-compliance.

Condition	YES	NO
1. PC/AT Compatibility		
Does system contain all necessary MP-compatible circuitry?		
Will system boot and run DOS and Microsoft Windows?		
2. Memory Subsystem		
Are system memory address map, cacheability, and shareability consistent with definitions in Table 3-1?		
Are memory-mapped I/O devices located at the top of the memory address space?		
If the system has external cache:		
Is cache coherence maintained by hardware?		
Is cache flushing supported by hardware?		
Is cache flushing limited to the local caches?		
Are all locked operations visible to all processors?		
Are locked operations guaranteed on aligned memory operations?		
Are memory writes observed externally in same order as programmed?		
3. Multiprocessor Interrupt Control		
Does each processor have its own local APIC?		
Is there a processor designated for booting?		
Does system support either PIC Mode or Virtual Wire Mode at power-on?		
Is Symmetric I/O Mode implemented?		
Are all APIC IDs unique?		
Are all local APIC IDs assigned by hardware or BIOS?		
Do local APIC IDs begin with zero?		
4. Reset Support		
Does system implement software-initiated processor-specific INIT (INIT IPI)?		
Do system soft and hard resets reset all processors?		
5. APIC Interval Timer		
On 82489DX, is CLK the only clock source of the APIC timer?		
6. Fault Resilient Booting		
If fault resilient booting feature is implemented:		
Is fault resilient booting feature software transparent?		
Are system signals: NMI, INTR, FERR#, IGNNE#, and A20M connected to BSP?		
7. MP Floating Pointer Structure and MP Configuration Table		
Is the MP floating pointer structure implemented?		
If the MP feature byte 1 (default configuration type) is nonzero:		
Does the system correctly implement one of the default configurations?		
If the MP feature byte 1 (default configuration type) is zero:		
Has the system created a correct MP configuration table?		
If the IMCR presence bit is set, is PIC Mode implemented?		
If the IMCR presence bit is not set, is Virtual-Wire Mode implemented?		

D
Multiple I/O APIC Multiple PCI Bus Systems

The information in this specification describes the majority of multiprocessor systems. This appendix provides clarifications for implementors who are considering designs with more than one I/O APIC. In particular, a number of proposed systems will incorporate multiple I/O APICs in order to support multiple PCI buses. This appendix provides guidance for implementors who wish to be sure that their designs comply with this specification.

D.1 Interrupt Routing with Multiple APICs

Two basic approaches to routing interrupts can be used when the system has more than one I/O APIC:

- The fixed routing scheme uses the same routing in both PIC or Virtual Wire Mode and symmetric I/O mode.
- The variable approach changes the routing when switching to symmetric I/O mode.

This section applies when a PCI interrupt is connected both to an I/O APIC input of its own and to the I/O APIC input of the EISA/ISA IRQ to which the interrupt is routed when in PIC or Virtual Wire Mode. This double routing is typically used to preserve PC AT compatibility at system start-up, allowing a system to boot from a disk connected to a PCI controller on a second PCI bus, for example. To prevent double delivery of this PCI interrupt once the system switches to symmetric I/O mode for an MP operating system, the duplicate routing must either be turned off or concealed from the operating system. If a PCI interrupt is only connected via an EISA/ISA IRQ, the EISA/ISA entry in the MP configuration table is sufficient to describe the routing.

The variable routing method described below is preferred since it is more flexible and offers best use of available system resources. Fixed routing is described here for compatibility with existing systems that do not implement a variable routing strategy.

D.1.1 Variable Interrupt Routing

In systems with variable interrupt routing, all PCI interrupts map to EISA/ISA IRQs when in PIC or Virtual Wire Mode. When switched to symmetric I/O mode, the system disables this routing and delivers the PCI interrupt through I/O APIC inputs different from those used by the EISA/ISA IRQs.

If IMCR is implemented, the hardware design can use this bit to enable/disable the routing of the PCI interrupts to EISA/ISA IRQs. On systems with IMCR, this operation might be the only one that is required of the operating system when switching to symmetric I/O mode, other than the actual programming of the I/O and Local APICs.

If IMCR is implemented but the system includes one or more I/O APICs that are not controlled through IMCR, the hardware must accomplish routing changes for such I/O APICs by some other means when the system switches into symmetric I/O mode. These routing changes must be done without requiring any additional intervention from software.

For systems without the IMCR register, the routing of the PCI interrupts to the EISA/ISA IRQ must be automatically disabled as the I/O APICs are programmed. Therefore, when the operating system programs the I/O APICs in accordance with the MP configuration table, the hardware must detect this operation and disable the routing mechanism without additional intervention by the operating system. This operation can be done globally for an entire system as soon as any APIC interrupts are enabled or it can be done on an interrupt-by-interrupt basis.

D.1.2 Fixed Interrupt Routing

Several implementations of fixed interrupt routing are possible, depending on hard wiring, via jumpers for example, or software means, such as chipset-specific registers. Since these implementations have no mechanism to disable the PCI interrupt to EISA/ISA IRQ routing, the MP configuration table must be set up carefully to avoid problems with duplicate interrupt mappings in symmetric I/O mode.

To avoid such problems on systems with fixed routing, PIC or Virtual Wire Mode interrupt routings must not be used by software when the system is in symmetric mode, since these routings cannot be disabled or altered. This situation implies two restrictions that must be placed on the way PCI interrupts are routed to EISA/ISA IRQs and on that way the MP configuration table is built:

- If a PCI interrupt is routed to an EISA/ISA IRQ that is used by an EISA/ISA device, that PCI interrupt must be delivered through the same I/O APIC input as that EISA/ISA IRQ. The connection from the PCI interrupt to the additional I/O APIC input must not be entered in the MP configuration table.
- If a PCI interrupt is routed to an EISA/ISA IRQ which is NOT used by an EISA/ISA device, that PCI interrupt must be delivered through its individual I/O APIC connection, and the connection of that EISA/ISA IRQ to its I/O APIC input should not be entered in the MP configuration table.

As an example, take a system with two I/O APICs, a PCI bus, and an EISA bus. Each EISA IRQ is connected to an input of one I/O APIC and each PCI interrupt is connected to the other I/O APIC. A fixed interrupt routing design could connect all PCI interrupts to a single EISA interrupt. This design does not give optimum performance, because PCI interrupts must be shareable, but it does allow all interrupts to be properly handled.

If an EISA device is connected to the same interrupt, the MP configuration table would not contain any entries for the second I/O APIC. The second I/O APIC is not used.

If no EISA device uses that interrupt, however, the MP configuration table could contain an entry for each PCI interrupt, describing its connection to the second I/O APIC. The EISA IRQ used by the PCI interrupts in PIC mode would not have an entry in the table. All other EISA IRQs would have an entry in the table describing connections to the first I/O APIC.

Fixed interrupt routing also implies a restriction on software that is implicit but important in the context of systems with more than one I/O APIC. The operating system must program I/O APICs to handle only the interrupts for which the MP configuration table contains corresponding I/O interrupt assignment entries. If the configuration table contains no entry for a given I/O APIC input, that interrupt must be left in the masked state.

D.2 Bus Entries in Systems with More Than One PCI Bus

To accommodate systems with more than one PCI bus within the confines of version 1.1 of this specification, construction of the bus entries on the MP configuration table must be handled in a very particular sequence:

1. Begin with bus entries for the PCI buses. Start at bus zero, using the actual PCI bus number as the bus ID for the bus entry.
2. Add entries for other buses. These entries can use bus ID numbers left vacant by the PCI bus entries.

This sequence implies that bus ID numbers do not have to increase sequentially by increments of one; the requirement is that they must appear in ascending order by bus ID number. This specific interpretation of the information presented in Table 4-7 ensures consistency between the information in the MP configuration table and the model for systems with multiple PCI buses that is presented in the formal PCI specification, which allows for more flexibility in bus numbering.

This numbering scheme requires bus entries in the MP configuration table to be sorted appropriately. For example, bus entries should appear in the order PCI (0), EISA (1), and PCI(4) in a system with three buses, two PCI buses numbered 0 and 4, and a single EISA bus numbered as 1.

D.3 I/O Interrupt Assignment Entries for PCI Devices

Section 4.3.4 defines the format of interrupt assignment entries. The example presented there does not, however, completely explain the semantics of the source bus IRQ field for PCI devices.

For PCI devices, the semantics for encoding PCI interrupts should mirror the PCI specification as follows:

Table D-1. I/O Interrupt Entry Source Bus IRQ Field for PCI Devices

Field	Offset (in bytes:bits)	Length (in bits)	Description
SOURCE BUS IRQ	5:0	2	Identifies the PCI interrupt signal, where 0x0 corresponds to INT_A#, 0x1 to INT_B#, 0x2 to INT_C# and 0x3 to INT_D#.
SOURCE BUS IRQ	5:2	5	Gives the PCI Device Number where the interrupt originates.
RESERVED	5:7	1	Reserved for future use.

E
Errata

The following sections provided here are intended to replace the corresponding sections in the main body of the specification. The sections provided here are shown in their entirety to provide context for the changes. Changes contained below are marked with double underlined text which shows changes relative to the version of the sections that are provided in the main body of the specification.

4.1 MP Floating Pointer Structure

An MP-compliant system must implement the MP floating pointer structure, which is a variable length data structure in multiples of 16 bytes. Currently, only one 16-byte data structure is defined. It must span a minimum of 16 contiguous bytes, beginning on a 16-byte boundary, and it must be located within the physical address as specified in the previous section. To determine whether the system conforms to the MP specification, the operating system must search for the MP floating pointer structure in the order specified in the previous section. Figure 4-2 shows the format of this structure, and Table 4-1 explains each of the fields.

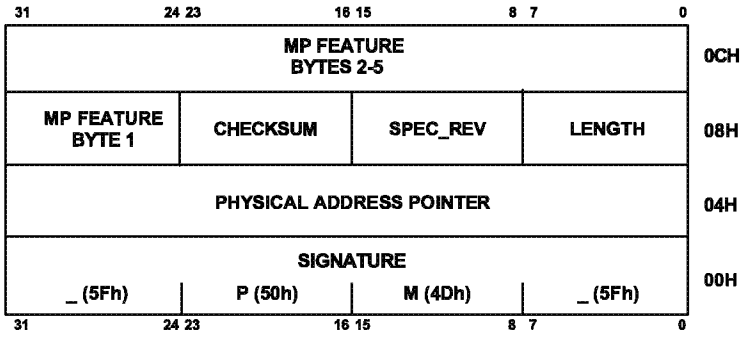


Figure 4-2. MP Floating Pointer Structure

Table 4-1. MP Floating Pointer Structure Fields

Field	Offset (in bytes:bits)	Length (in bits)	Description
SIGNATURE	0	32	The ASCII string represented by "_MP_" which serves as a search key for locating the pointer structure.
PHYSICAL ADDRESS POINTER	4	32	The address of the beginning of the MP configuration table. All zeros if the MP configuration table does not exist.
LENGTH	8	8	The length of the floating pointer structure table in paragraph (16-byte) units. The structure is 16 bytes or 1 paragraph long; so this field contains 01h.
SPEC_REV	9	8	The version number of the MP specification supported. A value of 01h indicates Version 1.1. A value of 04h indicates Version 1.4.
CHECKSUM	10	8	A checksum of the complete pointer structure. All bytes specified by the length field, including CHECKSUM and reserved bytes, must add up to zero.
MP FEATURE INFORMATION BYTE 1	11	8	**Bits 0-7:** MP System Configuration Type. When these bits are all zeros, the MP configuration table is present. When nonzero, the value indicates which default configuration (as defined in Chapter 5) is implemented by the system.
MP FEATURE INFORMATION BYTE 2	12:0 12:6	6 1	**Bits 0-5:** Reserved for future MP definitions. Bit 6: Multiple Clock Sources. When set, this bit indicates that the processors derive clock signals from different sources. Otherwise when not set this bit indicates that all processors share a single clock source.
	12:7	1	**Bit 7:** IMCRP. When the IMCR presence bit is set, the IMCR is present and PIC Mode is implemented; otherwise, Virtual Wire Mode is implemented.
MP FEATURE INFORMATION BYTES 3-5	13	24	Reserved for future MP definitions. Must be zero.

The MP feature information byte 1 specifies the MP system default configuration type. If nonzero, the system configuration conforms to one of the default configurations. The default configurations, specified in Chapter 5, may only be used to describe systems that always have two processors installed.

Bit 6 of MP feature information byte 2, the Multiple Clock Source bit, is used by the operating system to determine how processors derive clock sources. If the system design does not provide for a single clock source shared by all processors then this bit is set. Otherwise this bit is zero to indicate that all processors derive their clock from a common source. Bit 7 of MP feature

information byte 2, the IMCR present bit, is used by the operating system to determine whether PIC Mode or Virtual Wire Mode is implemented by the system.

The physical address pointer field contains the address of the beginning of the MP configuration table. If it is nonzero, the MP configuration table can be accessed at the physical address provided in the pointer structure. This field must be all zeros if the MP configuration table does not exist.

4.4.1 System Address Space Mapping Entries

System Address Space Mapping entries define the system addresses that are visible on a particular bus. Each bus defined in the Base Table can have any number of System Address Space Mapping entries included in the Extended Table. Thus, individual buses can be configured to support different address ranges, thereby decreasing the amount of bus traffic on a given bus and increasing the overall system performance. Each consecutive address range that is usable by the operating system to access devices on a given bus has a System Address Space Mapping entry. Figure 4-9 shows the format of each entry, and Table 4-14 explains each field.

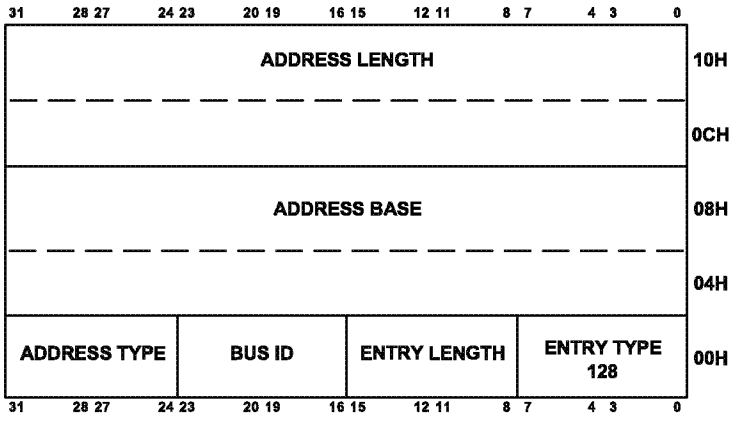


Figure 4-9. System Address Space Entry

Table 4-14. System Address Space Mapping Entry Fields

Field	Offset (in bytes:bits)	Length (in bits)	Description
ENTRY TYPE	0	8	Entry type 128 identifies a System Address Space Mapping Entry.
ENTRY LENGTH	1	8	A value of 20 indicates that an entry of this type is twenty bytes long.
BUS ID	2	8	The BUS ID for the bus where the system address space is mapped. This number corresponds to the BUS ID as defined in the base table bus entry for this bus.
ADDRESS TYPE	3	8	System address type used to access bus addresses must be: 0 = I/O address 1 = Memory address 2 = Prefetch address All other numbers are reserved.
ADDRESS BASE	4	64	Starting address
LENGTH	12	64	Number of addresses which are visible to the bus

If any main memory address is mapped to a software visible bus, such as PCI, it must be explicitly declared using a System Address Space Mapping entry.

In the case of a bus that is directly connected to the main system bus, system address space records and compatibility base address modifiers must be provided as needed to fully describe the complete set of addresses that are mapped to that bus. For example, in Figure 4-10, complete explicit descriptions must be provided for PCI BUS 0 and PCI BUS 1 even if one of the buses is programmed for subtractive decode.

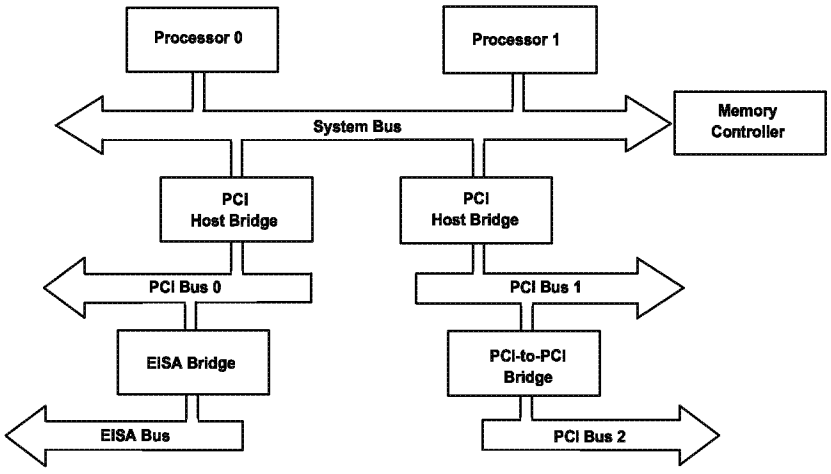


Figure 4-10. Example System with Multiple Bus Types and Bridge Types

Since all device settings must fall within supported System Address Space mapping for a given bus in order to be usable by the operating system, buses that do not support dynamically configurable devices (i.e., ISA, EISA) should support all possible addresses to that bus.

In general, the MP configuration table must provide entries to describe system address space mappings for all I/O buses present in the system. There are two exceptions to this rule:

1. For buses that are connected via PCI-to-PCI-bridge-specification-compliant bridges: in this case, the system address space mappings for such buses may be omitted from the configuration table if bus hierarchy entries for these buses are also omitted (refer to Section 4.4.2 for additional information). In such cases, the operating system is expected to discover the address space mapping by querying the PCI-to-PCI-bridge-specification-compliant bridge directly.
2. For buses that are connected via a parent I/O bus and for which the subtractive decode bit is set (refer to Section 4.4.2 for details).

Typically, this would mean that a minimal description of resources allocated to PCI buses in a system need only include System Address Space Mapping entries for PCI bus zero and any additional peer PCI buses (if present) where these buses are connected by PCI bridges that are specific to the chipset or host bus.

Note

System Address Mappings are unidirectional in nature. They only describe system addresses that propagate to the target bus from any given processor. For DMA support on the target bus, all memory addresses that contain real memory should be accessible directly by either the bus or by a bus-specific DMA controller. For buses with fewer than 32-bit address lines, all real memory at addresses that the bus can generate must be accessible for DMA.

4.4.2 Bus Hierarchy Descriptor Entry

If present, Bus Hierarchy Descriptor entries define how I/O buses are connected relative to each other in a system with more than one I/O bus. Bus Hierarchy Descriptors are used to supplement System Address Mapping entries to describe how addresses propagate to particular buses in systems where address decoding cannot be completely described by System Address Space Mapping entries alone.

In general, entries of this type are required for each bus that is connected to the system hierarchically below another I/O bus. The one exception to this is for PCI buses that are connected behind a PCI to PCI bridge specification compliant bridge. In this case the bus hierarchy entry may be omitted as the operating system is expected to discover such buses without the need to refer to the configuration table. Since bus hierarchy descriptor and system address space records are intended to work in concert, the configuration table may chose to omit both types of records for a bus behind a compliant bridge or it may provide both. The table must not however provide one without the other for buses behind compliant bridges. For example, given the system described in Figure 4-10, bus hierarchy entries are required for the EISA bus and may be provided for PCI BUS 2 since both have parent buses that are themselves I/O buses. If the configuration table does include a bus hierarchy entry for PCI BUS 2, then a complete set of corresponding system address space records must also be provided.

The Bus Hierarchy entry provides information about where in a hierarchical connection scheme a given bus is connected and the type of address decoding performed for that bus. Figure 4-11 shows the format of each entry, and Table 4-15 explains each field.

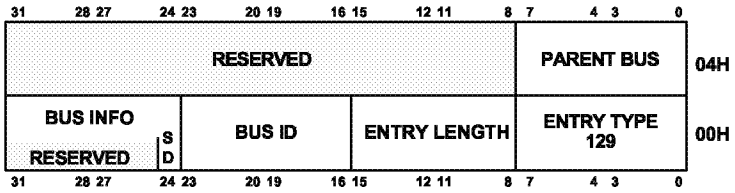


Figure 4-11. Bus Hierarchy Descriptor Entry

Table 4-15 Bus Hierarchy Descriptor Entry Fields

Field	Offset (in bytes:bits)	Length (in bits)	Description
ENTRY TYPE	0	8	Entry type 129 identifies a Bus Hierarchy Descriptor Entry.
ENTRY LENGTH	1	8	A value of 8 indicates that this entry type is eight bytes long.
BUS ID	2	8	The BUS ID identity of this bus. This number corresponds to the BUS ID as defined in the base table bus entry for this bus.
BUS INFORMATION:SD	3:0	1	Subtractive Decode Bus. If set, all addresses visible on the parent bus but not claimed by another device on the parent bus (including bridges to other buses) are useable on this bus.
PARENT BUS	4	8	Parent Bus. This number corresponds to the BUS ID as defined in the base table bus entry for the parent bus of this bus

For buses where the BUS INFORMATION:SD bit is set, System Address Mappings may not be needed. Since the bus is defined as being subtractive decode, the range of addresses that appear on the bus can be derived from address decoding information for parent and peer buses.

Glossary

82489DX: The 82489DX Advanced Programmable Interrupt Controller (APIC).

8259A: The 8259A Programmable Interrupt Controller (PIC) or its equivalent.

AP: Application processor, one of the processors not responsible for system initialization.

APIC: Advanced Programmable Interrupt Controller, either the 82489DX APIC or the integrated APIC on Pentium processors.

BIOS: Basic Input/Output Subsystem.

BSP: Bootstrap processor, the processor responsible for system initialization.

Cache coherency: A property of a cache/memory system that guarantees that a request for an item from memory will retrieve the most up-to-date value of that item, regardless of what cache or memory location currently holds that value.

CMOS RAM: The battery backed-up configuration memory of the PC/AT motherboard.

DP: A dual processor system is one with two processors.

ExtINT: A delivery mode of the Local Vector Table of a local APIC that causes delivery of a signal to the INT pin of the processor as an interrupt that originated in an externally connected 8259A-equivalent PIC. The ExtINTA output signal is also asserted. The INTA cycle that corresponds to the ExtINT delivery should be routed to the external PIC that is expected to supply the vector.

Flush: Write back all modified lines of a cache.

INIT: Unless otherwise specified, the processor-specific reset or system-wide soft reset functions. This definition is functional and sometimes bears no relationship to the actual signal name. For example, the term "INIT" may refer to the INIT signal on the Pentium processor or to the RESET signal on the Intel486 processor.

INIT IPI: A type of APIC interprocessor interrupt whose delivery mode is set to RESET. Upon delivery to the specified destinations, the destination APICs assert their PRST output signals. When the PRST lines are connected to the INIT or RESET inputs of their respective processors, an INIT IPI causes reinitialization of the destination processors.

Invalidate: Change the state of a cache line to the Invalid state.

IPI: Interprocessor interrupt.

MESI: A cache coherency protocol named after the states that cache lines may have: Modified, Exclusive, Shared, Invalid.

MP: A multiprocessor system is one with two or more processors.

PIC: Programmable Interrupt Controller.

PIC Mode: One of three interrupt modes defined by the MP specification. In this mode the APICs are effectively disabled, while interrupts are generated by 8259A-equivalent PICs and delivered directly to the BSP. This is a uniprocessor compatibility mode.

POST: Power-On Self Test, the first BIOS procedure executed after a RESET or INIT.

RESET: The system-wide hard reset. This definition is functional. It may refer to the RESET signal on both Pentium and Intel486 processors or the RESET signal of the 82489DX APIC.

Shutdown code: The value of CMOS RAM location 0Fh, which indicates that reason that a RESET was performed.

STARTUP IPI: A type of APIC interprocessor interrupt that is similar to an NMI with an embedded vector. It does not cause any change of state, but merely causes the targeted processor to start executing in Real Mode from address 000VV000h, VV being an 8-bit vector which is part of the IPI message. Startup vectors are limited to a 4K page boundary in the first 1 MB of the address space. STARTUP IPIs are not maskable, and can be issued at any time. The benefit of STARTUP-IPI compared to NMI is that it does not require the targeted APIC to be enabled, and it does not require the interrupt table to be programmed. Thus, the operating system's initialization procedure can use it to wake up an AP that has been sleeping since RESET or INIT. The STARTUP IPI is not supported by the 82489DX APIC.

Symmetric I/O Mode: One of three interrupt modes defined by the MP specification. In this mode, the APICs are fully functional, and interrupts are generated and delivered to the processors by the APICs. Any interrupt can be delivered to any processor. This is the only multiprocessor interrupt mode.

Symmetry: The relationship of equality among components of a multiprocessor system in which no processor is special with respect to its access to memory, interrupts, or I/O. For interrupts, symmetry means that any interrupt from any source can be routed to any processor and handled there. For I/O, it means that all I/O control registers, be they in memory space, I/O space, or some other special address space, are accessible to all processors. Per-processor control hardware, such as interrupt controllers or processor identification registers, must be at the same physical address for all processors.

Virtual Wire Mode: One of three interrupt modes defined by the MP specification. In this mode interrupts are generated by the 8259A-equivalent PICs, but delivered to the BSP by an APIC that is programmed to act as a "virtual wire"; that is, the APIC is logically indistinguishable from a hardwired connection. This is a uniprocessor compatibility mode.

Warm reset: A technique that allows the RESET or INIT signal to be asserted without actually causing the BIOS to run through its entire initialization procedure. If a value of 0Ah is placed in the **shutdown code**, the first instructions of the BIOS POST procedure read the warm-reset vector from system RAM location 40:67h, and jump to that address.

Write-back: A cache update policy wherein a modified cache line is not written back to main memory until the last possible instant—when another processor needs to access the data or when its location in the cache is needed for other data.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re patent of Michaelis *et al.*	§	Attorney Docket No.: 185945.012600
	§	
U.S. Patent No. 7,624,302	§	Customer No.: 67395
	§	
Issue Date: November 24, 2009	§	
	§	
Filing Date: October 25, 2004	§	
	§	
For: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR	§	

REQUEST FOR *EX PARTE* REEXAMINATION OF U.S. PATENT 7,624,302

Mail Stop "*Ex Parte* Reexam"
Attn: Central Reexamination Unit
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Commissioner:

Pursuant to the provisions of 35 U.S.C. §§ 301-307, Unified Patents, LLC ("Requester") hereby requests an *ex parte* reexamination of claims 1-27 (the "Challenged Claims") of U.S. Patent 7,624,302 (the "'302 Patent, Ex. 1001), which issued on November 24, 2009, based on U.S. Patent Application 10/972,588 (the "'588 Application"), which was filed on October 25, 2004. The *'302 Patent* was originally assigned to Hewlett-Packard Development Company, LP, as reflected in the assignment recorded in the U.S. Patent and Trademark Office ("USPTO") at reel/frame 015932/0117 and is currently assigned to Foras Technologies LTD., as reflected in the assignment recorded in the USPTO at reel and frame number 058992/0571.

This Request presents prior art references and analyses that are noncumulative of the prior

art that was before the Examiner during the original prosecution of the '302 Patent. As demonstrated herein, the Challenged Claims are invalid over these references. Requester therefore requests that an order for reexamination and an Office Action rejecting claims 1-27 be issued.

Ex Parte Patent Reexamination Filing Requirements

Pursuant to 37 C.F.R. § 1.510(b)(1), statements pointing out at least one substantial new question of patentability based on material, non-cumulative reference patents and printed publications for the Challenged Claims of the '302 Patent are provided in Section I of this Request.

Pursuant to 37 C.F.R. § 1.510(b)(2), reexamination of the Challenged Claims (1-27) of the '302 Patent is requested, and a detailed explanation of the pertinence and manner of applying the cited references to the Challenged Claims is provided in Section II of this Request.

Pursuant to 37 C.F.R. § 1.510(b)(3), copies of every patent or printed publication relied upon or referred to in the statement pointing out each substantial new question of patentability or in the detailed explanation of the pertinence and manner of applying the cited references are provided as Exhibits 1002-1017 (with Exhibits AA-CC forming the basis for the SNQ presented herein).

Pursuant to 37 C.F.R. §1.510(b)(4), a copy of the '302 Patent is provided as Exhibit 1001 of this Request, along with a copy of any disclaimer, certificate of correction, and reexamination certificate issued corresponding to the patent.

Pursuant to 37 C.F.R. § 1.510(b)(5), the attached Certificate of Service indicates that a copy of this Request, in its entirety, has been served on Patent Owner at the following address of record for Patent Owner, in accordance with 37 C.F.R. § 1.33(c):

FORAS TECHNOLOGIES LTD.
c/o 56436 - Hewlett Packard Enterprise
3404 E. Harmony Road
Mail Stop 79
Fort Collins, CO 80528

Also submitted herewith is the fee set forth in 37 C.F.R. § 1.20(c)(1).

Pursuant to 37 C.F.R. § 1.510(b)(6), Requester hereby certifies that the statutory estoppel provisions of 35 U.S.C. § 315(e)(1) and 35 U.S.C. § 325(e)(1) do not prohibit Requester from filing this *ex parte* patent reexamination request.

TABLE OF CONTENTS

TABLE OF EXHIBITS iii

I. SUBSTANTIAL NEW QUESTIONS OF PATENTABILITY 4

A. U.S. Patent 7,624,302 4

1. Technical Background 4

2. Summary 5

3. Discretionary Denial is not Warranted 9

4. Challenged Claims 9

5. Priority 12

6. Prosecution History 12

B. Level of Ordinary Skill in the Art 15

C. Claim Construction 15

1. Overview 15

2. "boot processor" (Claims 1, 2, 6, 13, 14, 17, 21, and 23-26) 16

D. Claim Interpretation under 35 U.S.C. 112(f) 18

1. "system firmware means for detecting . . ." (Claim 26) 18

2. "means for determining . . ." (Claim 26) 19

3. "system firmware means for switching . . ." (Claim 26) 19

E. List of Prior Art Patents and Printed Publication 20

F. Ground 1: Kalyanasundharam in view of Kennedy Presents Substantial New Questions of Patentability for Claims 1, 5-8, 13, 17, 21, 22, and 25-26 23

1. Overview of Kalyanasundharam 23

2. Overview of Kennedy 27

3. Motivation to Combine Kalyanasundharam and Kennedy 30

G. Ground 2: Kennedy in view of Nguyen and Safford Presents Substantial New Questions of Patentability for Claims 1-27 33

1. Overview of Kennedy 33

2. Overview of Nguyen 33

3. Overview of Safford 36

4. Motivation to Combine Kennedy, Nguyen, and Safford 38

H. Ground 3: Safford in light of Cepulis Presents Substantial New Questions of Patentability for Claims 1-4 46

1. Overview of Safford 46

2. Overview of Cepulis 46

II. DETAILED APPLICATION OF THE PRIOR ART TO EVERY CLAIM FOR WHICH REEXAMINATION IS REQUESTED 47

A. Ground 1 (Kalyanasundharam in view of Kennedy) 47

B. Ground 2 (Kennedy in view of Nguyen and Safford) 57

C. Ground 3: (Safford in view of Cepulis) 71

D. Secondary Considerations 76

III. DISCLOSURE OF CONCURRENT LITIGATION, REEXAMINATION, AND RELATED PROCEEDINGS 76

IV. CONCLUSION 77

TABLE OF EXHIBITS

Exhibit	Description
Ex. 1001	U.S. Patent No. 7,624,302 to Michaelis et al. (the "'302 Patent")
Ex. 1002	U.S. File History of the '302 Patent ("U.S. File History")
Ex. 1003	Declaration of Andrew Wolfe ("Wolfe Decl.")
Ex. 1004	U.S. Patent Application Pub. No. 2004/0123201 to Nguyen et al. ("Nguyen")
Ex. 1005	U.S. Patent No. 7,328,371 to Kalyanasundharam et al. ("Kalyanasundharam")
Ex. 1006	U.S. Patent No. 5,450,576 to Kennedy et al. ("Kennedy")
Ex. 1007	U.S. Patent No. 7,085,959 to Safford ("Safford")
Ex. 1008	U.S. Patent No. 6,625,749 to Quach ("Quach")
Ex. 1009	U.S. Patent Pub. No. 2005/0172164 to Fox ("Fox")
Ex. 1010	Wendy Bartlett, et. al., *Commercial Fault Tolerance: A Tale of Two Systems, IEEE Transactions on Dependable and Secure Computing*, Vol. 1, No. 1 (January-March 2004) ("Bartlett")
Ex. 1011	Intel, *Extensible Firmware Interface Specification*, Version 1.10 (Dec. 1, 2002) ("Intel EFI Specification")
Ex. 1012	U.S. Patent No. 7,191,328 to Hobson ("Hobson")
Ex. 1013	Bozidar Levi, *UNIX Administration, A Comprehensive Sourcebook for Effective Systems and Network Management* (2002) ("Levi")
Ex. 1014	U.S. Patent No. 5,491,788 to Cepulis et al. (Cepulis)
Ex. 1015	*Intel IA-64 Architecture Software Developer's Manual*, Revision 1.1 (July 2000) ("Intel Itanium Manual")
Ex. 1016	*Intel Multiprocessor Specification*, Version 1.4 (May 1997) ("Intel Multiprocessor Specification")
Ex. 1017	U.S. Patent No. 5,915,082 to Marshall ("Marshall")
Ex. AA	Appendix A: Claim chart comparing claims to Kalyanasundharam in view of Kennedy, as set forth in Ground 1.
Ex. BB	Appendix B: Claim chart comparing claims to Kennedy in view of Nguyen and Safford, as set forth in Ground 2.
Ex. CC	Appendix C: Claim chart comparing claims to Safford in light of Cepulis, as set forth in Ground 3.

I. SUBSTANTIAL NEW QUESTIONS OF PATENTABILITY

A. U.S. Patent 7,624,302

1. Technical Background

Since the 1960s and 1970s, computer system designers, such as IBM and Tandem, have been developing and releasing fault-tolerant systems with high availability. *Bartlett* (Ex. 1010) at 89. For example, in 1964, IBM shipped the S/360, which "was designed with robust error checking and was the first commercial computer to include Error Correction Code (ECC) in memory." *Id.* As another example, the Tandem NonStop system shipped in 1976 and included "processor pairs [that] are an effective design for recovering from transient problems." *Id.* at 91. Tandem and IBM both continually improved these systems, and by the early 1990s Tandem had moved to "tight lockstepping of a pair of MIPS chips" with a variant "being developed for future Intel Itanium(R)-based platforms." *Id.* at 91-92. For its part, in 2001 IBM introduced its "Autonomic Computing initiative," which focused on self-healing that included "dynamic CPU sparing, a process by which a failed CPU can be transparently replaced by a spare." *Id.* at 95. These systems were often operating-system centric, with "the operating system provid[ing] its own layer of fault detection and fail-fast response [and supplying] the infrastructure that allows applications to survive single processor failures." *Id.* at 89.

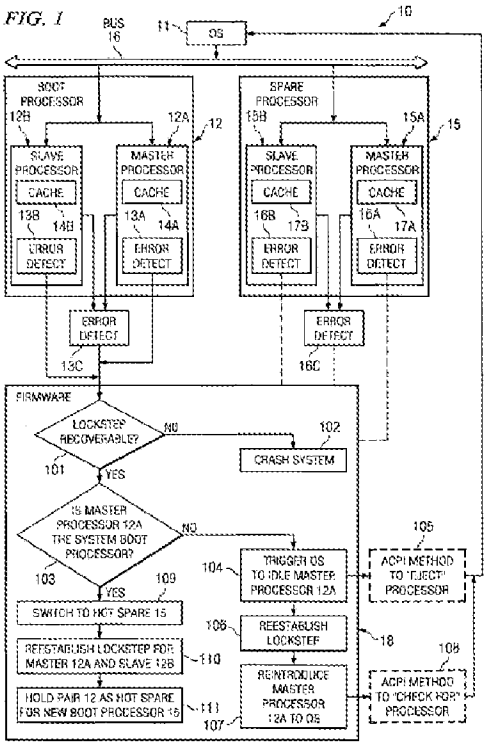
Around the same time that IBM and Tandem were iterating on their high-availability systems in which the OSs managed the hardware, Intel was working on an initiative that attempted to address some of challenges with OS management of hardware and released the Extensible Firmware Interface (EFI). *See Intel EFI Specification* (Ex. 1011) at 1-6. One of the primary goals of the EFI was "abstraction of the OS from the firmware" to allow the OS "to be constructed with far less knowledge of the platform and firmware that underlie those interfaces." *Id.* In other words, Intel proposed "a backward-compatible solution in which the EFI layer is configured to emulate certain aspects of legacy BIOS operation." *Hobson* (Ex. 1012) at 1:43-46.

Within this this historical context, on October 25, 2004, Applicants for the '302 Patent filed a patent application claiming that they had invented a novel process for "detecting loss-of-lockstep (LOL) for a processor in a multiprocessor system" and maintaining high availability by "changing a system identifier of [a] spare processor to that of a [failed] boot processor . . . without shutting down the system's operating system." *'302 Patent* (Ex. 1001), Claim 1. However, as noted above and shown in greater detail below, detecting and recovering from loss-of-lockstep was not new,

and abstracting and emulating hardware management processes from an OS in a manner that allowed an OS to continue running through recovery was not new either. Furthermore, as explained in detail herein, it would have been obvious to combine these limitations in the manner claimed in the '302 Patent. And, as also explained herein, not only was the claimed invention of the '302 Patent obvious, the inventors of the '302 Patent were not the first to propose this claimed combination.

2. Summary

As noted, the '302 Patent is directed to a method for "detecting loss of lockstep (LOL) for a processor in a multi-processor system." Ex. 1001, Abstract. The method includes "determining that the processor for which the LOL is detected is assigned the role of boot processor, and switching the role of boot processor to a spare processor without shutting down the system's operating system." *Id.* Figure 1 below shows an example multi-processor system as described in the '302 Patent.



Ex. 1001, FIG. 1

The '302 Patent notes that, in some cases, processing errors may happen due to random occurrences. For example, alpha rays emitted by the sun may randomly flip an electromagnetic bit

within a processor or within a processor's cache memory. *Id.* at 1:64-67. To counteract such occurrences and provide a higher level of reliability, dual processors are set up in "lockstep" to process the same inputs and compare their outputs to ensure that the outputs of both processors are the same. *Id.* at 2:27-34. If the processing outputs are different, the processors are said to be out of lockstep, which may result in system crashes. *Id.* at 3:19-30.

The '302 Patent describes three steps for recovering from loss of lockstep detected on a boot processor in a multi-processor system. Each of these steps will be discussed in turn below.

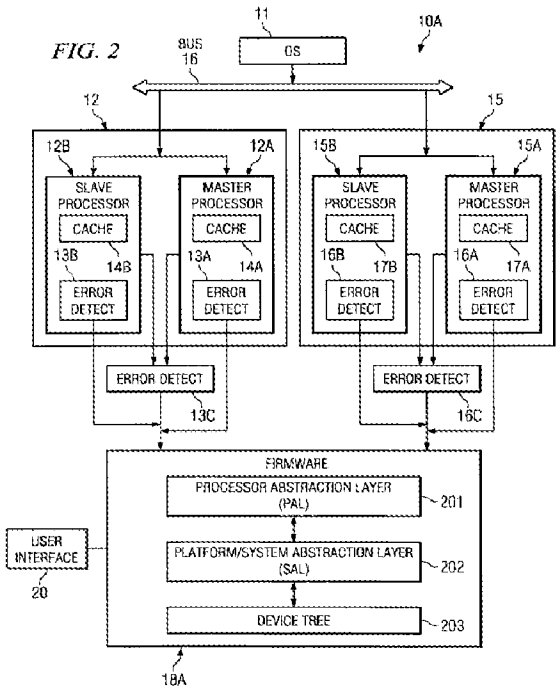
First, the '302 Patent determines that two processors operating in lockstep have experienced a loss of lockstep. *Id.* at 5:1-5. As shown above in FIG. 1 and as described in the specification, "System 10 includes OS 11, as well as master processor 12A and slave processor 12B (collectively referred to as a **lockstep processor pair 12**)." *Id.* at 6:45-47. In some cases, the master processor and slave processor are two different CPU cores of the same silicon chip, while in other cases, the master and slave processors are two separate processors. *Id.* at 6:47-53. In the example of FIG. 1, "**lockstep processor pair 12 is designated as the boot processor** for system 10." *Id.* at 58-60. "In certain embodiments lockstep processor pair 12 is designated as the system boot processor **by storing information to the system that specifies such designation**." *Id.* at 6:60-63. Both the master processor and the slave processor are coupled to the main communication bus 16 via an interface, but that interface "only communicates the output of master processor 12A back onto bus 16." *Id.* at 7:1-4. "The output of slave processor 12B is used solely for checking the output of master processor 12A." *Id.* at 7:4-5. [1]

The master and slave processors in the '302 Patent each include error detection logic for detecting errors such as data cache errors or lockstep mismatches. *Id.* at 7:6-13. "Error detect logic 13C is also included, which may include an XOR (exclusive OR) gate, for **detecting a lockstep mismatch between master processor 12A and slave processor 12B**." *Id.* at 7:15-18. "System 10 further includes a second lockstep processor pair 15 that is communicatively coupled to bus 16." *Id.* at 7:37-40. "Lockstep processor pair 15 includes master processor 15A and slave processor 15B." *Id.* at 7:40-41. In the example of FIG. 1, "**lockstep processor pair 15 is designated as a spare for boot processor 12**" *Id.* at 7:50-51.

[1] Unless otherwise specified, all **bold** and *italics* emphases and annotations below have been added. Text in **bold** is used to add emphasis. Text in italics is used in citations and to identify claim language.

Second, the '302 Patent determines that the processor for which LOL is detected is assigned the role of boot processor. "…[T]echniques are provided for switching the role of boot processor to the designated spare processor **responsive to detection of LOL for the boot processor**." *Id.* at 7:53-56. "Thus, the spare processor is used for recovery when LOL is detected for the system's boot processor." *Id.* at 7:56-58. "That is, the designated spare processor **assumes the role of boot processor in the event that LOL is detected for the boot processor**." *Id.* at 7:58-60.

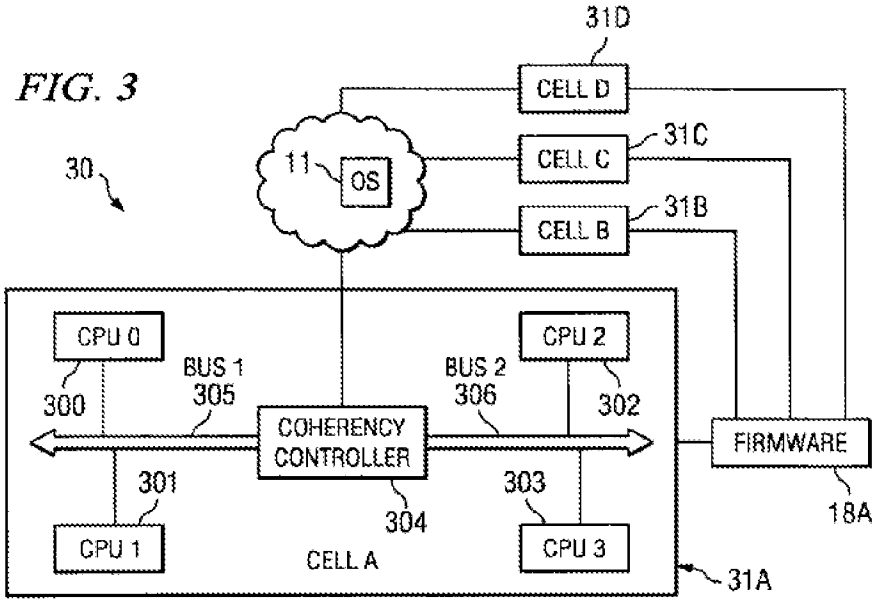
In some cases, the role of boot processor is determined by accessing a device tree. As shown in FIG. 2 below, a device tree 203 may be part of firmware 18A. The device tree "is stored to a data storage device (e.g., SRAM) such that it is accessible to the firmware 18. *Id.* at 9:4-6. "[T]he **device tree includes information identifying processor 12 as the system's boot processor** and identifying processor 15 as the spare." *Id.* at 9:7-9. "The information specifying the boot processor and/or spare may be received as input from a user in certain embodiments." *Id.* at 9:9-11. Or, in other embodiments, "the firmware 18 selects the boot processor and/or spare." *Id.* at 9:14-15. "The **device tree may**, in certain implementations, **include a field for each processor indicating whether such processor is the systems boot processor**." *Id.* at 9:53-56. In some cases, this field is alternatively referred to as a "LID register." *Id.* at 14:37-40.



Ex. 1001, FIG. 2

Third, upon determining that that the processor for which LOL is detected is assigned the role of boot processor, the '302 Patent makes changes that cause the operating system to recognize another processor as a system boot processor without shutting down the operating system. *Id.* at 4:23-27. In FIG. 3 below, for example, four different CPUs (CPU 0, CPU 1, CPU 2, and CPU3) are connected to a coherency controller 304 via Bus 1 (305) and Bus 2 (306). The coherency controller 304 ensures that the proper CPU responds when OS interrupts are received. *Id.* at 4:35-37. Each of the four CPUs is a lockstep pair, including both a master processor and a slave processor. *Id.* at 14:28-31.

FIG. 3



Ex. 1001, FIG. 3

In the example of FIG. 3, "during the boot-up of system 30, **CPU 2 (302) is designated as the boot processor**." *Id.* at 14:29-30. Accordingly, firmware 18A will identify another processor that is communicatively coupled to bus 306 to **designate as a spare** for such boot processor, such as CPU 3 (303)." *Id.* at 14:50-53. "During runtime if a LOL is detected for CPU 2 (302), then firmware 18A may…**switch the designation of boot processor to a designated spare, CPU 3 (303)**." *Id.* at 14:53-57. "In performing this switch the LID register of the CPU that lost lockstep is copied into the LID register of the spare CPU and the spare CPU pair will then assume the ID of the failed CPU and start responding to interrupts directed to that ID." *Id.* at 14:57-60. "**The LID of the failed CPU is quickly re-written to a different value** so that it will no longer respond to interrupts directed to that ID." *Id.* at 14:61-63. "CPU 2 (302) can then be reset to recover its

lockstep, and such CPU 2 (302) **may then be designated as a spare for the new boot processor CPU 3 (303)**." *Id.* at 14:63-65. Firmware 18A may perform this switching of the role of boot processor to a spare processor without shutting down the operating system. *Id.* at 17:1-5.

3. Discretionary Denial is not Warranted

Requester submits that, because a substantial new question is presented herein that has not been previously adjudicated by the U.S. Patent Office, 35 U.S.C. § 325(d) does not warrant denial of this Reexam Request. Indeed, 35 U.S.C. § 325(d) notes that:

> Notwithstanding sections 135(a), 251, and 252, and chapter 30, during the pendency of any post-grant review under this chapter, if another proceeding or matter involving the patent is before the Office, the Director may determine the manner in which the post-grant review or other proceeding or matter may proceed, including providing for the stay, transfer, consolidation, or termination of any such matter or proceeding. In determining whether to institute or order a proceeding under this chapter, chapter 30, or chapter 31, the Director may take into account whether, and reject the petition or request because, the same or substantially the same prior art or arguments previously were presented to the Office.

Accordingly, 35 U.S.C. § 325(d) does not warrant denial of this Reexam Request.

4. Challenged Claims

Claims 1-27 are challenged:

1. A method comprising:

detecting loss of lockstep (LOL) for a processor in a multi-processor system;

determining that the processor for which said LOL is detected is assigned a role of boot processor;

changing a system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected, to a different value, which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor; and

changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to a spare processor without shutting down the system's operating system.

2. The method of claim 1 wherein the spare processor is a hot spare that is held in idle until switched the role of boot processor.

3. The method of claim 1 wherein the spare processor is an active spare.

4. The method of claim 3 wherein said switching further comprises:

causing the system's operating system to idle the active spare.

5. The method of claim 1 wherein the spare processor is a processor module that includes a master and slave processors that operate in lockstep.

6. The method of claim 1 wherein said determining that the processor for which said LOL is detected is assigned the role of boot processor comprises:

accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor.

7. The method of claim 1 further comprising: determining the processor that is assigned the role of spare processor.

8. The method of claim 7 wherein said determining the processor that is assigned the role of spare processor comprises:

accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of spare processor.

9. The method of claim 1 wherein said determining step is performed by system firmware.

10. The method of claim 1 wherein said switching step is performed by system firmware.

11. The method of claim 1 wherein the processor for which said LOL is detected includes a lockstep pair of processors, and wherein said switching comprises:

determining one of the lockstep pair of processors that is not the cause of said LOL; and

copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to said spare processor.

12. The method of claim 11 wherein said spare processor includes a lockstep pair of processors, and wherein said copying comprises:

copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to each of said lockstep pair of processors of said spare processor.

13. A method comprising:

system firmware determining that loss of lockstep (LOL) is detected for a lockstep pair of processors that are assigned the role of boot processor in a system;

determining one of said lockstep pair of processors that is not the cause of said LOL;

copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to a spare processor;

changing a system identifier, used by the system's operating system to recognize the boot processor, of said lockstep pair of processors that are assigned the role of boot processor, to a different value, which causes the s system's operating system to not recognize the lock step pair of processors for which said LOL is detected as the processor; and

changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to said spare processor.

14. The method of claim 13 wherein the spare processor is a hot spare that is held in idle until switched the role of boot processor.

15. The method of claim 13 wherein the spare processor is an active spare.

16. The method of claim 15 further comprising: causing the system's operating system to idle the active spare.

17. The method of claim 13 wherein said switching causes the system's operating system to recognize said spare processor as said boot processor without shutting down said operating system.

18. The method of claim 13 wherein said determining step is performed by system firmware.

19. The method of claim 13 wherein said copying step is performed by system firmware.

20. The method of claim 13 wherein said switching step is performed by system firmware.

21. A system comprising:

a plurality of processor modules that each include a master processor and a slave processor that operate in lockstep;

an operating system;

error detection logic operable to detect loss of lockstep (LOL) for at least one of said processor modules; and

system firmware operable, responsive to the detection of LOL for a first of said processor modules, to determine whether said first processor module is assigned a role of system boot processor;

wherein if determined that said first processor module is assigned the role of system boot processor, said system firmware is further operable to cause said operating system to recognize another processor module as system boot processor without shutting down said operating system.

22. The system of claim 21 wherein the another processor module is a processor module assigned a role of spare processor.

23. The system of claim 21 wherein the another processor module is held in idle until said system firmware causes said operating system to recognize said another processor module as said system boot processor.

24. The system of claim 21 wherein the system firmware causes the operating system to idle the another processor module before causing said operating system to recognize said another processor module as said system boot processor.

25. The system of claim 21 further comprising:

non-volatile data storage storing information that assigns the role of system boot processor to one of said plurality of processor modules.

26. A system comprising:

system firmware means for detecting loss of lockstep (LOL) for a processor in a multi-processor system;

means for determining whether the processor for which said LOL is detected is assigned the role of boot processor; and

system firmware means for switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which said LOL is detected is assigned the role of boot processor.

27. The system of claim 26 wherein said processor in said multi-processor system is an Itanium Processor Family (IPF) processor.

5. Priority

The earliest possible priority date for the '302 Patent is October 25, 2004, which corresponds to the filing date of U.S. Patent Application No. 10/927,588 ("the '588 application"), which issued as the '302 patent. *See* Ex. 1002.

6. Prosecution History

The '588 application was filed on October 25, 2004, with 30 original claims. *U.S. File*

History (Ex. 1002).

Two non-final rejections were issued during the course of prosecution of the '302 Patent. In response to the first and second Office Actions, the Applicant traversed both rejections without amendment, arguing that neither reference specifically taught the assignment of the role of boot processor and then switching the role of boot processor to a spare processor without shutting down the system's operating system. In its replies, Applicant argued that none of the cited references taught or suggested a boot processor or changing the role of boot processor to another processor without shutting down the operating system. Applicant moved the case to allowance with a Supplemental Amendment, amending independent claims 1 and 16 to include the limitations of dependent claims 13-15, as described further below.

The '588 application was first examined in 2007, with a first Non-Final Office Action being sent on September 12, 2007. *Id.* at p. 226. In the first Non-Final Office Action (NFOA), the Examiner rejected claims 1-30 under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1-32 of U.S. Patent Application No. 10/973,077 and claims 1-26 of U.S. Patent Application No. 10/972,796, both of which were filed concurrently with the '302 Patent. *Id.* Further, in the first NFOA, claims 1-4, 7, 9-23, and 29 were rejected under 35 U.S.C. 102(e) as being anticipated by U.S. Patent No. 7,085,959 to Safford (Ex. 1007). Claims 5-6, 8, 24-28, and 30 were rejected under 35 U.S.C. 103(a) as being obvious in view of Safford and U.S. Patent No. 5,915,082 to Marshall et al. (Ex. 1017). *Id.*

In response to the NFOA, the Applicant asked that the double patenting rejection be held in abeyance until allowable subject matter had been reached. Regarding the 35 U.S.C. 102(e) and 103(a) rejections, the Applicant traversed the rejections without amendment, arguing that neither Safford nor Marshall **assigned the role of boot processor to a specific processor** and then **switched the role of boot processor to a spare processor** without shutting down the system's operating system. *Id.* at p. 220. Applicant stated that "Safford does not discuss **transfer of system information (which processor is the boot processor)** that remains permanent and survives a system reset." *Id.* Applicant further argued that Marshall was not a valid prior art reference under 35 U.S.C. 103(c), as both the '302 Patent and the Marshall reference were commonly assigned to the same entity, Hewlett-Packard Development Company. *Id.*

After reviewing this response, the Examiner issued a second, Non-Final Office Action on March 10, 2008. In the second NFOA, the Examiner maintained the obviousness-type double

patenting rejection as presented previously. *Id.* at p. 190. The Examiner also indicated that claims 24-28 were allowable (subject to the double patenting rejection being overcome). The Examiner maintained the 35 U.S.C. 102(e) of claims 1-4, 7, 9-23, and 29 based on the reasons set forth in the first Office Action. *Id.* at p. 192. In response to the Applicant's arguments, the Examiner noted that the concept of switching roles without shutting down the operating system was not present in the rejected claims. *Id.* at p. 194. The Examiner additionally rejected claims 1-23 and 29-30 as being obvious over U.S. Patent No. 6,625,749 to Quach (Ex. 1008) in view of U.S. Patent No. 7,251,749 to Fox (Ex. 1009). *Id.* at p. 197.

In response to the second NFOA on June 5, 2008, the Applicant again requested that the obviousness-type double patenting rejection be held in abeyance and again argued that neither Safford nor Quach in view of Fox taught or suggested assigning processor roles and then switching those roles without shutting down the system's operating system. Applicant stated: "1. Independent claims 1, 16, 24, and 29 specify a boot processor. Safford does not discuss a boot processor." *Id.* at p. 185. "2. Safford does not teach or suggest switching the role of boot processor." *Id.* "3. Rebooting a single processor is not the same as switching the role of boot processor." *Id.* No claims were amended in this second response.

After filing the written response on June 5, 2008, Applicant conducted an interview with the Examiner on September 5, 2008, and filed a Supplemental Amendment on September 10, 2008, amending independent claim 29 to include "system firmware" as the means for performing its designated limitations, and amending independent claims 1 and 16 to include the limitations of dependent claims 13-15 and cancelling claims 13-15. These amendments included:

> *changing a system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected, to a different value, which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor; and*
>
> *changing a system identifier of the spare processor to that of the boot processor, thereby*...
>
> *Id.* at p. 165.

On September 23, 2008, the Examiner issued a Notice of Allowance without any indication as to the reasons for allowance or how the amendment distinguished over the prior art. *Id.* at p. 148. Applicant thereafter submitted an RCE and IDS on December 9, 2008. The Examiner issued a second Notice of Allowance on February 12, 2009, again without any indication as to the reasons

for allowance. *Id.* at p. 113. Applicant then submitted another RCE and IDS on May 12, 2009. *Id.* at p. 107. The Examiner then issued a third Notice of Allowance on July 14, 2009, again without any discussion of how the claims were patentable over the prior art. *Id.* at p. 18. Applicant paid the issue fee on October 6, 2009, and the '302 Patent issued on November 24, 2009. The '302 Patent is subject to a terminal disclaimer filed on September 10, 2008, which disclaims any term beyond U.S. Patent Application No. 10/973,077, filed on October 25, 2004, and U.S. Patent No. 7,366,948, also filed on October 25, 2004. Still further, on February 16, 2010, the U.S. Patent and Trademark Office issued a certificate of correction for the '302 Patent, correcting minor errors in the claims in accordance with previous amendments. *Id.* at p. 6.

B. Level of Ordinary Skill in the Art

A Person of Ordinary Skill in The Art ("POSITA") would have had, as of October 25, 2004: (1) at least a Bachelor's Degree in electrical engineering, electronics engineering, computer engineering, or a related scientific field; and (2) at least one to two years of experience in the computer science field, including chip design and software engineering, with additional education substituting for less experience and vice versa. Wolfe Decl. (Ex. 1003) ¶ 45.

C. Claim Construction

1. Overview

The claims should be construed according to their broadest reasonable interpretation. "During reexamination ordered under 35 U.S.C. 304, and also during reexamination ordered under 35 U.S.C. 257, claims are given the broadest reasonable interpretation consistent with the specification and limitations in the specification are not read into the claims (*In re Yamamoto*, 740 F.2d 1569, 222 USPQ 934 (Fed. Cir. 1984)). Requester reserves the right to advocate a different claim interpretation in any other forum.

The claims of the '302 Patent include limitations that invoke 35 U.S.C. § 112(f) ("112(f)"). 35 U.S.C. § 112(f) (limitations "may be expressed as a means . . . for performing a specified function without the recital of structure, material, or acts in support thereof, and such [limitation] shall be construed to cover the corresponding structure, material, or acts described in the specification and equivalents thereof."). "In making the assessment of whether the limitation in question is a means-plus-function term . . . [,] the essential inquiry is . . . whether the words of the claim are understood by persons of ordinary skill in the art to have a sufficiently definite meaning as the name for structure." *Williamson v. Citrix Online, LLC,* 792 F.3d 1339, 1348 (Fed. Cir. 2015).

2. "boot processor" (Claims 1, 2, 6, 13, 14, 17, 21, and 23-26)

Claims 1, 2, 6, 13, 14, 17, 21, and 23-26 recite a "boot processor," which should be given the broadest reasonable interpretation that is both "consistent with the ordinary and customary meaning of the term" and "consistent with the use of the term in the specification and drawings." *See* Manual of Patent Examining Procedure (MPEP) § 2111. Under this standard, a "boot processor" is a common component that was found in computer systems at the time of the alleged invention of the '302 Patent and would have been understood by a POSITA to be **any processor designated for handling bootstrap functions**. *See Wolfe Decl.* (Ex. 1003) ¶ 51. Furthermore, a boot processor can also be referred as a bootstrap processor (BSP) or by a computer-readable designator (e.g., logical CPU 0). *Id.*, ¶¶ 49-51.

As noted, a boot processor was a critical device for any computing system and was a common component included in computing systems. *Id.* Indeed, "[t]he execution of the bootstrap program is always automatically initiated when the system is powered-on or when a system hardware reset is applied. It is also initiated when the system is rebooted from the system console." *Levi* (Ex. 1013) § 4.2.1. Furthermore, "[t]he origin of the word boot (as in, 'to boot the system') is bootstrapping, which is the process of bringing a computer system to life and ready to use. ('Bootstrapping' is actually the 'nerd word' for starting up a computer.)." *Id.*

As of the priority date, a POSITA would have understood that in multi-processor system architectures, a boot processor may either be selected by default or through a selection process during bootup. *Wolfe Decl.* (Ex. 1003) ¶¶ 49-51. For example, "upon start-up of the computer system, one default boot CPU is designated to boot the system." *Kennedy* (Ex. 1006) at 2:18-20. Alternatively, as noted in the Intel Itanium Manual, which the '302 Patent incorporates by reference, "firmware selects a Bootstrap processor (BSP) in multi-processor (MP) configurations early in the boot sequence." Ex. 1015 § 24.1.1. And in some systems (e.g., fault-tolerant systems), the processor designated as the boot processor may change over time. *Wolfe Decl.* (Ex. 1003) ¶¶ 49-51. For example, the Intel MultiProcessor Specification explains how an operating system can determine and track the ID of a designated BSP:

> While all processors in an MP-compliant system are functionally identical, one of the processors will be designated as the bootstrap processor (BSP) at system initialization by the system hardware or by the system hardware in conjunction with the BIOS. The rest of the processors are designated as the application processors (APs). The BSP is responsible for booting the operating system. Once the MP operating system is up and running, the BSP functions as an AP. Usually a

> processor is designated as the BSP because it is capable of controlling all system hardware, including AP startup and shutdown. The operating system must determine and remember the APIC ID of the designated BSP, so it can keep the BSP operating as the last running processor during system shutdown. The BSP is not necessarily the first processor, especially in fault-tolerant MP systems in which any available processor can be designated as the BSP.

Ex. 1016 § B.1.

In some multi-processor system architectures, a boot processor may be referred to by a logical, rather than a physical, position (e.g., logical CPU 0 or CPU L0 rather than physical CPU 0 or CPU P0) that is indicative of its designation for controlling other system hardware since the role of boot processor may often be assigned to a processor regardless of its physical position within a system. *Wolfe Decl.* (Ex. 1003) ¶¶ 49-51. For example, during boot-processor selection,

> If CPU P0 is not functioning properly and cannot perform even the most basic functions, it is designated as inoperative. The hardware system then awakens CPU P1 and repeats the process of testing the CPU. If CPU P1 is operational, then it is designated as CPU L0, and it boots the remainder of the system. On the other hand, if CPU P1 also fails, it is also given an inoperative designation. The computer system then turns on CPU P2, and repeats the process. The process repeats until an operational microprocessor is found to perform the CPU L0 functions.

Cepulis (Ex. 1014) at 2:52-64.

The proposed interpretation of boot processor is consistent with the context provided by the '302 Patent, which references the boot process of an Itanium Processor Family (IPF) system:

> The boot-up process of a traditional IPF system, for example, proceeds as follows: when the system is first powered on, there are some sanity checks (e.g., power on self-test) that are performed by microprocessors included in the system platform, which are not the main system processors that run applications. After those checks have passed, power and clocks are given to a boot processor (which may, for example, be master processor 12A). Ex. 1001 at 11:61-12:2.

As another indicator that the '302 patent's usage of boot processor is consistent with the ordinary and customary meaning, the '302 patent discusses using the process outlined in the Intel Multiprocessor Specification to switch the role of a boot processor by "changing a system identifier, used by the system's operating system to recognize the boot processor . . . to a different value." *Id.*, claim 1.

As shown, the broadest reasonable interpretation for "boot processor" that is both "consistent with the ordinary and customary meaning of the term" and "consistent with the use of the term in the specification and drawings" is any processor designated for handling bootstrap functions including any processor that is referred to as a boot processor, that is referred to as a bootstrap processor (BSP), or that is defined by its logical designation as a boot processor (e.g., logical CPU 0 or CPU L0).

D. Claim Interpretation under 35 U.S.C. 112(f)

1. "system firmware means for detecting . . ." (Claim 26)

Claim 26 of the '302 Patent recites "system firmware means for detecting loss of lockstep (LOL) for a processor in a multi-processor system;..." Because the limitation recites a modifier ("system firmware") before "means," the limitation does not require interpretation under 112(f). In particular, "...examiners will apply 35 U.S.C. 112(f) to a claim limitation if: ... (C) the term 'means' or 'step' or the generic placeholder is not modified by sufficient structure, material, or acts for performing the claimed function." MPEP § 2181(I). In this instance, the generic placeholder term "means" is modified by the term "system firmware," which would have been recognized by a POSITA as being "sufficient structure . . . for performing the claimed function."

That said, even if the "system firmware means" requires interpretation under 112(f) and would include equivalents, Requester notes that the '302 Patent describes structure for the system firmware means in its specification. The '302 Patent states:

> Processor architecture generally comprises corresponding **supporting firmware**, such as firmware 18 of system 10. For example, as described further below in conjunction with the specific example of FIG. 2, the IPF processor architecture comprises such Supporting firmware as Processor Abstraction Layer (PAL), System Abstraction Layer (SAL), and Extended Firmware Interface (EFI). **Such supporting firmware may enable**, for example, **the OS to access a particular function implemented for the processor**... Accordingly, the Supporting firmware. Such as the PAL, is implemented to provide an **interface to the processor(s)** for accessing the functionality provided by such processor(s). Each of those interfaces provide standard, published procedure calls that are Supported. *Id.* at 8:34-59.

Accordingly, Requester proposes that the corresponding structure for performing the claimed function of detecting loss of lockstep for a processor in a multi-processor system includes supporting firmware, including a Processor Abstraction Layer, a System Abstraction Layer, and an Extended Firmware Interface, and equivalents thereof.

2. "means for determining . . ." (Claim 26)

Claim 26 of the '302 Patent recites a "means for determining whether the processor for which said LOL is detected is assigned the role of boot processor." Because the limitation recites only "means" and no other terms that may connote specific structure, 112(f) applies. *Williamson v. Citrix Online, LLC*, 792 F.3d 1339, 1348 (Fed. Cir. 2015).

The '302 Patent discloses that "[a]s described further herein, in certain embodiments, **a device tree** is stored to a data storage device (e.g., SRAM) such that it is accessible to the firmware 18, wherein the **device tree includes information identifying processor 12 as the system's boot processor** and identifying processor 15 as the spare." 'Ex. 1001 at 9:4-8. The '302 Patent further indicates that "**device tree 203 includes information designating which processor is the boot processor**, as well as a spare processor for the boot processor." *Id.* at 13:21-23. Accordingly, Requestor proposes that the corresponding structure for performing the claimed function of determining whether the processor for which said LOL is detected is assigned the role of boot processor includes a device tree, and firmware or hardware for accessing the processor designators in the device tree, and equivalents thereof.

3. "system firmware means for switching . . ." (Claim 26)

Claim 26 of the '302 Patent recites "system firmware means for switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which said LOL is detected is assigned the role of boot processor." As discussed above, because the limitation recites a modifier ("system firmware") before "means," the phrase does not require interpretation under 112(f). See, MPEP § 2181(I). In this instance, the generic placeholder term "means" is modified by the term "system firmware," which would have been recognized by a POSITA as being "sufficient structure . . . for performing the claimed function.

Notwithstanding, Requester notes that the '302 Patent clearly describes structure for the system firmware means in its specification. The '302 Patent states:

> Processor architecture generally comprises corresponding **supporting firmware**, such as firmware 18 of system 10. For example, as described further below in conjunction with the specific example of FIG. 2, the IPF processor architecture comprises such Supporting firmware as Processor Abstraction Layer (PAL), System Abstraction Layer (SAL), and Extended Firmware Interface (EFI). **Such supporting firmware may enable**, for example, **the OS to access a particular function implemented for the processor**... Accordingly, the Supporting firmware.

> Such as the PAL, is implemented to provide an **interface to the processor(s)** for accessing the functionality provided by such processor(s). Each of those interfaces provide standard, published procedure calls that are Supported. *Id.* at 8:34-59.

Id. at 8:34-59. As such, Requester proposes that the corresponding structure for performing the claimed function of switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which said LOL is detected is assigned the role of boot processor includes supporting firmware, including a Processor Abstraction Layer, a System Abstraction Layer, and an Extended Firmware Interface, or equivalents thereof.

E. List of Prior Art Patents and Printed Publication

As shown below, Requester submits that the prior art references raise a new "substantial question of patentability" because "the teaching of the (prior art) patents and printed publications is such that a reasonable examiner would consider the teaching to be important in deciding whether or not the claim is patentable." *See* MPEP § 2242.

For example, the references discussed below and in further detail in the attached charts, when considered alone or in combination, teach or suggest each limitation of the Challenged Claims, including failing over from one processor to another processor. Further, with the exception of U.S. Patent No. 7,085,959 to Safford (which is only applied to a small subset of dependent claims), the references relied upon in this Request are new; the remaining references relied upon herein were not previously considered. However, Safford is only applied to a few dependent claims, and the disclosure of Safford is not applied in the same manner as it was during prosecution. In particular, the Safford is merely relied on for the disclosure of "hot spare processors" and is not relied on as the primary basis for unpatentability. The "same question of patentability as to the claim has not been decided by the Office in an earlier concluded examination or review of the patent" at least because the same prior art combinations were not before the Office during prosecution of the '302 Patent or during a prior post-grant proceeding challenging any claim of the '302 Patent, any of which that are known to Requester have been listed below.

Because these documents present substantial new questions of patentability, reexamination should be ordered and the challenged claims should be canceled. Reexamination of the Challenged Claims is requested in view of the following references:

Exhibit 1004 (U.S. Patent Application Publication No. 2004/0123201 to Nguyen et al. ("Nguyen")): Nguyen is a prior art printed publication because it was filed as an application on December 19, 2002, which is before the earliest possible priority date of October 25, 2004.

Nguyen is analogous art because it is both in the same field (providing recovery from failure in a high-availability computing system) as the '302 Patent and reasonably pertinent to the problem (avoiding computer system crashes due to processor failures) allegedly solved by the '302 Patent. Ex. 1001 at 3:7-21. Specifically, Nguyen describes a high-reliability multi-processor system that provides recovery from failure in a high-availability computing system. *Nguyen*, (Ex. 1004) ¶¶ [0027] and [0045]. Nguyen specifically indicates that processor failover occurs in response to a loss of lockstep between processors. *Id.*

Nguyen is prior art under pre-AIA 35 U.S.C. § 102(a).

Exhibit 1005 (U.S. Patent No. 7,328,371 to Kalyanasundharam et al. ("Kalyanasundharam")): Kalyanasundharam is a prior art printed publication because it was filed as an application on October 15, 2004, which is before the earliest claimed effective filing date of October 25, 2004.

Kalyanasundharam is analogous art because it is both in the same field (providing recovery from failure in a high-availability computing system) as the '302 Patent and reasonably pertinent the problem (avoiding computer system crashes due to processor failures) allegedly solved by the '302 Patent. Ex. 1001 at 3:7-21. Specifically, Kalyanasundharam describes a redundant processing system that establishes multiple processing core pairs, at least one of which performs redundancy checks to ensure that the processor cores remain in lockstep. *Kalyanasundharam* (Ex. 1005) at 4:15-25, 4:61-65. Kalyanasundharam is from the same field of high reliability computing that provides recovery from failure in a high-availability computing system. *Id.*

Kalyanasundharam is prior art under pre-AIA 35 U.S.C. § 102(e).

Exhibit 1006 (U.S. Patent No. 5,450,576 to Kennedy et al. ("Kennedy")): Kennedy is a prior art printed publication because it was filed as an application on April 26, 1993, which is more than a year before the earliest claimed effective filing date of October 25, 2004.

Kennedy is analogous art because it is both in the same field (providing recovery from failure in a high-availability computing system) as the '302 Patent and reasonably pertinent to the problem (avoiding computer system crashes due to processor failures) allegedly solved by the '302

Patent. Ex. 1001 at 3:7-21. Specifically, Kennedy describes a high-availability system that initializes an alternative boot processor when a default boot processor fails to properly boot. *Kennedy* (Ex. 1006) at 2:29-38, 2:50-56. The Kennedy reference is in the same field of high-reliability computing that provides recovery from failure in a high-availability computing system. *Id.*

Kennedy is prior art under pre-AIA 35 U.S.C. § 102(b).

Exhibit 1007 (U.S. Patent No. 7,085,959 to Safford ("Safford")): Safford is a prior art printed publication because it was filed as an application on July 3, 2002, which is before the earliest claimed effective filing date of October 25, 2004.

Safford is analogous art because it is both in the same field (providing recovery from failure in a high-availability computing system) as the '302 Patent and reasonably pertinent the problem (avoiding computer system crashes due to processor failures) allegedly solved by the '302 Patent. Safford generally describes a multi-processor system capable of recovering from a loss of lockstep detected in connection with a certain processor pair. *Safford* (Ex. 1007), Abstract. Safford's system includes logic for changing the roles of the processors in response to a loss of lockstep. *Id.* at 3:27-30. The Safford reference is in the same field of high-reliability computing that provides recovery from failure in a high-availability computing system.

Safford is prior art under pre-AIA 35 U.S.C. § 102(a).

Exhibit 1014 (U.S. Patent No. 5,491,788 to Cepulis et al. ("Cepulis")): Cepulis is a prior art patent because it was filed as an application on September 10, 1993, and issued on February 13, 1996, which is before the earliest claimed effective filing date of October 25, 2004.

Cepulis is prior art at least under pre-AIA 35 U.S.C. § 102(a) and pre-AIA 35 U.S.C. § 102(b).

As shown below, Requester submits that these prior art references raise new "substantial [questions] of patentability" because "the [teachings] of the (prior art) patents and printed publications [are] such that a reasonable examiner would consider the [teachings] to be important in deciding whether or not the [claims are] patentable." *See* MPEP § 2242.

Of the references listed above, Safford was the sole reference cited and allegedly considered during prosecution of the '302 Patent. Safford is being presented herein "in a new light, or in a different way, as compared with its use in the earlier examination(s), in view of a material new argument or interpretation presented in [the instant] request." *See* MPEP § 2242. Thus, while

Safford may be considered "old art," reliance on these references in the instant request is proper. *See Id.*

Safford and Cepulis are relied upon herein in proposed rejections and present new, non-cumulative technological teachings that were not previously considered and discussed on the record during prosecution of the application that resulted in the '302 Patent or during prosecution of any other prior proceeding involving the '302 Patent. *See* MPEP § 2216. The instant request presents Safford in combination with Cepulis to disclose, teach, or suggest each of the limitations of the Challenged Claims. The instant request presents new, non-cumulative technological teachings of Cepulis to place Safford in "a new light or a different way that escaped review during earlier examination." *See* MPEP § 2242.

F. Ground 1: Kalyanasundharam in view of Kennedy Presents Substantial New Questions of Patentability for Claims 1, 5-8, 13, 17, 21, 22, and 25-26

1. Overview of Kalyanasundharam

Kalyanasundharam generally describes a redundant processing system that establishes multiple processing core pairs "in which at least a first processor core is redundantly executing code that a second processor core is also executing" and that "compare[s] communications from the first processor core to communications from the second processor core to **verify correct execution of the code**." *Kalyanasundharam* (Ex. 1005), Abstract.

Like in the '302 Patent, Kalyanasundharam's redundant processing system implements processors that operate in lockstep. *Id.* at 4:15-17. Kalyanasundharam indicates that ". . . the processor cores 18A-18B **may execute in lockstep (that is, may execute the same instructions on the same clock cycle)**." *Id.* Such is illustrated in FIG. 1, in which a pair of processor cores 18A and 18B are shown to operate in lockstep as controlled by node controller 20A.

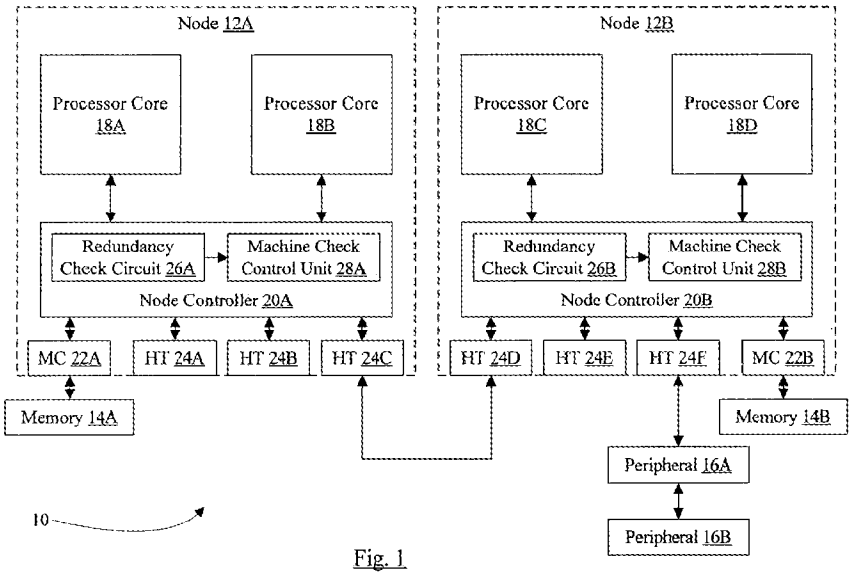


Fig. 1

Ex. 1005, FIG. 1

Furthermore, as in the '302 Patent, which designates one processor pair as master and the remaining processor pairs as slaves, Kalyanasundharam's system designates one processor pair as a "**primary processor core**" and refers to other processor cores as "**auxiliary processor cores**." *Id.* at 4:58-61. Kalyanasundharam further defines the primary and auxiliary cores, indicating that "[t]he primary processor core may be the **processor core whose execution results will be used as the result** of executing the redundantly-executed software code." *Id.* at 4:61-63. "The auxiliary processor core is used to **verify correct execution** by the primary processor core." *Id.* at 4:63-65.

Kalyanasundharam notes that problems can occur when processors lose lockstep with each other. Kalyanasundharam states that "[a]n execution error may be **any indication that an error has occurred** in the execution of the software code, **detected via the redundant execution**." *Id.* at 3:63-65. "In the embodiment of FIG. 1, it may not be clear which processor core 18A-18B experienced the error, but the error is detected." *Id.* at 3:65-67. "Diagnostic software may be used to **determine which processor core 18A-18B experienced the error** and **what corrective measures are to be taken**." *Id.* at 3:67-4:3.

Once Kalyanasundharam's system identifies which processor core experienced the error, it will disable the primary processor core and select one of the auxiliary processor cores to become the new primary processor core. *Id.* at 13:16-22. Thus, just as the '302 Patent determines whether a boot processor experienced an error and then switches the boot processor

to a spare processor, Kalyanasundharam's system will "**disable the primary processor core and select one of the auxiliary processor cores to be the primary processor core** (block 102)." *Id.* This is shown below in step 102 of Kalyanasundharam's FIG. 6. In this manner, Kalyanasundharam establishes primary and secondary processor cores and, upon detection of an error at the primary processor core, switches processing from the primary processor core to a secondary processor core. *Id.* at 4:15-25, *see also* 4:61-65.

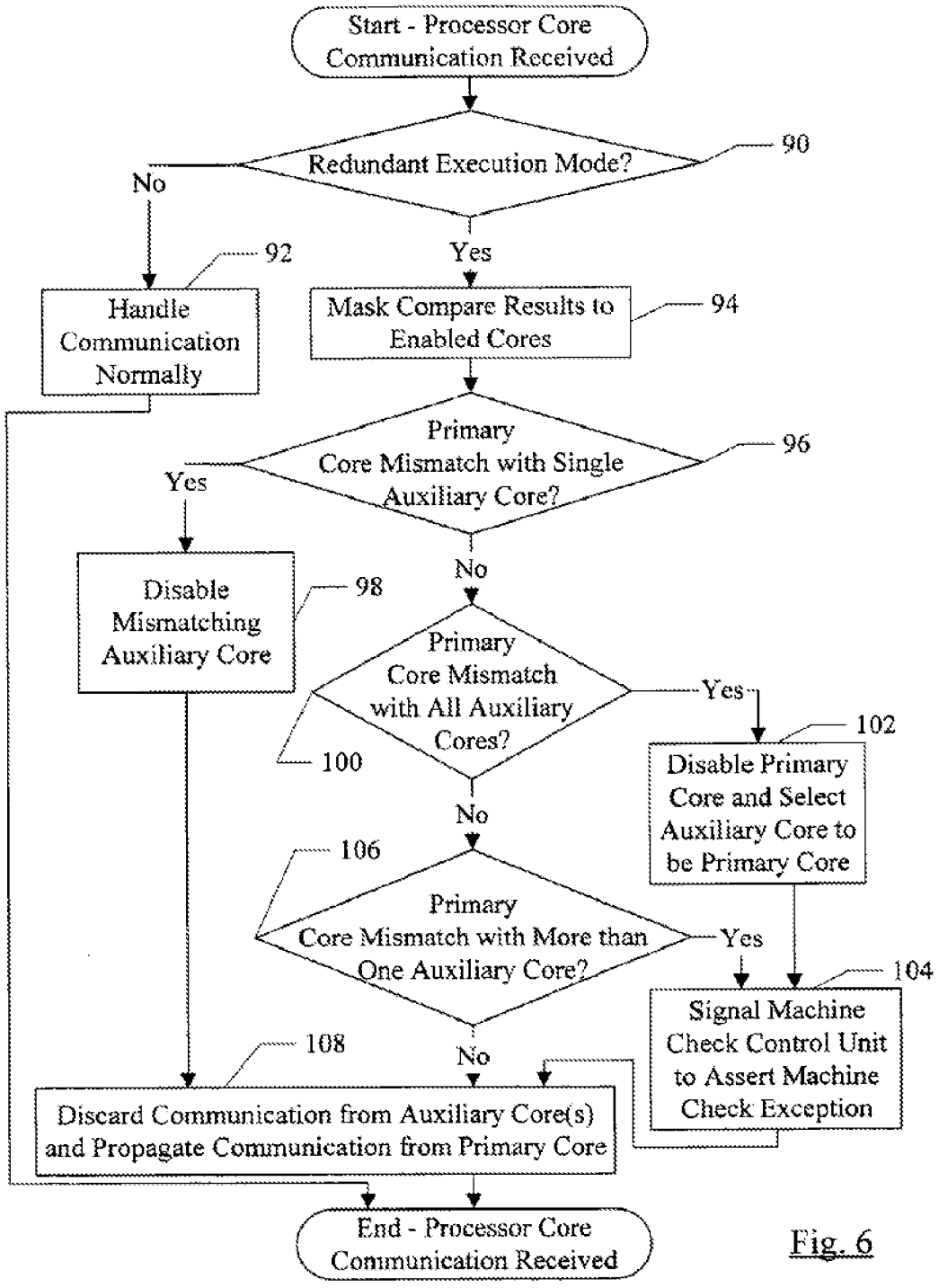


Fig. 6

Ex.1005, FIG. 6

Kalyanasundharam is analogous to the '302 Patent. For example, Kalyanasundharam is from the same field of endeavor (*e.g.*, providing recovery from failure in a high-availability computing system) and is reasonably pertinent to a particular problem faced by the inventors of the '302 Patent (*e.g.*, how to recover from processor failures in multi-processor systems). *Wolfe*

Decl. (Ex. 1003) ¶ 63, *see also '302 Patent* at 4:64-67. For example, as described in the '302 Patent, "[h]igh availability is a major desire for many customers having large multi-processor systems," but "Silent Data Corruption ('SDC') is a difficult problem . . . "for large systems having many processors because cosmic events typically occur more frequently as the processor count goes up." *'302 Patent* (Ex. 1001) at 1:55-56, 3:20-25. Similarly, Kalyanasundharam notes that errors or data mismatches may occur between primary processor cores and auxiliary processor cores and, in response, it "disable[s] the primary processor core and select[s] one of the auxiliary processor cores to be the primary core processor." *Kalyanasundharam* at 13:16-22. This allows Kalyanasundharam's system to continue processing inputs even after a noted failure. *Id.* at 13:23-27. In identifying the problem of providing highly-available multiple processor systems that are susceptible to errors and data mismatches, Kalyanasundharam is reasonably pertinent to this problem faced by the inventors of the '302 Patent. *Wolfe Decl.* (Ex. 1003) ¶ 63.

2. Overview of Kennedy

Kennedy generally describes a high-availability multi-processor computing system for "controlling initialization and self-test operations." Ex. 1006, Abstract. Like the '302 Patent's high-availability processing system, the multi-processor system in Kennedy is configured "... **to automatically transfer default system initialization operations** from a default processor to a first alternative processor **if the default processor fails**, and to **automatically transfer initialization operations to a second alternative processor if the first alternative processor fails**, and so forth depending upon how many alternative processors are installed in the multiprocessor system." *Id.*

Like the assignment of a boot processor role to a given processor in the '302 Patent, Kennedy determines which processor will boot the system and designates that processor as the "default boot processor." *Id.* at 7:7-10. Kennedy states that, "... upon start-up of the computer system, **one default boot CPU is designated to boot the system**, and **if this CPU fails, control transfers automatically to an alternate processor**." *Id.* at 2:17-21. In reference to FIG. 1 below, Kennedy notes, "provision needs to be made to **determine which processor will boot the system, and if this 'default' boot processor fails** to boot the system**, an alternative processor,** if one is installed, **should take over boot operations** to enhance the availability of the system." *Id.* at 7:5-13.

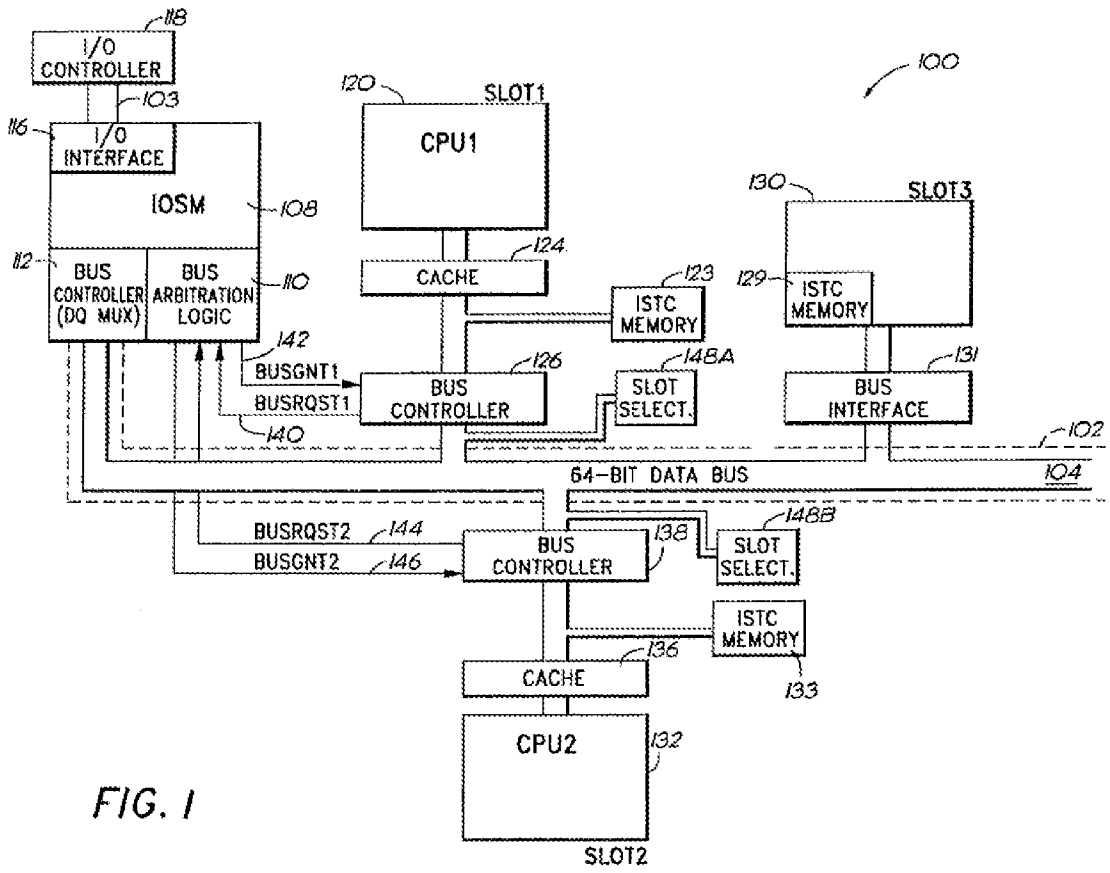


FIG. 1

Ex. 1006, FIG. 1

Just as the '302 Patent designates "boot" and "spare" processors, Kennedy similarly designates "boot" and "alternate" processors. In reference to FIG. 3 below, Kennedy states that, "In the present embodiment, **slot 1 contains the default boot CPU**." *Id.* at 10:64-65. If the default boot CPU in slot 1 fails to boot the system, an alternative CPU will take over boot operations. *Id.* at 7:8-13. Kennedy specifically explains that, "If the CPU1 120 does not complete boot operations within the time allotted, then the CPU2 132 **takes over default boot operations as an alternative boot processor instead of CPU1 120**." *Id.* at 13:38-42. Kennedy continues, "If the CPU1 120 does boot properly, then the **CPU2 132 checks in as an additional processor rather than taking over as a boot processor**." *Kennedy* (Ex. 1006) at 13:42-44. Thus, like in the '302 Patent, Kennedy designates a boot processor and, upon detecting a failure in the boot processor, switches to an alternative processor. *Id.* at 2:29-38, *see also* 2:50-56.

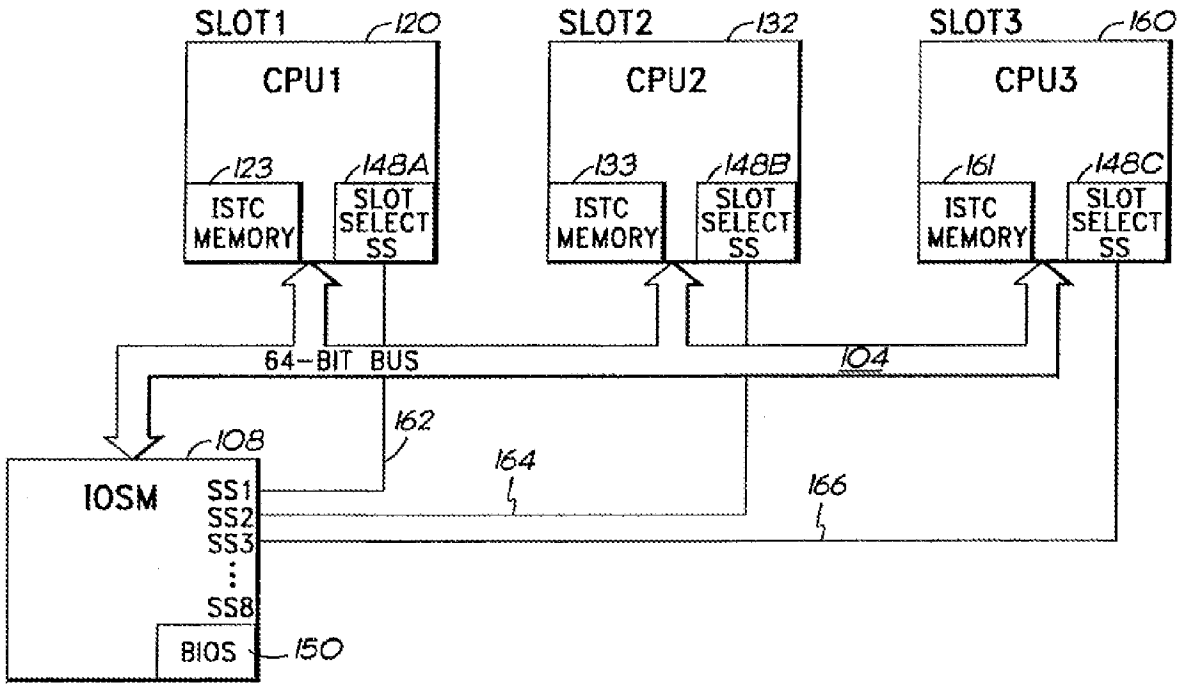


FIG. 3

Ex. 1006, FIG. 3

Kennedy is analogous to the '302 Patent; it is from the same field of endeavor (e.g., providing recovery from failure in a high-availability computing system) and is reasonably pertinent to the particular problem the '302 Patent was trying to solve (e.g., how to recover from processor failures in multi-processor systems). *Id.* at 2:15-21, *see also '302 Patent* at 4:64-67. For example, the '302 Patent is focused on the issue that, in trying to recover from failure, "in certain system architectures problems arise in attempting to idle (or eject) the boot processor from the system." *Id.* at 6:25-26. Similarly, Kennedy is directed to addressing "[a] number of problems [that] hamper the development of such a multiprocessor architecture, such as, determining which processor boots the system, and if one processor fails to boot the system, enabling another alternative processor to take over boot operations." *Kennedy* (Ex. 1006) 1:60-64. Thus, Kennedy is reasonably (directly) related to a problem the inventors of the '302 Patent faced: recovering from failure of a boot processor. *Wolfe Decl.* (Ex. 1003) ¶ 64.

3. Motivation to Combine Kalyanasundharam and Kennedy

As explained in detail in Appendix A, the subject matter of the claims for which reexamination is requested would have been obvious based on the combination of Kalyanasundharam in view of Kennedy. *Wolfe Decl.* (Ex. 1003) ¶ 63. Kalyanasundharam generally describes a process of recovering from an error by disabling a failed primary processor core and selecting an auxiliary processor core to function as the new primary processor core. *Kalyanasundharam* (Ex. 1005) at 13:16-23. Kalyanasundharam further expressly indicates that the error can be a mismatch between processor outputs or loss of lockstep between processor cores. *Id.* at 10:26-42. While Kalyanasundharam describes the process of failing over from a primary processor core to an auxiliary processor core in response to such a mismatch, it fails to explicitly mention a "boot processor" as recited in the '302 Patent. Notably, Kalyanasundharam does not limit its "primary" or "secondary" processor cores to any specifically defined function. That is, the disclosure of Kalyanasundharam, while not explicitly disclosing a "boot processor," does not teach or suggest that its processor cores cannot be "boot processors."

Kennedy, in an analogous reference, explicitly describes the process of failing over from a failed boot processor to an alternative processor that takes over the functions of the failed boot processor. *Kennedy* (Ex.1006) at 13:38-44. A POSITA would have been motivated to combine Kennedy with Kalyanasundharam to provide the added reliability of boot processor failover to Kalyanasundharam's described methods of processor failover. *Wolfe Decl.* (Ex. 1003) ¶ 65. Specifically, with regard to Kalyanasundharam's system, which fails over from a primary processor core to a secondary processor core in response to a loss of lockstep, a POSITA would have been motivated to add the boot processor and spare processor described by Kennedy to extend its failover functionality to a broader number of processor types. This would provide loss of lockstep failover not just between primary and secondary processor cores, but also between boot and spare processors. Because the Kalyanasundharam and Kennedy systems both describe a similar computing architecture in which processors fail over to backup processors upon the occurrence of a failure, the POSITA would reasonably expect to succeed when combining these teachings. This combination of Kalyanasundharam with Kennedy would have rendered the claims of the '302 Patent obvious, as is explained in further detail below with regard to multiple KSR rationales that each support a conclusion of obviousness. *Wolfe Decl.* (Ex. 1003) ¶¶ 65, 67.

"Combining prior art elements according to known methods to yield predictable results" MPEP § 2143.I(A)

A POSITA would have been motivated to combine known methods of processor failover, as described in Kalyanasundharam, with the boot-specific processor failover methods described in Kennedy. *Wolfe Decl.* (Ex. 1003) ¶ 65. Even when combining the processor failover elements of Kalyanasundharam with the boot-processor-specific failover elements of Kennedy, each element would perform the same function, and the resulting combination of elements would provide a predictable outcome. *Id.*

For example, when Kalyanasundharam's system detects an error or data mismatch between a primary processor core and an auxiliary processor core, it "disable[s] the primary processor core and select[s] one of the auxiliary processor cores to be the primary core processor." *Kalyanasundharam* (Ex. 1005) at 13:16-22. This allows Kalyanasundharam's system to continue processing inputs after a noted failure. *Id.* at 13:23-27. Similarly, Kennedy states that if a default boot processor fails, "control transfers automatically to an alternate processor." *Kennedy* at 2:18-20. Kennedy follows by stating that "This provides fault tolerance and high availability for the system." *Id.* 2:21-22.

A POSITA would have been motivated to combine the known prior art elements of processor failover from a primary processor core to an auxiliary processor core, as taught by Kalyanasundharam, with the teachings of boot processor failover, as described by Kennedy, to increase the reliability and availability of the underlying system. *Wolfe Decl.* These processes, when combined, would create the predictable outcome of a highly reliable system that would continue to process input data or would continue to perform boot processes even after a failure in one processor has occurred. *Wolfe Decl.* (Ex. 1003) ¶ 67.

In both Kalyanasundharam and Kennedy, the known method of carrying out the changeover from a failed processor to an alternative processor was to alter the identifier of the processor. *Wolfe Decl.* (Ex. 1003) ¶ 68. For instance, in Kalyanasundharam, an OE field of a processor's configuration register may be changed from "primary processor core" to an "auxiliary processor core" or, in Kennedy, a SLOT_ID field for a processor may be changed from "boot processor" to "alternative processor." *Kalyanasundharam* (Ex. 1005) at 14:48-64, and *Kennedy* (Ex. 1006) at 13:32-44. Providing processor failover by changing a processor's identifier was a known process where each element performs the same function as it did separately. As such, a

POSITA would be motivated to add the functionality of boot processor failover by changing slot identifiers, as taught by Kennedy, to the known method of changing configuration register fields, as taught by Kalyanasundharam. *Id.* Moreover, a POSITA would have recognized that the results of this combination are predictable, providing an outcome where an alternative processor performs the functions of a failed processor. *Wolfe Decl.* (Ex. 1003) ¶ 68.

"Some teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention." MPEP § 2143.I(G)

Kalyanasundharam's multiprocessor system is designed for high availability. *Kalyanasundharam* (Ex. 1005) at 1:8-11. Kalyanasundharam indicates that "many servers are implemented on fault-tolerant systems so that the servers may be available as close to 24 hours a day as possible." *Id.* at 1:14-17. Kalyanasundharam further states that, "Any mission-critical computing functions may benefit from being assigned to fault-tolerant computer systems." *Id.* at 17-19. With reference to mission-critical computing functions, Kennedy states that, "In a multiprocessor environment, one critical determination is which processor will boot the system." *Kennedy* (Ex. 1006) at 10:2-3. Kennedy further states that, "…if one processor fails, provision for disabling the non-functional processor and allowing another alternative processor to boot the system enhances the availability of the system." As such, a POSITA would be aware that booting a computer system qualifies as a "mission-critical" computing function that, according to Kalyanasundharam, would benefit from being assigned to a fault-tolerant computer system. *Kalyanasundharam* (Ex. 1005) at 1:17-19. *Wolfe Decl.* (Ex. 1003) ¶ 69.

A POSITA would further be motivated to enhance the availability of Kalyanasundharam's system by specifically noting that failover can occur between boot processors, as taught by Kennedy, and as claimed in the '302 Patent. *Wolfe Decl.* (Ex. 1003) ¶ 70. Combining teachings of Kalyanasundharam and Kennedy would lead a POSITA to arrive at the claimed invention of the '302 Patent, namely a system that, in response to detecting an error (e.g., a loss of lockstep (LOL)), determines whether the processor that experienced the error (e.g., the LOL) is a boot processor and, if so, causes the system to recognize another processor as the boot processor (see, for example, claim 21 in the '302 Patent). Kennedy recognizes a failed processor as a boot processor and switches to an alternative processor, while Kalyanasundharam recognizes a failed primary processor and switches to an auxiliary processor in response to a loss of lockstep. *Kennedy* (Ex.

1006) at 7:7-13, and Kalyanasundharam 13:16-22. A POSITA would have reasonably expected that this combination would enhance the availability of Kalyanasundharam's system by incorporating the boot processor failover method of Kennedy. *Kalyanasundharam* (Ex. 1005) at 1:20-25. *See also Wolfe Decl.* (Ex. 1003) ¶¶ 69, 70. Indeed, both Kalyanasundharam's and Kennedy's systems describe a similar computing architecture in which one set of processors fails over to another set of processors upon the occurrence of a failure. As such, the POSITA would reasonably expect to succeed when combining these teachings.

G. Ground 2: Kennedy in view of Nguyen and Safford Presents Substantial New Questions of Patentability for Claims 1-27

1. Overview of Kennedy

A discussion of Kennedy appears above. *Supra* § F.2. That discussion is equally applicable to SNQ2 and will not be repeated here for the sake of brevity.

2. Overview of Nguyen

Nguyen generally describes a processor that includes first and second execution cores that operate in functional redundancy checking (FRC) mode. *Nguyen* (Ex. 1004), Abstract. As explained in Nguyen, a "single processor enabled for FRC may include replicated instruction execution cores on which the same instruction code is run." *Id.* ¶ [0006]. "In FRC mode, one of execution cores 120 may be designated as the master. The master refers to the execution core that is responsible for updating the resources shared by execution cores 120." *Id.* ¶ [0027]. "The other execution core 120 may be designated as the slave. The slave is responsible for generating results from the same code sequence to be checked against those of the master." *Id.*

Nguyen's processor implements multiple "[e]xecution cores 120 [that] operate in lock step during normal FRC mode[.]" *Id.* ¶ [0024]. The configuration of Nguyen's processor, whose execution cores 120 operate in lockstep, is illustrated in FIG. 1, and is reproduced below for convenience.

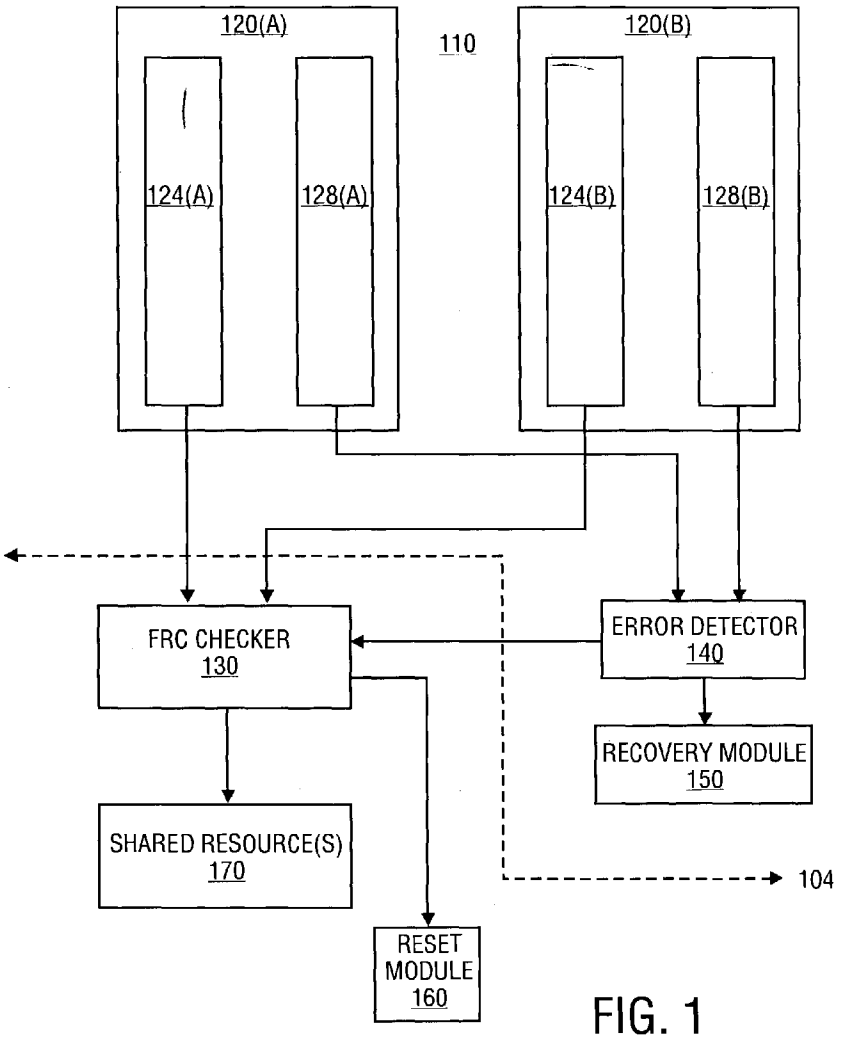


FIG. 1

Ex. 1004, FIG. 1

Also like in the '302 Patent, Nguyen's processor includes data storage for storing information that assigns different roles to the execution cores. For example, Nguyen's processor maintains "a bit in a status register of each execution core 120 . . . to indicate its status as master or slave. The bit may be set, for example, when the system is booted or reset." *Id.* ¶ [0047]. "[T]he master/slave status of an execution may be changed dynamically to allow recovery for an error in either core." Id.

Other processor status indicators indicate that a specific processor is in charge of booting the underlying computer system. Indeed, as will be recognized by a POSITA, every computing system implicitly requires at least one processor to boot the computer system during startup. Wolfe

Decl. (Ex. 1003) ¶¶ 49-51. Or, at a minimum, a POSITA would recognize that it was very common for computing systems to have at least one boot processor to perform boot operations at startup. *Id*. This process, known as "bootstrapping," as explained above, requires that at least one processor is designated as the boot processor and, in this role, performs all appurtenant boot-related tasks. Supra § I.B.1. Notably, Nguyen does not limit its processors to any specifically defined function. That is, the disclosure of Nguyen, while not explicitly disclosing a "boot processor," does not teach or suggest that its processors cannot be a "boot processor."

Further like in the '302 Patent, Nguyen's processor includes logic for detecting a loss of lockstep, specifically at a master core. Nguyen notes that "for FRC-enabled processors, such errors are likely to be manifested as FRC errors, since they take the execution cores out of lock-step." Id. ¶ [0008]. In cases where Nguyen's system detects a loss of lockstep (an FRC error), "the system goes through a reset operation rather than a shorter recover operation." *Nguyen* (Ex. 1004), ¶ [0040]. Nguyen is analogous art to the '302 Patent. For example, Nguyen is from the same field of endeavor (e.g., providing recovery from failure in a high-availability computing system) and is reasonably pertinent to a problem faced by the inventors of the '302 Patent (e.g., how to recover from processor failures in multi-processor systems). Wolfe Decl. (Ex. 1003) ¶ 101, see also '302 Patent at 4:64-67.

For example, as described in the '302 Patent, "[h]igh availability is a major desire for many customers having large multi-processor systems," but "Silent Data Corruption ('SDC') is a difficult problem . . . "for large systems having many processors because cosmic events typically occur more frequently as the processor count goes up." '302 Patent (Ex. 1001) at 1:55-56, 3:20-25. Similarly, Nguyen notes that "[s]oft errors pose a major challenge to both" the design goals of "high levels of reliability and availability" and even identifies the problematic soft errors caused by cosmic events such as "collisions between high-energy particles, e.g. alpha particles, and charge storing nodes." Nguyen (Ex. 1004) ¶ [0004]. In identifying the problem of providing highly-available multiple processor systems that are susceptible to soft errors caused by cosmic events, Nguyen is reasonably (directly) pertinent to this same problem identified by the inventors of the '302 Patent.

3. Overview of Safford

Safford generally describes a multi-processor system capable of recovering from a loss of lockstep detected in connection with a certain processor pair. Ex. 1007, Abstract. As explained in Safford, "to improve reliability of processing assets, a computer system employs lock stepped processor cores that operate in a master/check pair." *Id.* at 2:55-57. "Each of two processors in the pair processes the same code sequences, and the resulting outputs of the processors are compared by a logic circuit located near external interfaces of the two processors." *Id.* at 2:57-60. "Any difference in the processor outputs indicates the existence of an error." *Id.* at 2:60-63. The configuration of Safford's system, whose processors operate in lockstep, is illustrated in FIG. 2 and is reproduced below for convenience.

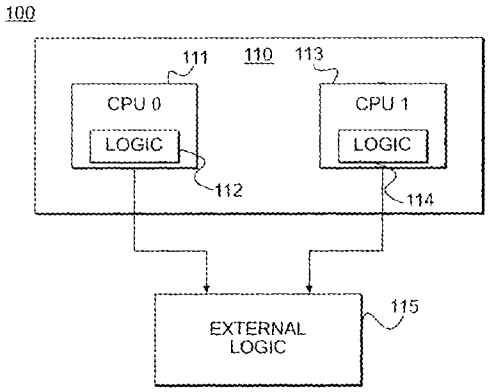


FIG. 2

Ex. 1007, FIG. 2

Like in the '302 Patent, Safford's system includes data storage for storing information that assigns different roles to the processors. For example, in Safford's system, "the architected states of each of the microprocessors 22 and 24 is copied from the memory 25 into the microprocessors 22 and 24, respectively." *Id.* at 2:2-4. "[T]he node controller 130 may store the architected state of the processors 111, 113, 121, 123, 125, and 127, either internally in the node controller 130, or in another component of the computer system 100." *Id.* at 4:6-10. "The node controller 130 has available at all time [*sic*] a current architected state of the processors to which the node controller 130 is coupled." *Id.*

Further like in the '302 Patent, Safford's system includes logic for changing the roles of the processors in response to a loss of lockstep. For example, "[u]sing the impending loss of lock

step signal, the computer system 100 may continue operating (processing) using one of the processors 111 and 113 that did not experience an error." *Id.*, 3:27-30. In addition, Safford's system recovers from the loss of lockstep by transferring the role of the failed processor pair to a hot standby processor pair. For example, Safford's "processor 111 signals the node controller 130 that the first processor pair 111/113 has broken lock step and that the first processor pair 111/113 should be taken 'off-line.'" *Id.* at 4:37-40. Then Safford's "node controller 130 copies the architected state of the first processor pair 111/113 to the hot standby processor pair 125/127." *Id.* at 4:40-43. After the architected state is copied to the hot standby processor pair 125/127, "[t]he processor pair 125/127 . . . becomes the logical CPU 0 in the computer system 100." *Id.* at 4:52-54.

To change the roles of the processors in this way, Safford's system follows the process illustrated in FIG. 4, which is reproduced below for convenience.

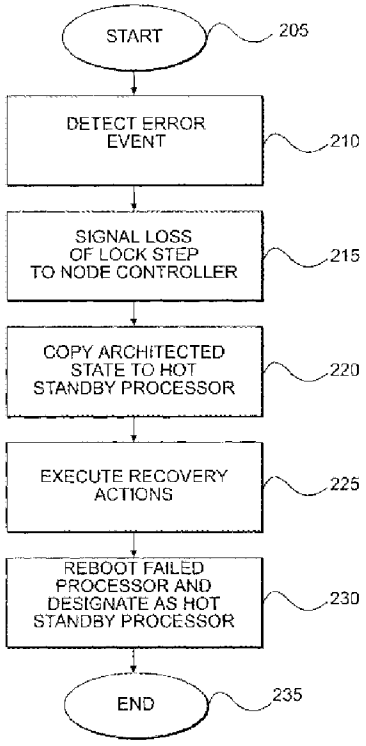


FIG. 4

Ex. 1007, FIG. 4

Safford is analogous art to the '302 Patent because Safford is both from the same field of endeavor (e.g., providing recovery from failure in a high-availability computing system) as the '302 Patent and is reasonably pertinent to at least one problem faced by the inventors of the '302 Patent (e.g., system crashes caused by loss of lockstep). *Madisetti Decl.* (Ex. 1003) ¶ 103. For example, the '302 Patent, in focusing on "the increased desire for many systems to maintain high availability" notes the problem that "crashing the system each time LOL is detected is not an attractive proposition." Ex. 1001 at 3:16-18. Similarly, Safford in focusing on "improv[ing] availability of the processor assets of the computer system," notes that "loss of lock step, if not promptly corrected, may cause the computer system 18 to crash." *Id.* at 3:21-22 and 1:45-47.

4. Motivation to Combine Kennedy, Nguyen, and Safford

As mentioned above, Kennedy describes failing over a boot processor to an alternate boot processor in response to a failure of the boot processor. See, e.g., *Kennedy* (Ex. 1006) at 7:5-13. For example, Kennedy states that "one default boot CPU is designated to boot the system, and if this CPU fails, control transfers automatically to an alternate processor." *Id.* at 2:17-21. To the extent that Kennedy does not explicitly describe failing over the boot processor **in response to a loss of lockstep** for the boot processor, Kennedy as modified by Nguyen would have rendered failover of a boot processor obvious. *See, e.g., Nguyen* (Ex. 1008), ¶¶ [0008] and [0054]-[0055].

For example, Nguyen describes detecting a loss of lockstep within a processor. *Id.* Nguyen describes a processor with multiple "[e]xecution cores 120 [that] operate in lock step during normal FRC mode." *Id.* ¶ [0024]. "In FRC mode, one of execution cores 120 may be designated as the master." *Id.* ¶ [0027]. "The master refers to the execution core that is responsible for updating the resources shared by execution cores 120." *Id.* "The other execution core 120 may be designated as the slave." *Id.* "The slave is responsible for generating results from the same code sequence to be checked against those of the master." *Id.* If an FRC error occurs at the processor, the master and slave cores have lost lock step and a recovery process in instigated. *Id.* ¶ [0024].

A POSITA would have been further motivated to combine the teachings of Kennedy, Nguyen and Safford to improve the reliability of Kennedy's system (e.g., by failing over the boot processor to an alternative processor in response to a loss of lockstep). *Wolfe Decl.* (Ex. 1003) ¶ 106. For example, a POSITA would have been motivated to modify Kennedy's system, which implements boot processor failover in response to an error, by Nguyen's teachings about detection

of an error, i.e., a loss of lockstep. *See Id.* A POSITA would have been motivated to do so because a POSITA would have recognized the obvious benefits and advantages of enabling Kennedy's system to dynamically fail over boot processors in response to a loss of lockstep, especially in view of Kennedy's express objective of "provid[ing] fault tolerance and high availability." *Kennedy* (Ex. 1006) at 2:21-22; *see also Wolfe Decl.* (Ex. 1003) ¶ 106.

To the extent that Kennedy and Nguyen are not interpreted as changing a system identifier for a boot processor upon such a loss of lockstep, Kennedy and Nguyen, as modified by Safford, would have rendered such changing of a system identifier obvious. For instance, Safford's system implements multiple processors operating in a lock step mode. *Safford* (Ex. 1007) at 2:55-60. One pair of these processors "operating in a lock step mode" initially "appear[s] to the computer system…to be…a **logical CPU 0.**" *Id.* at 3:4-6. A POSITA would recognize that the "logical CPU 0" designation is equivalent to a system identifier used by the system to determine which processor is boot processor. *Id.*, s*ee also Wolfe Decl.* (Ex. 1003) ¶ 107. Later, that **logical CPU 0** designation is reassigned to another processor during failover. *See Safford* (Ex. 1007) at 3:4-5, 4:52-53. A new processor pair 125/127 then becomes the **logical CPU 0** in the computer system 100." *Id.* As such, the system identifier "logical CPU 0" is reassigned to a working processor during failover. *Id.*

A POSITA would have recognized Safford's reassignment of the logical CPU 0 designation from one processor to another as being a change in system identifier used by the operating system to identify the boot processor. *Wolfe Decl.* (Ex. 1003) ¶ 108. The POSITA would have been motivated to combine Safford's change of system identifier with Kennedy's change of processor during failover (and specifically during a loss of lockstep, as taught by Nguyen) to further increase the likelihood of a successful failover between processors. *Wolfe Decl.* (Ex. 1003) ¶ 108. Moreover, a POSITA would have been motivated to do so because Safford's changed system identifier would enable Kennedy's system to seamlessly transfer executing processes from failing boot processor to an alternative processor without affecting the operating system (OS), the applications running on Nguyen's system, or the high availability of Kennedy's system. *Id.* Indeed, a POSITA would have recognized the obvious benefits and advantages of providing Kennedy's system with a changeable system identifier, especially in view of Safford's suggestion that, upon processor failure, the failed processors are taken offline, and a new processor pair "becomes the

logical CPU 0 in the computer system 100." *Safford* (Ex. 1007) at 4:38-53; *see also Wolfe Decl.* (Ex. 1003) ¶ 108.

A POSITA would have had a reasonable expectation of success in combining the teachings of Kennedy, Nguyen, and Safford. *Wolfe Decl.* (Ex. 1003) ¶ 109. For example, Kennedy describes failing over a boot processor to an alternate boot processor. *See, e.g., Kennedy* (Ex. 1006) at 7:5-13. Even if Kennedy does not explicitly describe detecting a loss of lockstep, Kennedy as modified by Nguyen would have rendered such a technique obvious. *See, e.g., Nguyen* (Ex. 1004), ¶¶ [0008] and [0054]-[0055]. For example, Nguyen describes detecting a functional redundancy checking (FRC) error (*e.g.,* loss of lockstep) in the master execution core. *See Id.* As mentioned above, the '302 Patent describes such cores as each being an example of a processor. *See, e.g., '302 Patent* (Ex. 1001) at 6:45-47. Because Kennedy's teachings are directed to failing over the boot processor, because Nguyen describes a loss of lockstep as triggering such processor failover, and because Safford describes changing a system identifier during failover, the proposed combination of Kennedy, Nguyen, and Safford would have been expected to succeed. *See, e.g., Wolfe Decl.* (Ex. 1003) ¶ 109. It was well within the skill of a POSITA to modify Kennedy's system by Nguyen and Safford in the manner proposed, and Kennedy did not teach away from or discourage applying such a modification. *See Id.* More specifically, Kennedy does not limit or otherwise discourage application of its techniques from being triggered by a loss of lockstep.

Moreover, even if neither Kennedy nor Nguyen explicitly changed a system identifier upon failover, Kennedy as modified by Nguyen and Safford would have rendered obvious such an identifier change. *See, e.g., Safford* (Ex. 1007) at 4:38-53. It was well within the skill of a POSITA to further modify Kennedy's system by Safford in the manner proposed, and Kennedy did not teach away from or discourage applying such a further modification. *See, e.g., Wolfe Decl.* (Ex. 1003) ¶ 110. That is, Kennedy does not limit or otherwise discourage application of its techniques to implementations that may implement such a change in system identifier during failover. Accordingly, the proposed combination of Kennedy, Nguyen, and Safford—in which Kennedy's system is modified by Nguyen and Safford to achieve dynamically replacing a failed boot processor with an alternative processor in response to a loss of lockstep while, in the process, changing a system identifier—would have been reasonably expected to succeed. *Id.* Consistent

with the Supreme Court's holding in *KSR*, the rationales provided below further support this conclusion of obviousness.

"Combining Prior-Art Elements According to Known Methods to Yield Predictable Results" MPEP § 2143.I(A)

A POSITA would have combined the teachings of Kennedy, Nguyen, and Safford according to known methods to yield predictable results. *Wolfe Decl.* (Ex. 1003) ¶ 111. Combining Kennedy, Nguyen, and Safford would yield predictable results because, pursuant to MPEP § 2143, (1) together, Kennedy, Nguyen, and Safford disclose each element of claims 1-27, (2) a POSITA could have combined these references by known methods, (3) these elements perform the same function in isolation and in the combination, and (4) the POSITA would have recognized the results of the proposed combination as being predictable. As shown below, the proposed combination of Kennedy, Nguyen, and Safford satisfies all the prongs of this rationale.

First, the combination of Kennedy, Nguyen, and Safford discloses each element recited in the applicable claims. (See Claim Chart in Appendix B).

Second, a POSITA could have combined the prior-art elements by known methods. *Wolfe Decl.* (Ex. 1003) ¶ 113. For example, Kennedy described failing over boot processors to spare processors when an error is detected. *See, e.g., Kennedy* (Ex. 1006) at 1:16-19, 2:15-22, 7:5-13; *see also Wolfe Decl.* (Ex. 1003) ¶ 113. Nguyen described a system that determines whether an error has occurred such as a loss of lockstep between processor cores (i.e., that a processor is out of lockstep). *See, e.g., Nguyen* (Ex. 1004) ¶¶ [0008], [0054]-[0055]. The POSITA would have combined Kennedy's well known boot processor failover method with Nguyen's system of detecting a loss of lockstep to fail over from a processor having the designation "boot processor" to "alternative processor", as taught by Kennedy." *Kennedy* (Ex. 1006) at 2:15-22, 7:5-13; *Wolfe Decl.* (Ex. 1003) ¶ 113. Still further, Safford described changing a system identifier so that a spare processor receives the designation of logical CPU 0 upon failure of the previous CPU 0. *See, e.g., Safford* (Ex. 1007) at 4:38-53. A POSITA would have been motivated to combined Kennedy's modified system with Safford's change of system identifier by the methods disclosed in Safford to perform the boot processor failover without affecting Kennedy's OS or corresponding applications. *Id.*, *see also Wolfe Decl.* (Ex. 1003) ¶ 113.

Third, each prior-art element performs the same function in isolation and in the combination. *Wolfe Decl.* (Ex. 1003) ¶ 114. For example, in isolation and in the combination, Kennedy's system automatically transfers the role of boot processor to a spare processor in response to a detected failure, Nguyen's system detects a loss of lockstep, and Safford's system changes a system identifier. *Id.* Accordingly, the functions performed by the prior-art elements do not change from the individual Kennedy, Nguyen, and Safford references to the proposed combination.

Fourth, the POSITA would have recognized the results of the proposed combination as being predictable. *Wolfe Decl.* (Ex. 1003) ¶ 115. For example, the POSITA would have recognized that, by failing over Kennedy's boot processor to an alternate processor in response to Nguyen's detection of a loss of lockstep while implementing Safford's change in system identifier, Kennedy's modified system would have been able to predictably fail over the boot processor to an alternate processor in response to a loss of lockstep. *Id.*

"Applying a Known Technique to a Known Device Ready for Improvement" MPEP § 2143.I(D)

A known technique could have been applied to a known device ready for improvement to yield predictable results. *Wolfe Decl.* (Ex. 1003) ¶ 116. Combining Kennedy, Nguyen, and Safford would apply a known technique to a known device because, pursuant to MPEP § 2143, (1) Kennedy disclosed a base device upon which the claimed invention can be seen as an improvement, (2) Nguyen disclosed a known technique that is applicable to the base device, (3) Safford disclosed a known technique that is applicable to the base device, (4) a POSITA would have recognized that applying the known technique would have yielded predictable results, and (5) the predictable results would have improved the base device. As shown below, the proposed combination of Kennedy, Nguyen, and Safford satisfies all the prongs of this rationale.

First, Kennedy disclosed a base device upon which the claimed invention can be seen as an improvement. *Wolfe Decl.* (Ex. 1003) ¶ 117. For example, Kennedy disclosed a device that replaces the boot processor to an alternate processor in response to a failure of the boot processor. *See, e.g., Kennedy* (Ex. 1006) at 7:5-13. Similarly, the claimed invention transfers the role of boot processor to a spare processor in response to a failure such as a loss of lockstep. *'302 Patent* (Ex. 1001), claim 1. Accordingly, an improvement appears to be that the claimed invention fails over

from the boot processor to the spare processor in response to a loss of lockstep. *Wolfe Decl.* (Ex. 1003) ¶ 117.

Second, the prior art also disclosed known techniques that are applicable to the base device. *Wolfe Decl.* (Ex. 1003) ¶ 118. For example, Nguyen described a known technique for detecting a functional redundancy checking (FRC) error (*e.g.,* loss of lockstep) in the master execution core. *See, e.g., Nguyen* (Ex. 1004), ¶¶ [0008] and [0054]-[0055]; *see also Wolfe Decl.* (Ex. 1003) ¶ 118. Safford described a known technique of changing a system identifier in response to a processor failure. *Safford*, (Ex. 1007) at 4:38-53. Because Nguyen's known technique involves detecting a loss of lockstep, because Kennedy involves replacing a failed processor with an alternative processor, and because Safford's known technique involves changing a system identifier during failover, Nguyen's and Safford's known techniques would have been applicable to Kennedy's device, which is very similar to and includes the same components and functionality as Kennedy's device. *See, e.g., Wolfe Decl.* (Ex. 1003) ¶ 118.

Third, a POSITA would have recognized that applying the known techniques to the base device would have yielded predictable results. *Wolfe Decl.* (Ex. 1003) ¶ 119. For example, the POSITA would have recognized that applying Nguyen's and Safford's known techniques to Kennedy's device would have yielded predictable results. *Id.* More specifically, the POSITA would have recognized that, by applying Nguyen's and Safford's known techniques, Kennedy's device would have been able to predictably fail over the boot processor to the alternative processor by changing a system identifier of a failed processor in response to a loss of lockstep. *Id.*

Fourth, the predictable results would have improved the base device. *Id.* For example, the predictable results would have improved the reliability of Kennedy's device (*e.g.,* by failing over the boot processor to an alternative processor by changing a system identifier in response to loss of lockstep). *Id.* More specifically, Kennedy's device would become more reliable by failing over the boot processor to an alternative processor such that the alternative processor immediately assumes responsibility for booting the underlying system in case a reboot is needed. *Id.* By doing so, Kennedy's device as modified by Nguyen and Safford would avoid a situation in which the underlying system attempts to reboot without a designated, functional boot processor (i.e., a logical CPU 0) after a loss of lockstep. *Wolfe Decl.* (Ex. 1003) ¶ 120.

"Some Teaching, Suggestion, or Motivation in the Prior Art Would Have Led a POSITA to Combine Prior-Art Teachings to Arrive at the Claimed Invention" MPEP § 2143.I(G)

The teachings of Kennedy, Nguyen, and Safford would have led a POSITA to arrive at the claimed invention because, pursuant to MPEP § 2143, (1) there was some teaching, suggestion, or motivation either in the references themselves or in the knowledge generally available to one of ordinary skill in the art to combine the references' teachings and (2) there was a reasonable expectation of success for the combination. *Wolfe Decl.* (Ex. 1003) ¶ 121. As shown below, the proposed combination of Kennedy, Nguyen, and Safford satisfies all the prongs of this rationale.

First, Nguyen emphasized and/or reiterated certain teachings, suggestions, and/or motivations that would have led a POSITA to combine the teachings of Kennedy, Nguyen, and Safford. *See, e.g., Nguyen* (Ex. 1007) ¶ [0006], [0008], [0054]; *see also Wolfe Decl.* (Ex. 1003) ¶ 122. For example, Nguyen taught that "[e]rrors that are not detected at the FRC boundary can lead to [Silent Data Corruption] SDC." *Id.* ¶ [0006]. Further, Nguyen taught that, "for FRC-enabled processors, such errors are likely to be manifested as FRC errors, since they take the execution cores out of lock-step." *Id.* ¶ [0008]. Nguyen also taught that soft errors, which pose a major challenge to both high reliability and high availability, "result from collisions between high-energy particles, e.g. alpha particles, and charge storing nodes." *Id.* ¶ [0004]. Any or all of these teachings, suggestions, and/or motivations from Nguyen would have led a POSITA to combine the teachings of Kennedy, Nguyen, and Safford. *Wolfe Decl.* (Ex. 1003) ¶ 122.

Still further, Safford emphasized and/or reiterated certain teachings, suggestions, and/or motivations that would have led a POSITA to combine the teachings of Kennedy, Nguyen, and Safford. *See, e.g., Safford* (Ex. 1006) at 2:55-63; *see also Wolfe Decl.* (Ex. 1003) ¶ 123. For example, Safford taught that its technique of changing system identifiers would ensure that at least one "good" processor would continue processing, while the "bad" processor would be turned off and could be recovered at a later time. *Id.* at 3:43-50. To ensure that the bad processor is no longer implemented by the system, Safford changes the "logical CPU 0" designation from the failed processor to another processor (e.g., from failed processors 111/113 to spare processors 125/127). *Id.* at 4:47-57. Changing the system identifier of a processor would lend further stability to the system, thereby increasing its reliability and high availability. *Wolfe Decl.* (Ex. 1003) ¶ 123. Any

or all of these teachings, suggestions, and/or motivations from Safford would have led a POSITA to combine the teachings of Kennedy and Safford. *Id.*

Even disregarding the express teachings, suggestions, and/or motivations cited from Nguyen, the POSITA would have been motivated to combine Kennedy, Nguyen, and Safford based on their own knowledge. *See, e.g.,* MPEP § 2143.I(G); *see also DyStar Textilfarben GmbH & Co. Deutschland KG v. C.H. Patrick Co.*, 464 F.3d 1356, 1360 (Fed. Cir. 2006). "Because the desire to enhance commercial opportunities by improving a product or process is universal—and even common-sensical—we have held that there exists in these situations a motivation to combine prior art references even absent any hint of suggestion in the references themselves." *Id.* "In such situations, the proper question is whether the ordinary artisan possesses knowledge and skills rendering him *capable* of combining the prior art references." *Id.* The POSITA would have possessed the knowledge and skills to combine the teachings of Kennedy, Nguyen, and Safford in this way. *See, e.g., Wolfe Decl.* (Ex. 1003) ¶ 124. Still further, the POSITA would have been motivated to combine Kennedy with Nguyen and Safford to increase the reliability of its high-reliability system by failing over from a boot processor to an alternative processor while changing the system identifier of the failed processor in response to a loss of lockstep.

Second, there was a reasonable expectation of success for the proposed combination of Kennedy, Nguyen, and Safford. *Wolfe Decl.* (Ex. 1003) ¶ 125. For example, Kennedy described failing over a boot processor to an alternate boot processor in response to a failure of the boot processor. *See, e.g., Kennedy* (Ex. 1005) at 7:5-13. Even if Kennedy did not explicitly describe failing over the boot processor in response to a loss of lockstep for the boot processor, Kennedy as modified by Nguyen did. *See, e.g., Nguyen* (Ex. 1004), ¶¶ [0008] and [0054]-[0055]. Even if neither Kennedy nor Nguyen is interpreted as explicitly changing a system identifier in response to a failure, Kennedy as modified by Nguyen and Safford would have rendered obvious such a change in system identifier. Kennedy does not discourage or otherwise limit its disclosure to replacing the boot processor with an active spare processor in response to a non-lockstep error, and it was well within the skill of a POSITA to modify Kennedy's system by Nguyen and Safford in the manner proposed, nor does Kennedy discourage the change of a system identifier. *See, e.g., Wolfe Decl.* (Ex. 1003) ¶ 125. Accordingly, the proposed combination of Kennedy, Nguyen, and Safford—in which Kennedy's system is modified by Nguyen and Safford to achieve dynamically

replacing a failed boot processor with an alternative processor while changing the system identifier of the failed processor in response to a loss of lockstep—would have been reasonably expected to succeed. *Id.*

Still further, to the extent that Kennedy and Nguyen do not explicitly describe the use of "hot spare" processors when failing over after a loss of lockstep, Safford recovers from the loss of lockstep by transferring the role of the failed processor pair to a hot standby processor pair. *Id.* at 4:37-40. As noted above, the hot standby processor pairs of Safford are ready to assume the functionality of a failed processor at any moment. *Id.* As such, a POSITA would be motivated to further improve the reliability of high-availability computer processing systems by transferring the role of a failed processor pair to a hot standby processor pair. Implementing hot standby processing pairs ensures that the failover occurs quickly and seamlessly and, as such, the POSITA would be motivated to combine Safford with the teachings of Kennedy and Nguyen. *Id.* at 4:40-54. Moreover, because each of the computing systems described in Nguyen, Kennedy, and Safford describe failover from one processor to another processor, the POSITA would reasonably expect to succeed in adding the hot spare processor pair to the computing architectures of Kennedy and Nguyen. *Wolfe Decl.* (Ex. 1003) ¶ 126.

H. Ground 3: Safford in light of Cepulis Presents Substantial New Questions of Patentability for Claims 1-4

1. Overview of Safford

A discussion of Safford appears above. *Supra* § G.3. That discussion is equally applicable to SNQ3 and will not be repeated here for the sake of brevity.

2. Overview of Cepulis

Cepulis generally describes "a multiprocessor computer system [that] handles the failure of one or more of its processors without totally disabling the system." *Cepulis* (Ex. 1014), Abstract. Cepulis describes a boot process for a multiprocessor computer system that includes designating one of the multiprocessor computer system's physical processors as a boot processor or "**first logical CPU**." *Id.* Moreover, the boot processor in Cepulis "retains its designation until power is cycled" unless it fails at which time its "designation is transferred to another active CPU." *Id.*

Like Safford, Cepulis uses the "**logical CPU 0**" designator to identify its boot processor as explained in greater detail in § II.A.

Cepulis is analogous art to the '302 Patent. For example, Cepulis is from the same field of endeavor (e.g., providing recovery in a high-availability multi-processor system) and is reasonably pertinent to the problem faced by the inventors of the '302 Patent. As noted above, the '302 Patent is directed to recovering from processor errors in multiprocessor systems in a manner that avoids crashing an operating system). *'302 Patent* (Ex. 1001) at 3:22-28. Similarly, Cepulis is directed to "a multiprocessor computer system [that] handles the failure of one or more of its processors without totally disabling the system." Cepulis (Ex. 1014), Abstract.

II. DETAILED APPLICATION OF THE PRIOR ART TO EVERY CLAIM FOR WHICH REEXAMINATION IS REQUESTED

A. Ground 1 (Kalyanasundharam in view of Kennedy)

Kalyanasundharam in view of Kennedy renders obvious each and every element of claims 1, 5-8, 13, 17, 21, 22, and 25-26. For ease of reference, Requester outlines examples from each reference that disclose or render obvious each associated claim limitation below. Further, a framework for a POSITA's background knowledge and motivations to combine are provided by Requester's expert, Andrew Wolfe (see Ex. 1003). A more detailed mapping of SNQ1 is provided in Appendix A.

Claim 1: ***[1P] A method comprising:***

Kalyanasundharam and Kennedy teach a method of processor failover by disclosing: "FIG. 6 is a flowchart illustrating operation of one embodiment of the node controller including the redundancy check circuit shown in FIG. 5 in response to receiving a communication from a processor core." *Kalyanasundharam* (Ex. 1005) at 2:7-10.

[1A] detecting loss of lockstep (LOL) for a processor in a multi-processor system

Kalyanasundharam and Kennedy teach detecting loss of lockstep (LOL) for a processor in a multi-processor system by disclosing an error detecting mechanism that checks for errors within processors 18A-18B that execute in lockstep. "In embodiments in which **the processor cores 18A-18B execute in lockstep**, the redundancy check circuit 26A may check that corresponding communications are concurrently received from the processor cores 18A-18B (e.g. received on the same clock cycle). For example, if a valid communication is received from one of the processor cores 18A-18B and a valid communication is not concurrently received from the other processor core 18A-18B, **an execution error may be detected**. *Id.* at 4:27-35).

[1B] determining that the processor for which said LOL is detected is assigned a role of boot processor

Kalyanasundharam and Kennedy teach determining whether a processor module is assigned a role of system boot processor by disclosing "provision needs to be made to determine which processor will boot the system, and if this "default" boot processor fails to boot the system, an alternative processor, if one is installed, should take over boot operations to enhance the availability of the system." *Kennedy* (Ex. 1006) at 7:5-13. Kalyanasundharam describes use of multiple pairs of processors that operate in lockstep:

> The processor cores may be configured as N/2 pairs of redundant processor cores. Each pair may redundantly execute different software code from other pairs. In some embodiments, the processor cores may be preassigned into pairs for redundant execution purposes. *Kalyanasundharam* (Ex. 1005) at 13:60-64.

Kalyanasundharam further notes: "If the masked comparison results indicate that the communication from the primary processor core mismatches with all auxiliary processor cores (that are enabled) (decision block 100, "yes" leg), the control circuit 84 may **disable the primary processor core and select one of the auxiliary processor cores to be the primary processor core** (block 102)." *Kalyanasundharam* (Ex. 1005) at 13:16-22. While Kalyanasundharam determines, upon detecting a loss of lockstep, whether a processor is a primary processor core, Kalyanasundharam fails to specifically indicate that the processor is a "boot processor." In analogous art, Kennedy shows a multiprocessor system that identifies default boot processors and alternative boot processors should the default boot processor fail to boot. *Kennedy* (Ex. 1006) at 7:5-13). This is directly analogous to being assigned the role of boot processor in the '302 Patent's system. "In the multiprocessor system 100 which utilizes a system bus 102, a number of problems can arise if more than one processor type is installed on the 64-bit bus 104. For instance, provision needs to be made to **determine which processor will boot the system**, and if this **"default" boot processor** fails to boot the system, **an alternative processor**, if one is installed, **should take over boot operations** to enhance the availability of the system." *Id.* It would have been obvious to combine Kalyanasundharam with Kennedy to expand Kalyanasundharam's ability to fail over to many different processors including, specifically, boot processors, as noted by Kennedy. Because Kalyanasundharam and Kennedy are both high-reliability computing systems, a POSITA would

reasonably expect to succeed when combining the boot processor of Kennedy to the primary and secondary processor cores of Kalyanasundharam. *Wolfe Decl.* (Ex. 1003) ¶ 75.

[1Ci] changing a system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected, to a different value

Kalyanasundharam and Kennedy teach changing a system identifier, used by the system's operating system to recognize the boot processor, to a different value by disclosing a control circuit that can "update the PC field of the configuration register 42 to reflect the newly selected primary processor core." *Kalyanasundharam* (Ex. 1005) at 13:22-29. Updating the PC field of a configuration register in Kalyanasundharam is equivalent (or at least would have been obvious) to changing the system identifier of claim 1 in the '302 patent because Kalyanasundharam's system uses the PC field to recognize which processor is the primary core and which is the secondary core and, as such, changes to the PC field will dictate how the system determines the identity of available processor cores. *Wolfe Decl.* (Ex. 1003) ¶ 76. Kalyanasundharam states: "As with the discussion above for disabling the auxiliary processor core, the control circuit 84 may simply **disable the primary processor core** from comparison **by clearing the corresponding enable bit in the configuration register 42**, or may disable the primary processor core from executing at all. The control circuit 84 may also **update the PC field of the configuration register 42** to reflect the newly selected primary processor core." *Kalyanasundharam* (Ex. 1005) at 13:22-29.

To the extent that changing a primary processor core to a secondary processor core (or vice versa) by changing an enable bit in a configuration register is not interpreted as changing a system identifier of a boot processor, Kennedy states, "If the CPU1 120 in slot 1 does not boot (proper boot is indicated by a change in the level of the SLOT SELECT signal line 193 from high to low) before the timer 184 in slot 2 times-out, then the CPU2 132 exits the reset state and attempts to boot the system as an alternate boot CPU." *Kennedy* (Ex. 1006) at 15:9-14. "If the CPU1 120 does not complete boot operations within the time allotted, then the CPU2 132 **takes over default boot operations as an alternative boot processor instead of CPU1** 120. If the CPU1 120 does boot properly, then the CPU2 132 checks in as an additional processor rather than taking over as a boot processor." *Kennedy* (Ex. 1006) at 13:38-44).

[1Cii] which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor

Kalyanasundharam and Kennedy teach causing the system's operating system to not recognize the processor for which said LOL is detected as the boot processor by disclosing that if one CPU fails to complete boot operations, another CPU will be used by the operating system as boot processor. Furthermore, to the extent that Kalyanasundharam is not interpreted to include a boot processor, Kalyanasundharam as modified by Kennedy, would cause the system's operating system to not recognize the processor for which said LOL is detected as the boot processor by indicating that slot select signals used by the operating system for communication would be changed in the event of a failure to boot: "If the CPU1 120 does not complete boot operations within the time allotted, then the CPU2 132 **takes over default boot operations as an alternative boot processor instead of CPU1** 120. If the CPU1 120 does boot properly, then the CPU2 132 checks in as an additional processor rather than taking over as a boot processor." *Kennedy* (Ex. 1006) at 13:38-44.

[1Di] changing a system identifier of the spare processor to that of the boot processor, Kalyanasundharam and Kennedy teach changing a system identifier of the spare processor to that of the boot processor by disclosing a control circuit that can "update the PC field of the configuration register 42 to reflect the newly selected primary processor core." *Kalyanasundharam* (Ex. 1005) at 13:22-29. Kalyanasundharam states: "As with the discussion above for disabling the auxiliary processor core, the control circuit 84 may simply **disable the primary processor core** from comparison **by clearing the corresponding enable bit in the configuration register 42**, or may disable the primary processor core from executing at all. The control circuit 84 may also **update the PC field of the configuration register 42** to reflect the newly selected primary processor core." *Id.* Moreover, to the extent that changing a primary processor core to a secondary processor core (or vice versa) by changing an enable bit in a configuration register is not interpreted as changing a system identifier of a boot processor, Kennedy states, "provision needs to be made to **determine which processor will boot the system, and if this 'default' boot processor fails** to boot the system**, an alternative processor,** if one is installed, **should take over boot operations** to enhance the availability of the system." *Kennedy* (Ex. 1006) at 7:5-13. Thus, as modified by Kennedy, Kalyanasundharam's system would have used bits in the configuration register to indicate that the new selected processor core would be the boot processor.

[1Dii] thereby switching the role of boot processor to a spare processor without shutting down the system's operating system.

Kalyanasundharam and Kennedy teach switching the role of boot processor to a spare processor without shutting down the system's operating system by disclosing a control circuit that can "update the PC field of the configuration register 42 to reflect the newly selected primary processor core." *Kalyanasundharam* (Ex. 1005) at 13:22-29. Since the configuration register is part of a redundancy checking circuit, changes to the configuration register would not require an operating system shutdown to implement. *Wolfe Decl.* (Ex. 1003) ¶ 80. To the extent that Kalyanasundharam is not interpreted to include a boot processor, Kalyanasundharam as modified by Kennedy, would cause the system's operating system to switch the role of boot processor to a spare processor: "If the CPU1 120 does not complete boot operations within the time allotted, then the CPU2 132 **takes over default boot operations as an alternative boot processor instead of CPU1** 120. If the CPU1 120 does boot properly, then the CPU2 132 checks in as an additional processor rather than taking over as a boot processor." *Kennedy* (Ex. 1006) at 13:38-44). Thus, as modified by Kennedy, Kalyanasundharam's system would have used bits in the configuration register to indicate that the new selected processor core would be the boot processor.

Claim 5: ***[5] wherein the spare processor is a processor module that includes a master and slave processors that operate in lockstep***

Kalyanasundharam teaches that the spare processor is a processor module that includes a master and slave processors that operate in lockstep by disclosing multiple processor core pairs, where one pair of processor cores is a primary core and another pair of processor cores is an auxiliary processor core that, together, are equivalent to master and slave processors.

> The processor cores may be configured as N/2 pairs of redundant processor cores. Each pair may redundantly execute different software code from other pairs. In some embodiments, the processor cores may be preassigned into pairs for redundant execution purposes. *Kalyanasundharam* (Ex. 1005) at 13:60-64.
>
> Generally, one of the processor cores 18A-18B may be designated as a primary processor core and the other processor core 18A-18B is referred to as the auxiliary processor core. The primary processor core may be the processor core whose execution results will be used as the result of executing the redundantly-executed software code. The auxiliary processor core is used to verify correct execution by the primary processor core." *Kalyanasundharam* (Ex. 1005) at 4:58-65.

Claim 6: ***[6] wherein said determining that the processor for which said LOL is detected is assigned the role of boot processor comprises accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor.***

To the extent that Kalyanasundharam is not interpreted to disclose accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor, Kalyanasundharam as modified by Kennedy teaches accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor. Kennedy states: "The CPU1 120 has a corresponding **non-volatile memory 123** which contains **configuration information and CPU specific initialize and self test code** (ISTC) for the CPU1 120…" *Kennedy* (Ex. 1006) at 5:15-18). It would have been obvious to a POSITA to combine Kalyanasundharam with Kennedy to increase the reliability of the system by storing and accessing CPU-specific information from non-volatile memory, as taught by Kennedy. *Id.*, *see also Wolfe Decl.* (Ex. 1003) ¶ 82.

Claim 7: ***[7] further comprising: determining the processor that is assigned the role of spare processor.***

Kalyanasundharam as modified by Kennedy teaches determining the processor that is assigned the role of spare processor by disclosing: "…provision needs to be made to **determine which processor will boot the system**, and if this **"default" boot processor fails** to boot the system, **an alternative processor**, if one is installed, **should take over boot operations** to enhance the availability of the system." *Kennedy* (Ex. 1006) at 7:5-13). It would have been obvious to a POSITA to combine Kalyanasundharam with Kennedy to increase the reliability of the system by identifying which processors are assigned certain roles, as taught by Kennedy. *Id.*, *see also Wolfe Decl.* (Ex. 1003) ¶ 83.

Claim 8: ***[8] wherein said determining the processor that is assigned the role of spare processor comprises: accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of spare processor.***

Kalyanasundharam as modified by Kennedy teaches accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of spare processor by disclosing an alternative processor that takes over if the default boot processor fails. Kennedy states: "…provision needs to be made to **determine which processor will boot the system**, and if this **"default" boot processor fails** to boot the system, **an alternative processor**, if one is installed, **should take over boot operations** to enhance the availability of the system." *Id.* at 7:5-13). It would have been obvious to a POSITA to combine Kalyanasundharam with Kennedy to avoid system errors by identifying which processors are assigned certain roles, as taught by Kennedy. *Id.*, *see also Wolfe Decl.* (Ex. 1003) ¶ 84.

Claim 13: ***[13P] A method comprising:***

See Limitation [1P].

[13A] system firmware determining that loss of lockstep (LOL) is detected for a lockstep pair of processors that are assigned the role of boot processor in a system;

See Limitation [1A].

[13B] determining one of said lockstep pair of processors that is not the cause of said LOL

See Limitation [1B].

[13C] copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to a spare processor

Kalyanasundharam and Kennedy teach copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to a spare processor by disclosing that state information for processor communications is stored. Kalyanasundharam states that "[a]dditionally, the SRO control unit 32 may be configured to update state in the SRO 30 for a given communication in response to completion of various internal activities. For example, if a probe is in the SRO 30 and the corresponding responses from the processors 18A-18B are received for the probe, the SRO control unit 32 may update the state in the entry storing the probe to record the receipt of the response and to record the coherency State reported in the response." *Kalyanasundharam* (Ex. 1005) at 7:34-43.

[13Di] changing a system identifier, used by the system's operating system to recognize the boot processor, of said lockstep pair of processors that are assigned the role of boot processor, to a different value

See Limitation [1Ci].

[13Dii] which causes the s system's operating system to not recognize the lock step pair of processors for which said LOL is detected as the processor

See Limitation [1Cii].

[13E] changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to said spare processor.

See Limitations [1Di] and [1Dii].

Claim 17: (Claim 17 recites substantially the same subject matter as Claim 1 ([1Dii]) and is therefore rejected for the same reasons as Claim 1 above.

Claim 21: ***[21P] A system comprising:***

Kalyanasundharam and Kennedy disclose a system 10 of FIG. 1 that includes nodes 12A-12B, memories 14A-14B, and peripheral devices 16A-16B" *Kalyanasundharam* (Ex. 1005) at 2:26-29.

[21A] a plurality of processor modules that each include a master processor and a slave processor that operate in lockstep

Kalyanasundharam and Kennedy teach a plurality of processor modules that each include a master processor and a slave processor that operate in lockstep by disclosing primary and auxiliary processors that operate in lockstep. "In one embodiment, **the processor cores 18A-18B may execute in lockstep** (that is, may execute the same instructions on the same clock cycle)." *Kalyanasundharam* (Ex. 1005) at 4:15-18. "Generally, one of the processor cores 18A-18B may be designated as a primary processor core and the other processor core 18A-18B is referred to as the auxiliary processor core. The primary processor core may be the processor core whose execution results will be used as the result of executing the redundantly-executed software code. **The auxiliary processor core is used to verify correct execution by the primary processor core**." *Id.* at 4:58-65).

[21B] an operating system;

"Generally, the processor cores 18A-18B may use the 15 interface(s) to the node controller 20A to communicate with other components of the computer system 10 (e.g., peripheral devices 16A-16B, processor cores 18C-18D, the memory controllers 22A-22B, etc.)." *Id.*, 3:15-18.

[21C] error detection logic operable to detect loss of lockstep (LOL) for at least one of said processor modules

Kalyanasundharam and Kennedy teach error detection logic operable to detect loss of lockstep (LOL) by disclosing an error detecting mechanism that checks for errors within processors 18A-18B that execute in lockstep. "In embodiments in which **the processor cores 18A-18B execute in lockstep**, the redundancy check circuit 26A may check that corresponding communications are concurrently received from the processor cores 18A-18B (e.g., received on the same clock cycle)." For example, if a valid communication is received from one of the processor cores 18A-18B and a valid communication is not concurrently received from the other processor core 18A-18B, **an execution error may be detected**. *Kalyanasundharam* (Ex. 1005) at 4:27-35.

[21Di] system firmware operable, responsive to the detection of LOL for a first of said processor modules, to determine whether said first processor module is assigned a role of system boot processor

Kalyanasundharam and Kennedy teach system firmware operable, responsive to the detection of LOL, to determine whether a processor module is assigned a role of system boot processor by disclosing "provision needs to be made to determine which processor will boot the system, and if this "default" boot processor fails to boot the system, an alternative processor, if one is installed, should take over boot operations to enhance the availability of the system." *Kennedy* (Ex. 1006) at 7:5-13. Kalyanasundharam notes: "If the masked comparison results indicate that the communication from the primary processor core mismatches with all auxiliary processor cores (that are enabled) (decision block 100, "yes" leg), the control circuit 84 may **disable the primary processor core and select one of the auxiliary processor cores to be the primary processor core** (block 102)." *Kalyanasundharam* (Ex. 1005) at 13:16-22. While Kalyanasundharam determines, upon detecting a loss of lockstep, whether a processor is a primary processor core, Kalyanasundharam fails to specifically indicate that the processor is a "boot processor." In analogous art, Kennedy shows a multiprocessor system that identifies default boot processors and alternative boot processors should the default boot processor fail to boot. This is directly analogous to being assigned the role of boot processor in the '302 Patent's system. "In the multiprocessor system 100 which utilizes a system bus 102, a number of problems can arise if more than one processor type is installed on the 64-bit bus 104. For instance, provision needs to

be made to **determine which processor will boot the system**, and if this **"default" boot processor** fails to boot the system, **an alternative processor**, if one is installed, **should take over boot operations** to enhance the availability of the system." *Kennedy* (Ex. 1006) at 7:5-13. It would have been to combine Kalyanasundharam with Kennedy to expand Kalyanasundharam's ability to fail over to many different processors including, specifically, boot processors, as noted by Kennedy. Because Kalyanasundharam and Kennedy are both high-reliability computing systems, a POSITA would reasonably expect to succeed when combining the boot processor of Kennedy to the primary and secondary processor cores of Kalyanasundharam. *Wolfe Decl.* (Ex. 1003) ¶ 94.

[21Dii] wherein if determined that said first processor module is assigned the role of system boot processor, said system firmware is further operable to cause said operating system to recognize another processor module as system boot processor without shutting down said operating system.

Kalyanasundharam and Kennedy teach system firmware that causes the operating system to recognize another processor module as system boot processor without shutting down the operating system by disclosing a control circuit that can "update the PC field of the configuration register 42 to reflect the newly selected primary processor core." *Kalyanasundharam* (Ex. 1005) at 13:22-29. Kalyanasundharam states: "As with the discussion above for disabling the auxiliary processor core, the control circuit 84 may simply **disable the primary processor core** from comparison **by clearing the corresponding enable bit in the configuration register 42**, or may disable the primary processor core from executing at all. The control circuit 84 may also **update the PC field of the configuration register 42** to reflect the newly selected primary processor core." *Id.*

Claim 22: ***[22] wherein the another processor module is a processor module assigned a role of spare processor.***

Kalyanasundharam and Kennedy teach that the other processor module is a processor module assigned a role of spare processor by disclosing an alternative processor that takes over if the default boot processor fails. Kennedy states: "…provision needs to be made to **determine which processor will boot the system**, and if this **"default" boot processor fails** to boot the system, **an alternative processor**, if one is installed, **should take over boot operations** to enhance the availability of the system." *Id.* at 7:5-13). It would have been obvious to a POSITA to combine Kalyanasundharam with Kennedy to increase reduce system errors by assigning processor roles, as taught by Kennedy. *Id.*, *see also Wolfe Decl.* (Ex. 1003) ¶ 96.

Claim 25: (Claim 25 recites substantially the same subject matter as Claim 8 ([8]) and is therefore rejected for the same reasons as Claim 8 above.

Claim 26: (Claim 26 recites substantially the same subject matter as Claim 21 ([21]) and is therefore rejected for the same reasons as Claim 21 above.

B. Ground 2 (Kennedy in view of Nguyen and Safford)

Kennedy in view of Nguyen and Safford renders obvious each and every element of claims 1-27. For ease of reference, Requester outlines examples from each reference that disclose or render obvious each associated claim limitation below. Further, a framework for a POSITA's background knowledge and motivations to combine are provided by Requester's expert. *See Wolfe* (Ex. 1003). A more detailed mapping of SNQ2 is provided in Appendix B.

Claim 1: ***[1P] A method comprising:***

Kennedy describes a method outlined in as follows: "The architecture provides a method and apparatus to interface multiple processors to the backplane system bus such that upon start-up of the computer system, one default boot CPU is designated to boot the system, and if this CPU fails, control transfers automatically to an alternate processor."." *Kennedy* (Ex. 1006) at 2:15-21.

[1A] detecting loss of lockstep (LOL) for a processor in a multi-processor system

Kennedy, as modified by Nguyen and Safford, teaches detecting loss of lockstep for a processor in a multi-processor system by disclosing "execution cores 120 [that] operate in lock step" and an FRC checker that compares "signals from execution cores 120 . . . to detect soft errors in either execution core." *Nguyen* (Ex. 1004) ¶ [0029]. "Execution cores 120 operate in lock step during normal FRC mode…" *Id.* ¶ [0024]. "In FRC mode, signals from execution cores 120 may be processed by FRC checker 130, which compares them **to detect soft errors in either execution core**." *Id.* ¶ [0029]. These soft errors or mismatches that are detected between processor pair

outputs are equivalent to detecting a loss of lockstep. *Wolfe Decl.* (Ex. 1003) ¶ 130. Indeed, Kennedy as modified by Nguyen further notes, "For this reason, FRC checker 130 is temporarily disabled **if error detector 140 detects an error in either error pipeline 128**, since execution cores 120 are **no longer in lock step**." *Id.* ¶ [0023]. As such, Kennedy as modified by Nguyen detects errors that indicate that two execution cores are no longer operating in lock step. This is equivalent to "detecting a loss of lockstep" as claimed in claim 1 of the '302 Patent. *Wolfe Decl.* (Ex. 1003) ¶ 130. To the extent that Nguyen is not held to teach a multi-processor system, Kennedy states, "The present invention provides a **multiprocessor interconnection architecture** with a backplane system bus for use with multiple processors." *Kennedy* (Ex. 1006) at 2:13-15. A POSITA would have been motivated to improve the reliability and robustness of Kennedy's multiprocessor system by detecting loss of lockstep, as taught by Nguyen, in order to fail over to an alternate boot processor in scenarios where a processor's failure is due to loss of lockstep. *Wolfe Decl.* (Ex. 1003) ¶ 130.

[1B] determining that the processor for which said LOL is detected is assigned a role of boot processor

Kennedy, Nguyen, and Safford teach determining that the processor for which said LOL is detected is assigned a role of boot processor by disclosing detecting a recoverable error and allowing a processor designated as "alternative processor" to take over for a processor designated as "boot processor." *Kennedy* (Ex. 1006) at 7:9-12. Kennedy discloses that: "…provision needs to be made to determine which processor will boot the system, and if this "default" boot processor fails to boot the system, an alternative processor, if one is installed, should take over boot operations to enhance the availability of the system." *Id.* at 7:5-13. Kennedy further states: "When multiple CPUs are installed in the system, if the **default boot processor exceeds the time allotted for it to boot (it fails)**, an alternative boot processor can take over boot operations as explained below. *Id.* at 9:65-68. Thus, Kennedy determines both that a recoverable failure has occurred, and that the failure occurred at a default boot processor. *Wolfe Decl.* (Ex. 1003) ¶ 131.

To the extent that Kennedy does not specifically indicate that the error detected was a loss of lockstep, Nguyen determines that an LOL occurs on its processor cores. *Nguyen*, (Ex. 1004) ¶ [0027]. "Execution cores 120 operate in lock step during normal FRC mode…" *Id.* ¶ [0024]. "In FRC mode, signals from execution cores 120 may be processed by FRC checker 130, which compares them **to detect soft errors in either execution core**." *Id.* ¶ [0029]. These soft errors or

mismatches that are detected between processor pair outputs are equivalent to detecting a loss of lockstep. Indeed, Nguyen further notes, "For this reason, FRC checker 130 is temporarily disabled **if error detector 140 detects an error in either error pipeline 128**, since execution cores 120 are **no longer in lock step**." *Id.* ¶ [0023]. As such, Nguyen detects errors that indicate that its two execution cores are no longer operating in lock step. This is equivalent to "detecting a loss of lockstep" as claimed in claim 1 of the '302 Patent. *Wolfe Decl.* (Ex. 1003) ¶ 132.

In this manner, the combination of Kennedy, Nguyen, and Safford determines whether a recoverable failure has occurred and whether the failure has occurred at a designated default boot processor. *Kennedy* (Ex. 1006) at 7:5-13. Nguyen indicates that such recoverable failures may include a loss of lockstep. *Nguyen*, (Ex. 1004) ¶ [0027]. Because Kennedy, Nguyen, and Safford describe high-availability computing architectures, a POSITA would be motivated to combine these references and would have a reasonable expectation that combining the two references would lead to a more reliable multiprocessor system. *Wolfe Decl.* (Ex. 1003) ¶ 133.

[1Ci] changing a system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected, to a different value

Kennedy as modified by Nguyen and Safford teaches this limitation by disclosing that, "To improve reliability of processing assets, a computer system employs lock stepped processor cores that operate in a **master/check pair**." *Safford* (Ex. 1007) at 2:55-57. Safford further describes one pair of these processors "operating in a lock step mode" as initially "appear[ing] to [a] computer system…to be…a **logical CPU 0,**" but later that **logical CPU 0** designation is reassigned to another processor upon failover. *See Safford* (Ex. 1007) at 3:4-5, 4:52-53. For example, "The processor pair 125/127 then becomes the **logical CPU 0** in the computer system 100". *Id.* A POSITA would have recognized Safford's logical CPU 0 as being the designation used by computing system 100 to identify its boot processor at least because, in multi-processor system architectures like Safford's, "**logical CPU 0**" was commonly used to indicate that a processor had been selected as a boot processor. *See, e.g., Cepulis* (Ex. 1014) at 2:52-64, *see also Wolfe Decl.* (Ex. 1003) ¶ 134. For example, during boot-processor selection,

> If CPU P0 is not functioning properly and cannot perform even the most basic functions, it is designated as inoperative. The hardware system then awakens CPU P1 and repeats the process of testing the CPU. If CPU P1 is operational, then it is designated as CPU L0, and it boots the remainder of the system. On the other hand, if CPU P1 also fails, it is also given an inoperative designation. The computer system then turns on CPU P2, and repeats the process. The process repeats until an operational microprocessor is found to perform the CPU L0 functions.[2]
> *Cepulis* (Ex. 1014) at 2:52-64.

A POSITA would have also recognized "CPU 0" (and other designators taught by Safford, such as "CPU 1") as being system designations used by computing system 100 to identify processors 111 and 113 in Safford's system (e.g., processors 111/113 are initially recognized as "logical CPU 0" and, after failure of one of these processors, processors 125/127 are designated as "logical CPU 0." *Safford* (Ex. 1007) at 3:4-5, 4:52-53. The use of system designators by a computing system is generally necessary when the role of boot processor can be assigned and reassigned to a processor regardless of its physical position within a system. *See Wolfe Decl.* (Ex. 1003) ¶ 135; *see also Safford* (Ex. 1007) at 4:52-53 ("The processor pair 125/127 then becomes the **logical CPU 0** in the computer system 100"). Thus, a POSITA would have understood "**logical CPU 0**" to be the boot-processor designation used by an operating system to determine which of its processors is the boot processor and/or to distinguish its boot processor from its non-boot processors. *See Wolfe Decl.* (Ex. 1003) ¶ 135. Moreover, Safford teaches the storing of state information for failed processors. *Safford*, (Ex. 1007) at 4:40-51. Because the state information of Safford's processors is copied from memory, a POSITA would have understood the memory to be storing the boot-processor designation (*e.g.,* logical CPU 0) for the processors so assigned by Safford's system. *See Id.* In view of these citations and the knowledge that a boot processor was a common component identified as "logical CPU 0" in computer systems (*e.g.,* Safford's system), a POSITA would have understood Safford as disclosing data storage that stores information assigning the role of boot processor to one of the processors and changing that assignment by changing the identifier "logical CPU 0" during a failover. *See supra* § I.C.2; *see also Wolfe Decl.* (Ex. 1003) ¶ 135. Still further, A POSITA would have combined Safford with Kennedy and Nguyen because each is concerned with providing processor failover mechanisms in high-availability environments, and because Kennedy and Nguyen would perform the failover more

[2] Cepulis is provided herein to explain the meaning of the term "boot processor" used in Safford. *See* M.P.E.P. § 2131.01.

reliably when changing a system identifier as taught by Safford. *Id., see also Wolfe Decl.* (Ex. 1003) ¶ 135.

[1Cii] which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor

Kennedy, Nguyen, and Safford teach causing the system's operating system to not recognize the processor for which said LOL is detected as the boot processor by disclosing that a failed processor pair 111/113 is taken off-line, that its state information is copied to another processor pair (e.g., a "hot standby" processor 125/127), and that [t]he processor pair 125/127 then becomes the logical CPU 0 in the computer system 100, and the computer system 100 operates without a hot standby processor pair. *Safford* (Ex. 1007) at 4:38-53. Thus, in Kennedy's system as modified by Nguyen and Safford, in the event of a processor failure (such as a loss of lockstep, as taught by Nguyen), a new processor is designated as the logical CPU 0, and the old (failed) processor no longer retains that designation. *Id.*, see also *Wolfe Decl.* (Ex. 1003) ¶ 136. A POSITA would have been motivated to improve the reliability of Kennedy's system by incorporating a change in system identifiers, as taught by Safford, to ensure that its operating system would no longer refer to failed processors as "logical CPU 0" when booting. *Wolfe Decl.* (Ex. 1003) ¶ 136.

[1Di] changing a system identifier of the spare processor to that of the boot processor

Kennedy as modified by Safford teaches that "To improve reliability of processing assets, a computer system employs lock stepped processor cores that operate in a **master/check pair**." *Safford* (Ex. 1007) at 2:55-57. Safford further describes one pair of these processors "operating in a lock step mode" as initially "appear[ing] to the computer system…to be…a **logical CPU 0,**" but later that **logical CPU 0** designation is reassigned to another processor. *See Safford* (Ex. 1007) at 3:4-5, 4:52-53 ("The processor pair 125/127 then becomes the **logical CPU 0** in the computer system 100"). *Id., see also Wolfe Decl.* (Ex. 1003) ¶ 137. By changing the former standby processor 125/127 to the "logical CPU 0" processor, the system identifier of the spare processor (e.g., CPU 1) is changed to that of the boot processor (e.g., CPU 0). *Id., see also Wolfe Decl.* (Ex. 1003) ¶ 137. In light of the above citations, a POSITA would have understood "**logical CPU 0**" to be the boot-processor designation used by an operating system to determine which of its processors is the boot processor and/or to distinguish its boot processor from its non-boot processors. *See Wolfe Decl.* (Ex. 1003) ¶ 137. Moreover, a POSITA would have been motivated to improve the reliability of Kennedy's system by changing these system identifiers during failover, as taught by Safford, to

ensure that its operating system would properly refer to the designated "logical CPU 0" when operating the processor to which failover occurred. *Id.*

[1Dii] thereby switching the role of boot processor to a spare processor without shutting down the system's operating system

Kennedy as modified by Nguyen and Safford teaches this limitation. Kennedy notes that an error can lead to switching the role of boot processor from one processor to another: "The architecture provides a method and apparatus to interface multiple processors to the backplane system bus such that upon start-up of the computer system, one default boot CPU is designated to boot the system, and if this CPU fails, **control transfers automatically to an alternate processor**." *Kennedy* (Ex. 1006) at 2:15-21. Moreover, Kennedy states that: "…provision needs to be made to determine which processor will boot the system, and if this "default" boot processor fails to boot the system, an alternative processor, if one is installed, should take over boot operations to enhance the availability of the system." *Id.* at 7:5-13. As such, Kennedy as modified by Nguyen and Safford teaches an automatic switching of roles from default boot processor to alternative processor or vice versa. *Wolfe Decl.* (Ex. 1003) ¶ 138.

To the extent that Kennedy is not interpreted to teach role switching of a boot processor to a spare processor without shutting down the system's operating system, Nguyen teaches switching from a processor to a spare processor without shutting down the system's operating system. In combination, Kennedy would be modified by Nguyen and Safford to include PAL-based implementations that do not require a system shutdown. Specifically, Kennedy, as modified by Nguyen and Safford, would switch from a boot processor to an alternative processor using a PAL-based, firmware implementation that would allow the switch between processors to be performed quickly enough to avoid trigging a timeout period that would otherwise be enforced by the operating system. *Wolfe Decl.* (Ex. 1003) ¶ 139. Nguyen notes that "PAL-based implementations of the recovery mechanism should be able to complete quickly enough to avoid triggering a time-out period enforced by the operating system. *Nguyen* (Ex. 1004) ¶ [0045]." Nguyen states:

> As noted above, the recovery mechanism for errors detected within the FRC boundary may be handled by **various combinations of hardware, software and firmware modules**. One embodiment of the recovery mechanism employs code that is closely associated with the processor. For example, the Itanium® Processor Family of Intel® Corporation, employs a **layer of firmware** called the processor abstraction layer (PAL), which **provides**

> **an abstraction of the processor to the rest of the computing system**. Implementing recovery in the PAL **hides the recovery process from system level code**, such as the system abstraction layer (SAL), e.g. BIOS, **and the operating system**. PAL-based implementations of the recovery mechanism should be able to complete quickly enough to **avoid triggering a time-out period enforced by the operating system**.
>
> *Id.*

In this example, implementing PAL-based firmware to implement the recovery mechanism quickly enough to avoid triggering a time-out period enforced by the operating system is equivalent to performing the recovery mechanism without shutting down the system's operating system. *Wolfe Decl.* (Ex. 1003) ¶ 140. Using a firmware-based PAL layer hides the recovery process from system level code such as the operating system. Changes in system level code would require an operating system shutdown to implement. *Id.* PAL-based implementations, however, provide recovery without triggering a time-out period or other operations enforced by the operating system. This is equivalent to avoiding an operating system shutdown. Accordingly, the recovery mechanisms discussed in Nguyen are carried out in firmware, without shutting down the operating system. *Id.* A POSITA would have been motivated to improve the efficiency of Kennedy's failover mechanism by implementing PAL-based firmware, as taught by Nguyen, in order to quickly fail over to an alternate boot processor in a seamless and prompt manner. *Id.*

Claim 2: ***[2] wherein the spare processor is a hot spare that is held in idle until switched the role of boot processor***

To the extent that Kennedy and Nguyen to not explicitly teach a hot spare that is held in idle until switched to the role of boot processor, Kennedy and Nguyen, when combined with Safford, teach that the spare processor is a hot spare that is held in idle until switched the role of boot processor by disclosing a spare processor that is initially idled and becomes active upon a first processor unit losing lock step. *Safford* (Ex. 1007) at 2:26-32. Safford states: "The processor 125 may be designated as a "**hot standby**" and is **sitting idle** in lock step mode with the processor 127. Should one of the processors 111, 113, 121, and 123 suffer an error, the hot standby processors 125, 127 may be used to **speed recovery** from the resulting loss of lock step. *Id.* at 4:25-29). A POSITA would have combined Safford with Kennedy and Nguyen because each is concerned with providing processor failover mechanisms in high-availability environments, and because Kennedy

and Nguyen would perform the failover more quickly when using a hot spare as taught by Safford. *Id., see also Wolfe Decl.* (Ex. 1003) ¶ 141.

Claim 3: ***[3] wherein the spare processor is an active spare***

To the extent that Kennedy and Nguyen to not explicitly teach an active spare that is used as a spare processor, Kennedy and Nguyen, when combined with Safford, teach that the spare processor is an active spare by disclosing "moving the architected state comprises **copying the architected state directly** from the loss of lock step processor unit **to the spare processor unit**." *Safford* (Ex. 1007) at 5:23-26). A POSITA would have combined Safford with Kennedy and Nguyen because each is concerned with providing processor failover mechanisms in high-availability environments, and because Kennedy and Nguyen would be able to perform additional processing when using an active spare, as taught by Safford. *Id., see also Wolfe Decl.* (Ex. 1003) ¶ 142.

Claim 4: ***[4] wherein said switching further comprises: causing the system's operating system to idle the active spare***

To the extent that Kennedy and Nguyen to not explicitly teach causing the system's operating system to idle the active spare, Kennedy and Nguyen, when combined with Safford, teach that switching further includes causing the system's operating system to idle the active spare by disclosing a "means for moving an architected state of a processor unit having the loss of lock step initiating event to a **spare processor unit**, the spare processor unit **operating idle** in lock step." *Safford* (Ex. 1007) at 6:2-5). A POSITA would have combined Safford with Kennedy and Nguyen because each is concerned with providing processor failover mechanisms in high-availability environments, and because Kennedy and Nguyen would be able to reduce processing resource usage by idling the active spare, as taught by Safford. *Id., see also Wolfe Decl.* (Ex. 1003) ¶ 143.

Claim 5: ***[5] wherein the spare processor is a processor module that includes a master and slave processors that operate in lockstep.***

Kennedy, in combination with Nguyen and Safford, teaches a multiprocessor system that includes boot processors and alternative processors. *Kennedy* (Ex. 1006) at 2:62-64 and 7:7-13. Kennedy, in combination with Nguyen and Safford, further discloses a pair of processors that operate in lockstep by disclosing that:

> In FRC mode, **one of execution cores 120 may be designated as the master**. The master refers to the execution core that is responsible for updating the resources shared by execution cores 120. **The other execution core 120 may be designated as the slave**. The slave is responsible for generating results from the same code sequence to be checked against those of the master."
>
> *Nguyen* (Ex. 1004) ¶ [0027].

These execution cores 120 "operate in lock step during normal FRC mode…" *Id.* ¶ [0024]. A POSITA would have combined Nguyen with Kennedy and Safford because each is concerned with providing reliable processor failover mechanisms in high-availability environments, and because Kennedy and Safford would perform processor failovers more reliably when using a spare processor with master and slave execution cores as taught by Nguyen. *Id., see also Wolfe Decl.* (Ex. 1003) ¶ 145.

Claim 6: ***[6] wherein said determining that the processor for which said LOL is detected is assigned the role of boot processor comprises accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor.***

Kennedy, as modified by Nguyen and Safford, teaches accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor by disclosing that "The CPU1 120 has a corresponding **non-volatile memory 123 which contains configuration information and CPU specific initialize and self test code (ISTC) for the CPU1 120**…" *Kennedy* (Ex. 1006) at 5:15-18. Kennedy further indicates that: "The slot select logic 148 is described in more detail in reference to the CPU1 120 installed in slot 1 as the default boot processor and the CPU2 132 installed in slot 2 as the alternate boot processor. *Id.* at 10:55-58, see also Table 2. It would have been obvious to combine Kennedy with Nguyen and Safford to increase the dependability of the system by storing configuration information in non-volatile memory, as taught by Kennedy. *Id., see also Wolfe Decl.* (Ex. 1003) ¶ 146.

Claim 7: ***[7] further comprising: determining the processor that is assigned the role of spare processor.***

Kennedy as modified by Nguyen and Safford teaches determining the processor that is assigned the role of spare processor by disclosing that "…provision needs to be made to **determine**

which processor will boot the system, and if this "default" boot processor fails to boot the system, **an alternative processor**, if one is installed, **should take over boot operations** to enhance the availability of the system." *Kennedy* (Ex. 1006) at 7:5-13, *see also* Nguyen (Ex. 1004) at ¶ [0026]. It would have been obvious to a POSITA to combine Kennedy with Nguyen and Safford to increase the reliability of the system by identifying which processors are assigned certain roles, as taught by Kennedy. *Id.*, *see also Wolfe Decl.* (Ex. 1003) ¶ 147.

Claim 8: ***[8] wherein said determining the processor that is assigned the role of spare processor comprises: accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of spare processor.***

Kennedy as modified by Nguyen and Safford teaches accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor by disclosing that "The CPU1 120 has a corresponding **non-volatile memory 123 which contains configuration information and CPU specific initialize and self test code (ISTC) for the CPU1 120**..." *Kennedy* (Ex. 1006) at 5:15-18. Kennedy further indicates that: "The slot select logic 148 is described in more detail in reference to the CPU1 120 installed in slot 1 as the default boot processor and the CPU2 132 installed in slot 2 as the alternate boot processor. *Id.* at 10:55-58, *see also* Table 2. It would have been obvious to a POSITA to combine Kennedy with Nguyen to avoid system errors by identifying which processors are assigned certain roles, as taught by Kennedy. *Id.*, *see also Wolfe Decl.* (Ex. 1003) ¶ 148.

Claim 9: ***[9] wherein said determining step is performed by system firmware.***

Kennedy as modified by Nguyen teaches determining whether an error has occurred for which recovery methods are to be used. *Kennedy,* (Ex. 1006) at 7:7-13. Kennedy as modified by Nguyen further teaches that a firmware-based processor abstraction layer is usable in such a recovery process. *Nguyen* (Ex. 1004) ¶ [0045]. In recovery scenarios, Kennedy as modified by Nguyen implements a firmware-based processor abstraction layer (PAL), which "**provides an abstraction of the processor** to the rest of the computing system." *Id.* "**Implementing recovery in the PAL** hides the recovery process from system level code, such as the system abstraction layer (SAL), e.g. BIOS, and the operating system." *Id.* As such, Kennedy as modified by Nguyen indicates that the process steps involved in its recovery processes, including determining steps, is performed by firmware. *Wolfe Decl.* (Ex. 1003) ¶ 149.

Claim 10: ***[10] wherein said switching step is performed by system firmware.***

Kennedy as modified by Nguyen teaches determining whether an error has occurred for which recovery methods are to be used. *Kennedy,* (Ex. 1006) at 7:7-13 Kennedy as modified by Nguyen further teaches that the switching step is performed by system firmware by implementing a firmware-based processor abstraction layer in the recovery process. *Nguyen* (Ex. 1004) ¶ [0045]. In recovery scenarios, Nguyen implements a firmware-based processor abstraction layer (PAL), which "provides an abstraction of the processor to the rest of the computing system." *Id.* "**Implementing recovery in the PAL** hides the recovery process from system level code, such as the system abstraction layer (SAL), e.g. BIOS, and the operating system." *Id.* As such, Kennedy as modified by Nguyen indicates that the process steps involved in its recovery processes, including switching steps, is performed by firmware.

Claim 11: ***[11] wherein the processor for which said LOL is detected includes a lockstep pair of processors, and wherein said switching comprises: determining one of the lockstep pair of processors that is not the cause of said LOL; and copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to said spare processor.***

Kennedy as modified by Nguyen and Safford teaches determining one of the lockstep pair of processors that is not the cause of said LOL and copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to said spare processor by disclosing, "the node controller 130 may store the architected state of the processors 111, 113, 121, 123, 125, and 127, either internally in the node controller 130, or in another component of the computer system 100." *Safford*, (Ex. 1007) at 4:10-14. While nodes 111 and 113 are indicated as having potentially lost lockstep, nodes 121, 123, 125, and 127 have not lost lockstep and, according to Safford, the node controller 130 is capable of copying and storing the state information of those processors that have not caused a loss of lockstep. *Id.* at 4:6-18. A POSITA would have been motivated to combine Safford's copying of state information for non-failed processors (i.e., processors that have not lost lockstep) in order to further bolster the reliability of its high-availability system through the copying of processor state information for non-failed, functioning nodes. *Wolfe Decl.* (Ex. 1003) ¶ 151.

Claim 12: ***[12] wherein said spare processor includes a lockstep pair of processors, and wherein said copying comprises: copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to each of said lockstep pair of processors of said spare processor.***

Kennedy as modified by Nguyen and Safford teaches copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to each of said lockstep pair of processors of said spare processor by disclosing, "the node controller 130 may store the architected state of the processors 111, 113, 121, 123, 125, and 127, either internally in the node controller 130, or in another component of the computer system 100." *Safford*, (Ex. 1007) at 4:10-14. While nodes 111 and 113 are indicated as having potentially lost lockstep, nodes 121, 123, 125, and 127 have not lost lockstep and, according to Safford, the node controller 130 is capable of copying and storing the state information of those processors that have not caused a loss of lockstep. *Id.* at 4:6-18. Safford specifically notes that the state information is copied to each processor in the pair of processors (e.g., to processor pairs 121/123 or 125/127). *Id.* A POSITA would have been motivated to combine Safford's copying of state information for non-failed (working) nodes in order to further increase the reliability of its high-availability system by keeping a store of state information copied from working nodes. *Wolfe Decl.* (Ex. 1003) ¶ 152.

Claim 13: ***[13P] A method comprising:***

See Limitation [1A].

[13A] system firmware determining that loss of lockstep (LOL) is detected for a lockstep pair of processors that are assigned the role of boot processor in a system

See Limitation [1A].

[13B] determining one of said lockstep pair of processors that is not the cause of said LOL

Kennedy as modified by Nguyen and Safford teaches determining one of said lockstep pair of processors that is not the cause of said LOL by disclosing detecting an error that is localized to one of the execution cores. Kennedy as modified by Nguyen and Safford states: "As long as the error is detected before it triggers an FRC reset, the corrupted data can be localized to one of the execution cores, leaving the **machine state data** of the other execution core **available for recovery**." *Nguyen* (Ex. 1004) ¶ [0046]).

[13C] copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to a spare processor

Kennedy as modified by Nguyen and Safford teaches copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to a spare processor by

disclosing "the node controller 130 may store the architected state of the processors 111, 113, 121, 123, 125, and 127, either internally in the node controller 130, or in another component of the computer system 100." *Safford*, (Ex. 1007) at 4:10-14. While nodes 111 and 113 are indicated as having potentially lost lockstep, nodes 121, 123, 125, and 127 have not lost lockstep and, according to Safford, the node controller 130 is capable of copying and storing the state information of those processors that have not caused a loss of lockstep. *Id.* at 4:6-18. A POSITA would be motivated to combine Safford's copying of state information for non-failed, working nodes in order to further increase the reliability of its high-availability system through the storing of processor state information copied from working nodes. *Wolfe Decl.* (Ex. 1003) ¶ 156.

[13Di] changing a system identifier, used by the system's operating system to recognize the boot processor, of said lockstep pair of processors that are assigned the role of boot processor, to a different value

See Limitation [1Ci].

[13Dii] which causes the system's operating system to not recognize the lock step pair of processors for which said LOL is detected as the processor

See Limitation [1Cii].

[13E] changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to said spare processor.

See Limitation [1Di].

Claim 14: (Claim 14 recites substantially the same subject matter as Claim 2 ([2]) and is therefore rejected for the same reasons as Claim 2 above).

Claim 15: (Claim 15 recites substantially the same subject matter as Claim 3 ([3]) and is therefore rejected for the same reasons as Claim 3 above).

Claim 16: (Claim 16 recites substantially the same subject matter as Claim 4 ([4]) and is therefore rejected for the same reasons as Claim 4 above).

Claim 17: (Claim 17 recites substantially the same subject matter as Claim 1 ([1Dii]) and is therefore rejected for the same reasons as Claim 1 above).

Claim 18: (Claim 18 recites substantially the same subject matter as Claim 9 and is therefore rejected for the same reasons as Claim 9 above).

Claim 19: (Claim 19 recites substantially the same subject matter as Claim 11 and is therefore rejected for the same reasons as Claim 11 above).

Claim 20: (Claim 20 recites substantially the same subject matter as Claim 10 and is therefore rejected for the same reasons as Claim 10 above).

Claim 21: (Claim 21 recites substantially the same subject matter as Claim 1 and is therefore rejected for the same reasons as Claim 1 above).

Claim 22: ***[22] wherein the another processor module is a processor module assigned a role of spare processor.***

Kennedy as modified by Nguyen and Safford teaches the other processor module is a processor module assigned a role of spare processor by disclosing that "…provision needs to be made to **determine which processor will boot the system**, and if this "default" boot processor **fails to boot the system, an alternative processor**, if one is installed, **should take over boot operations** to enhance the availability of the system." *Kennedy* (Ex. 1006) at 7:5-13. To the extent that Kennedy is not interpreted to disclose a processor module, Nguyen discloses a processor 110 that includes execution cores 120(a) and 120(b), which a POSITA would recognize as a processor module. *Nguyen* (Ex. 1004) ¶ [0027], *see also Wolfe Decl.* (Ex. 1003) ¶ 160. It would have been obvious to a POSITA to combine Kennedy with Nguyen and Safford to reduce system errors by assigning processor roles, as taught by Kennedy, and by changing system identifiers for those processors, as taught by Safford. *Id.*, *see also Wolfe Decl.* (Ex. 1003) ¶ 160.

Claim 23: (Claim 23 recites substantially the same subject matter as Claims 2 and 14 ([2] and [14]) and is therefore rejected for the same reasons as Claims 2 and 14 above).

Claim 24: (Claim 24 recites substantially the same subject matter as Claims 2 and 14 ([2] and [14]) and is therefore rejected for the same reasons as Claims 2 and 14 above).

Claim 25: (Claim 25 recites substantially the same subject matter as Claim 8 and is therefore rejected for the same reasons as Claim 8 above).

Claim 26: (Claim 26 recites substantially the same subject matter as Claim 1 and is therefore rejected for the same reasons as Claim 1 above).

Claim 27: ***[27] wherein said processor in said multi-processor system is an Itanium Processor Family (IPF) processor.***

Kennedy in combination with Nguyen and Safford teaches a multiprocessor system that includes boot and alternative processors. *Kennedy* (Ex. 1006) at 2:13-15. Kennedy in combination with Nguyen and Safford further teaches that a processor in such a multi-processor system is an Itanium Processor Family (IPF) processor by specifically disclosing use of an Itanium processor in its system. *Nguyen* (Ex. 1004) ¶ [0045]. Kennedy in combination with Nguyen specifically mentions that Itanium processors are used in its system: "For example, the **Itanium(R) Processor Family** of Intel(R) Corporation, employs a layer of firmware called the processor abstraction layer (PAL), which provides an abstraction of the processor to the rest of the computing System." *Id.* In view of the above citations, Kennedy as modified by Nguyen and Safford teaches that the processors would have been IPF processors. *See Wolfe Decl.* (Ex. 1003) ¶ 161. A POSITA would have been motivated to combine these disclosures from Kennedy, Nguyen, and Safford to improve the spatial efficiency of the system by consolidating the processors into the multicore processor package used by the IPF. *Id.* It was well within the skill of a POSITA to modify Kennedy's system by Nguyen and Safford in this manner, and thus this combination of Kennedy, Nguyen, and Safford would have been expected to succeed. *Id.* Accordingly, Kennedy as modified by Nguyen and Safford teaches this claim limitation.

C. Ground 3: (Safford in view of Cepulis)

Safford in view of Cepulis renders obvious each and every element of claims 1-4. For ease of reference, Requester outlines examples from each reference that disclose or render obvious each associated claim limitation below. Further, a framework for a POSITA's background knowledge and motivations to combine are provided by Requester's expert, Andrew Wolfe (see Ex. 1003). A more detailed mapping of Ground 3 is provided in Appendix C.

Claim 1:

[1P] A method comprising

To the extent the preamble is limiting, Safford discloses a "method . . . used to enhance recovery from loss of lock step in a multi-processor computer system." *Safford* (Ex. 1007), Abstract.

[1A] detecting loss of lockstep (LOL) for a processor in a multi-processor system

Safford discloses, or at least renders obvious, this limitation by disclosing (1) two "processors [that] may operate . . . in a lock step mode" and (2) associated "error detection and signaling logic" that are configure to "detect an error that will cause a loss of lock step." *Safford* (Ex. 1007) at 3:1-45. Additionally, Safford discloses "[detecting] a loss of lock step. . . [by comparing] outputs from [two] microprocessor cores" and detecting "[a] difference in processing" that "is by definition a loss of lock step." *Id.* at 1:51-55. Safford further discloses "detecting a loss of lock step initiating event in a processor." *Id.*, claim 13.

More specifically, Safford discloses that "the processor 111 detects an error event that indicates an impending loss of lock step. In block 215, the processor 111 signals the node controller 130 that the first processor pair 111/113 has broken lock step and that the first processor pair 111/113 should be taken 'off-line.'" *Safford* (Ex. 1007) at 4:35-40.

[1B] determining that the processor for which said LOL is detected is assigned a role of boot processor

Safford discloses, or at least renders obvious, this limitation by disclosing a pair of processors "operating in a lock step mode" that initially "appear to [a] computer system . . . to be . . . a logical CPU 0" but, later, have the logical CPU 0 designation reassigned to another processor. *See Safford* (Ex. 1007) at 3:4-5, 4:52-53 ("The processor pair 125/127 then becomes the logical CPU 0 in the computer system 100").

A POSITA would have recognized "logical CPU 0" as being the designation used by computing system 100 to identify its boot processor for at least the reason that, in multi-processor system architectures like that described in Safford, "logical CPU 0" is commonly used to indicate that a processor has been selected as a boot processor. *See Cepulis* (Ex. 1014) at 2:52-64, *see also Wolfe Decl.* (Ex. 1003) ¶ 172. For example, during boot-processor selection,

> If CPU P0 is not functioning properly and cannot perform even the most basic functions, it is designated as inoperative. The hardware system then awakens CPU P1 and repeats the process of testing the CPU. If CPU P1 is operational, then it is designated as CPU L0, and it boots the remainder of the system. On the other hand, if CPU P1 also fails, it is also given an inoperative designation. The computer system then turns on CPU P2, and repeats the process. The process repeats until an operational microprocessor is found to perform the CPU L0 functions.[3]
>
> *Cepulis* (Ex. 1014) at 2:52-64.

A POSITA would have also recognized "CPU 0" and "CPU 1" as being physical designations used by computing system 100 to identify processors 111 and 113. The use of both logical and physical designators by a computing system is generally necessary when the role of boot processor can be assigned and reassigned to a processor regardless of its physical position within a system. *See Wolfe Decl.* (Ex. 1003) ¶ 173, *see also Safford* (Ex. 1007) at 4:52-53 ("The processor pair 125/127 then becomes the logical CPU 0 in the computer system 100").

Thus, a POSITA would have understood "logical CPU 0" to be a boot-processor designation used by an operating system to determine which of its processors is a boot processor and/or to distinguish its boot processor from its non-boot processors. *See Wolfe Decl.* (Ex. 1003) ¶ 174. As such, *determining if* a processor is *assigned the role of boot processor* is performed by the operating system through recognition of the boot processor designation. *See Wolfe Decl.* (Ex. 1003) ¶ 174. Accordingly, Safford discloses, or at least renders obvious, *determining if* a processor is *assigned the role of boot processor* by disclosing a pair of processors "operating in a lock step mode" that initially "appear to [a] computer system . . . to be . . . a logical CPU 0" but, later, have the logical CPU 0 designation reassigned to another processor. *See Safford* (Ex. 1007) at 3:4-5, 4:52-53 ("The processor pair 125/127 then becomes the logical CPU 0 in the computer system 100"), *see also Wolfe Decl.* (Ex. 1003) ¶ 174.

[1C] changing a system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected, to a different value which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor

[3] Cepulis is provided herein to explain that a POSITA would have understood the meaning of the term "CPU 0" used in Safford to be the "boot processor." *See* MPEP § 2131.01.

Safford discloses, or at least renders obvious, this limitation by disclosing a pair of processors "operating in a lock step mode" that initially "appear to [a] computer system . . . to be . . . a logical CPU 0" but, later, have the logical CPU 0 designation reassigned to another processor. *See Safford* (Ex. 1007) at 3:4-5, 4:52-53 ("The processor pair 125/127 then becomes the logical CPU 0 in the computer system 100"). A POSITA would have recognized "logical CPU 0" as being the designation used by computing system 100 to identify its boot processor for at least the reason that, in multi-processor system architectures like that described in Safford, "logical CPU 0" is commonly used to indicate that a processor has been selected as a boot processor. *See Cepulis* (Ex. 1014) at 2:52-64, *see also Wolfe Decl.* (Ex. 1003) ¶ 175. For example, during boot-processor selection,

> If CPU P0 is not functioning properly and cannot perform even the most basic functions, it is designated as inoperative. The hardware system then awakens CPU P1 and repeats the process of testing the CPU. If CPU P1 is operational, then it is designated as CPU L0, and it boots the remainder of the system. On the other hand, if CPU P1 also fails, it is also given an inoperative designation. The computer system then turns on CPU P2, and repeats the process. The process repeats until an operational microprocessor is found to perform the CPU L0 functions.

Cepulis (Ex. 1014) at 2:52-64.

Moreover, Safford indicates that, after a first pair of processors has been taken offline, another pair of processors assumes the designation or identifier of CPU0, causing the operating system to no longer recognize processor pair 111 and 113 as the boot processor (i.e., CPU0):

> Alternatively, the node controller 130 may copy the state of the first processor pair 111/113 directly from the processors 111 and 113 to the processors 125 and 127 without any intermediate storage of the architected state in the node controller 130, or other component of the computer system 100. The processor pair 125/127 then becomes the logical CPU 0 in the computer system 100, and the computer system 100 operates without a hot standby processor pair.

Safford (Ex. 1007) at 4:47-54.

[1D] changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to a spare processor without shutting down the system's operating system

Safford indicates that, after a first pair of processors has been taken offline, another pair of processors assumes the designation or identifier of CPU0, thereby causing the operating system to recognize a new processor pair 125/127 as the boot processor (i.e., CPU0):

> Alternatively, the node controller 130 may copy the state of the first processor pair 111/113 directly from the processors 111 and 113 to the processors 125 and 127 without any intermediate storage of the architected state in the node controller 130, or other component of the computer system 100. The processor pair 125/127 then becomes the logical CPU 0 in the computer system 100, and the computer system 100 operates without a hot standby processor pair.

Safford (Ex. 1007) at 4:47-54.

To the extent that Safford in view of Cepulis does not disclose an operating system, a POSITA would recognize that "each processor [would have] its own memory and [would run] its own copy of the operating system." *Commercial Fault Tolerance* (Ex. 1010), section 4.2, *see also Wolfe Decl.* (Ex. 1003) ¶ 178. Moreover, to the extent that Safford in view of Cepulis does not explicitly state that the transition from boot to spare processor is performed without shutting down the operating system, a POSITA would recognize that the intent of Safford's recovery system is to recover from loss of lockstep without crashing or restarting the operating system. *Wolfe Decl.* (Ex. 1003) ¶ 178.

Claim 2: ***[2] wherein the spare processor is a hot spare that is held in idle until switched the role of boot processor.***

Safford states: "The processor 125 may be designated as a "**hot standby**" and is **sitting idle** in lock step mode with the processor 127. Should one of the processors 111, 113, 121, and 123 suffer an error, the hot standby processors 125, 127 may be used to **speed recovery** from the resulting loss of lock step. *Safford* (Ex. 1007) at 4:25-29).

Claim 3: ***[3] wherein the spare processor is an active spare.***

Safford teaches that the spare processor is an active spare by disclosing "moving the architected state comprises **copying the architected state directly** from the loss of lock step processor unit **to the spare processor unit**." *Id.* at 5:23-26).

Claim 4: ***[4] wherein said switching further comprises: causing the system's operating system to idle the active spare.***

Safford teaches that switching further includes causing the system's operating system to idle the active spare by disclosing a "means for moving an architected state of a processor unit having the loss of lock step initiating event to a **spare processor unit**, the spare processor unit **operating idle** in lock step." *Id.* at 6:2-5.

D. Secondary Considerations

This Request demonstrates that the Challenged Claims of the '302 Patent are unpatentable as obvious in view of the prior art references. The Applicant did not identify any evidence of secondary considerations during prosecution. Further, the clear teachings in the prior art outweigh any supposed "secondary considerations." *Graham v. John Deere Co. of Kansas City*, 383 U.S. 1, 36 (1966).

III. DISCLOSURE OF CONCURRENT LITIGATION, REEXAMINATION, AND RELATED PROCEEDINGS

Based on information available to Requester, the '302 patent is the subject of two District Court litigations, as listed below. Requester is unaware of any prior reexamination or other post-grant proceeding in which the '302 Patent is or has been involved.

- *Foras Technologies, Ltd. vs Nissan Motor Co., Ltd.*, Case No. 1:23-cv-00640 (W.D. Tex., June 6, 2023).
- *Foras Technologies, Ltd. vs Toyota Motor, Corp.*, Case No. 2:23-cv-00150 (E.D. Tex., Apr. 5, 2023).

IV. CONCLUSION

Reexamination and cancellation of claims 1-27 of the '302 patent is respectfully requested. The Commissioner is hereby authorized to charge Deposit Account 50-6990 under Docket No. 185945.012600 the Ex Parte Reexamination fee of $12,600 under 37 C.F.R. § 1.20(c)(1). Requester believes no other fee is due with this submission, however, the Commissioner is hereby authorized to charge any fee deficiency or credit any over-payment to Deposit Account 50-6990.

Please direct all correspondence in this matter to the undersigned.

Dated: September 1, 2023

Respectfully Submitted,

/Gregory Lunt/

By: Gregory Lunt
Reg. No. 57,354

GREENBERG TRAURIG, LLP
222 South Main Street, Suite 1730
Salt Lake City, UT 84101
Tel: (801) 478-6900

Counsel for Requester Unified Patents, LLC

Andrew Wolfe Ph.D.

2005 De La Cruz Blvd STE 142
Santa Clara, CA 95050
(408) 394-1096 (mobile)
Email: awolfe@awolfe.org

Education:

Ph.D. in Electrical and Computer Engineering, Carnegie Mellon University, 1992
Visiting Graduate Student, Center for Reliable Computing, Stanford University, 1988-1989
M.S. in Electrical and Computer Engineering, Carnegie Mellon University, 1987
B.S.E.E. in Electrical Engineering and Computer Science, The Johns Hopkins University, 1985

Recent Employment:

Lecturer, [September 2013-present]
Santa Clara University

Teaching graduate and undergraduate courses on embedded computing, mechatronics, real-time systems, computer architecture, design, and community service.

Consultant, [October 2002-present]
Wolfe Consulting

Consultant on processor technology, computer systems, consumer electronics, software, design tools, and intellectual property issues. Testifying and consulting expert for IP and other technology-related litigation matters.

Sample clients include:

Google	Apple	Samsung
Sony	Huawei	HTC
AT&T	Verizon Mobile	T-Mobile
IBM	Motorola/Lenovo	Nintendo
Activision	AMD	Western Digital
Sonos	Qualcomm	LG
Netflix	Roku	Tivo
Synaptics	Sawstop	3M
Medigus	TCL	ZTE
Egnyte	Acer	Waymo

Chief Technical Officer, [1999-2002]; Sr. VP of Business Development, [2001-2002]; VP, Systems Integration, S3 Fellow , [1998 – 1999]; Director of Technology, S3 Fellow , [1997 - 1998]
SONIC|blue, Inc, Santa Clara, CA (formerly S3 Inc.)

Strategic Business Development:
Developed and implemented strategy to reposition S3 from PC graphics into the leading networked consumer electronics company.

- Acquired Diamond Multimedia and coordinated integration of communications, Rio digital music, and workstation graphics divisions into S3.
- Identified and negotiated acquisitions to grow digital media businesses including Empeg, ReplayTV, and Sensory Science.
- Identified and negotiated strategic investments including Comsilica, Intellon, KBGear Interactive, Entridia, DataPlay and others.

- Developed strategy for integrated graphics/core-logic products and established a joint venture with Via Technologies to design and market these products.
- Negotiated divestiture of graphics chip business to Via and the workstation graphics division to ATI.

Product Planning and Development:
- Drove roadmap development within SONICblue product divisions.
- Managed Business Development for all product lines.
- Led New Product Development and Corporate Vision processes.
- Acting co-General Manager of Rio digital music business in 2nd half of 2001. Responsible for all areas of product development, business development, and cost management.
- Managed development of the Savage/MX and Savage/IX mobile 3D graphics accelerators and Savage/NB system logic products.

Public Relations, Public Policy and Investor Relations:
- Present company products and strategy at industry events such as CES, Comdex, and Microprocessor Forum.
- Discuss new products and initiatives with the press.
- Promote issues of interest to SONICblue to industry groups and in Washington.
- Brief analysts, and investors on company progress. Participate in quarterly conference calls.

IP Management and Licensing:
- Negotiated and managed partnership agreements including a critical cross-licensing agreement with Intel.
- Renegotiated technology-licensing agreements with IBM for workstation graphics products.
- Evaluated outside technology opportunities, managed video research and development, and managed corporate IP strategy with legal staff including patent filings, cross licensing, and litigation.

Consulting Professor , [1999-2002]
Stanford University, Stanford, CA

Teaching computer architecture and microprocessor design.

Assistant Professor [1991 - 1997]
Princeton University, Princeton, NJ

Teaching and research in the Electrical Engineering department. Research in embedded computing systems, multimedia, video signal processors, compiler optimization, and high performance computer architecture. Principal investigator or project manager for ~$6M in funded research.

Visiting Assistant Professor , [1992]
Carnegie Mellon University, Pittsburgh, PA

Research and preparation of teaching materials on advanced microprocessor designs including new superscalar and superpipelined processor architectures.

Founder and Vice President and Consultant, [1989 - 1995]
The Graphics Technology Company, Inc., Austin, TX

Founded company to develop touch-sensitive components and systems for the first generation of PDA devices and interactive public systems. Obtained financing from Gunze Corp., Osaka, Japan. Company is now part of 3M.

Senior Electrical Engineer, [1989]
ESL - TRW, Advanced Technology Division, Sunnyvale, CA

Designed the architecture for an Intel i860-based multiple-processor digital signal processing system for advanced military applications. Designed several FPGA interface chips for VME-bus systems.

Design Consultant, [1986 -1987]
Carroll Touch Division, AMP Inc., Round Rock, TX

Developed several new technologies for touch-screen systems. Designed the first ASIC produced for AMP, a mixed-signal interface chip for controlling touch-screen sensors. Developed the system electronics, system firmware, and customer utility software for numerous products including those based on the new ASIC.

Senior Design Engineer, [1983 -1985]
Touch Technology Inc., Annapolis, MD

Advisory Boards:

Director, Turtle Beach Corporation (NASDAQ:HEAR) (formerly Parametric Sound Corporation), KBGear Interactive, Inc., Comsilica, Inc., Rioport.com, various S3 subsidiaries.

Technical Advisory Boards, Ageia, Inc., Intellon, Inc., Comsilica, Inc., Entridia, Inc., Siroyan, Ltd., BOPS, Inc, Quester Venture Funds

Carnegie Mellon University Silicon Valley Advisory Board; Johns Hopkins University Tech Transfer Advisory Board

Awards:

IEEE Fellow - for contributions in hardware code compression of embedded software, power consumption analysis, and optimization, 2022
IEEE Computer Society Distinguished Contributor - 2021
Micro Test-of-Time Award (in recognition of one of the ten most influential papers of the first 25 years of the symposium), 2014
Business 2.0 "20 Young Executives You Need to Know", 2002
Walter C. Johnson Prize for Teaching Excellence, 1997.
Princeton University Engineering Council Excellence in Teaching Award, Spring 1996
AT&T/Lucent Foundation Research Award, 1996.
Walter C. Johnson Prize for Teaching Excellence, 1995
IEEE Certificate of Appreciation, 1995, 2001.
AT&T Foundation Research Award, 1993.
Semiconductor Research Corporation Fellow, 1986 - 1991.
Burroughs Corporation Fellowship in Engineering, 1985 - 1986.

Professional Activities:

Program Chair: Micro-24, 1991, Hot Chips 13, 2001.

General Chair: Micro-26, 1993, Micro-33, 2000.

Associate Editor: IEEE Computer Architecture Letters; ACM Transactions in Embedded Computing Systems

Speaker at CES, WinHec, Comdex, Intel Dev. Forum, Digital Media Summit, Microprocessor Forum, etc.

Keynote speaker at Micro-34, ICME 2002

IEEE B. Ramakrishna Rau Award committee – 2012-2016

IEEE Computer Society Awards Committee – 2015

CES Awards Judge – 2016

Entrepreneurship Mentor – Draper University

Over 50 refereed publications.

Publications since January 2006:

Wolfe, A., "Retrospective on Code Compression and a Fresh Approach to Embedded Systems", IEEE MICRO, July/Aug. 2016, Invited paper.

Patents:

U.S. Pat. 5,041,701 – *Edge Linearization Device for a Contact Input System*, Aug. 20, 1991.
U.S. Pat. 5,438,168 – *Touch Panel*, Aug. 1, 1995.
U.S. Pat. 5,736,688 – *Curvilinear Linearization Device for Touch Systems*, Apr. 7, 1998.
U.S. Pat. 6,037,930 – *Multimodal touch sensitive peripheral device*, March 14, 2000.
U.S. Pat. 6,408,421 – *High-speed asynchronous decoder circuit for variable-length coded data*, June 18, 2002.
U.S. Pat. 6,865,668 – *Variable-length, high-speed, asynchronous decoder circuit*, March 8, 2005
U.S. Pat. 7,079,133 – *Superscalar 3D Graphics Engine*, July 18, 2006
EP 1 661 131 B1 – *PORTABLE ENTERTAINMENT APPARATUS*, Jan. 21, 2009
U.S. Pat. 7,555,006 – *Method and system for adaptive transcoding and transrating in a video network*, June 30, 2009
U.S. Pat. 7,996,595 – *Interrupt Arbitration for Multiprocessors*, Aug. 9, 2011
EP 2 241 979 B1 – *Interrupt Arbitration for Multiprocessors*, Oct. 10, 2011
U.S. Pat. 8,131,970 – *Compiler Based Cache Allocation*, March 6, 2012
U.S. Pat. 8,180,963 – *Hierarchical read-combining local memories*, May 15, 2012
U.S. Pat. 8,193,941 – *Snoring Treatment*, June 5, 2012
U.S. Pat. 8,203,541 – *OLED display and sensor*, June 19, 2012
U.S. Pat. 8,243,045 – *Touch-sensitive display device and method*, August 14, 2012
U.S. Pat. 8,244,982 – *Allocating processor cores with cache memory associativity*, August 14, 2012
U.S. Pat. 8,260,996 – *Interrupt Optimization for Multiprocessors*, Sept. 4, 2012
101185761 (KR) – *Noise Cancellation for Phone Conversation,* Sept. 19, 2012
101200740 (KR) – *OLED display and sensor*, November 7, 2012
101200741 (KR) – T*ouch-sensitive display device and method*, November 7, 2012
U.S. Pat. 8,321,614 – *Dynamic scheduling interrupt controller for multiprocessors*, Nov. 27, 2012
U.S. Pat. 8,352,679 – *Selectively securing data and/or erasing secure data caches responsive to security compromising conditions*, Jan. 8, 2013
U.S. Pat. 8,355,541 – *Texture Sensing*, Jan. 15, 2013
U.S. Pat. 8,370,307 – *Cloud Data Backup Storage Manager*, Feb. 5, 2013
U.S. Pat. 8,398,451 – *Tactile Input Interaction*, March. 19, 2013
JP 5241032 B2 – *Routing Across Multicore Network Using Real World or Modeled Data*, April 13, 2013
ZL201010124820.3 – *Interrupt Optimization for Multiprocessors,* April 17, 2013
U.S. Pat. 8,428,438 – *Apparatus for Viewing Television with Pause Capability*, April 23, 2013
JP 5266197 B2 – *Data Centers Task Mapping,* May 10, 2013
U.S. Pat. 8,508,498 – *Direction and Force Sensing Input Device*, August 13, 2013
U.S. Pat. 8,547,457 – *Camera Flash Mitigation*, October 1, 2013
U.S. Pat. 8,549,339 – *Processor core communication in multi-core processor*, October 1, 2013
101319048 (KR) – *Camera Flash Mitigation*, October 10, 2013
U.S. Pat. 8,628,478 – *Microphone for remote health sensing*, January 14, 2014
101362017 (KR) – *Thread Shift: Allocating Threads to Cores*, Feb. 5, 2014
101361928 (KR) – *Cache Prefill on Thread Migration*, Feb. 5, 2014
101361945 (KR) – *Mapping Of Computer Threads onto Heterogeneous Resources*, Feb. 5, 2014
JP 5487307 B2 – *Mapping Of Computer Threads onto Heterogeneous Resources*, Feb. 28, 2014
JP 5484580 B2 – *Task Scheduling Based on Financial Impact,* Feb. 28, 2014
JP 5487306 B2 – *Cache Prefill on Thread Migration*, Feb. 28, 2014
101372623 (KR) – *Power Management for Processor*, March. 4, 2014
101373925 (KR) – *Allocating Processor Cores with Cache Memory Associativity*, March 6, 2014
U.S. Pat. 8,676,668 – *Method for the determination of a time, location, and quantity of goods to be made available based on mapped population activity*, March 18, 2014
U.S. Pat. 8,687,533 – *Energy Reservation in Power Limited Networks*, April 1, 2014
101388735 (KR) – *Routing Across Multicore Networks Using Real World or Modeled Data*, April 17, 2014
U.S. Pat. 8,725,697 – *Cloud Data Backup Storage*, May 13, 2014
U.S. Pat. 8,726,043 – *Securing Backing Storage Data Passed Through a Network*, May 13, 2014
ZL201010124826.0 – *Dynamic scheduling interrupt controller for multiprocessors*, May 14, 2014
JP 5547820 B2 – *Processor core communication in multi-core processor*, May 23, 2014
U.S. Pat. 8,738,949 – *Power Management for Processor*, May 27, 2014

U.S. Pat. 8,751,854 – *Processor Core Clock Rate Selection*, June 10, 2014
JP 5559891 B2 – *Thermal Management in Multi-Core Processor*, June 13, 2014
101414033 (KR) – *Dynamic Computation Allocation*, June 25, 2014
JP 5571184 B2 – *Dynamic Computation Allocation*, July 4, 2014
101426341 (KR) – *Processor core communication in multi-core processor*, May 23, 2014
U.S. Pat. 8,799,671 – *Techniques for Detecting Encrypted Data*, Aug 5, 2014
101433485 (KR) – *Task Scheduling Based on Financial Impact,* Aug. 18, 2014
U.S. Pat. 8,824,666 – *Noise Cancellation for Phone Conversation,* Sept. 2, 2014
U.S. Pat. 8,836,516 – *Snoring Treatment*, Sept. 16, 2014
U.S. Pat. 8,838,370 – *Traffic flow model to provide traffic flow information*, Sept. 16, 2014
U.S. Pat. 8,838,797 – *Dynamic Computation Allocation*, Sept. 16, 2014
U.S. Pat. 8,854,379 – *Routing Across Multicore Networks Using Real World or Modeled Data*, Oct. 7, 2014
JP 5615361 B2 – *Thread Shift: Allocating Threads to Cores*, Oct. 15, 2014
U.S. Pat. 8,866,621 – *Sudden infant death prevention clothing*, Oct. 21, 2014
U.S. Pat. 8,881,157 – *Allocating threads to cores based on threads falling behind threads*, Nov. 4, 2014
ZL201080024755.5 – *Camera Flash Mitigation*, Nov 5, 2014
U.S. Pat. 8,882,677 – *Microphone for remote health sensing*, Nov. 11, 2014
U.S. Pat. 8,924,743 – *Securing Data Cache through Encryption*, December 30, 2014
U.S. Pat. 8,994,857 – *Camera Flash Mitigation*, March 31, 2015
JP 5699140 B2 – *Camera Flash Mitigation*, April 8, 2015
ZL201080035189.8 – *Thread Shift: Allocating Threads to Cores*, June 10, 2015
ZL201180005030.6 – *Processor core communication in multi-core processor*, June 10, 2015
U.S. Pat. 9,143,814 – *Method and system for adaptive transcoding and transrating in a video network*, Sept 22, 2015
ZL201080035177.5 – *Mapping Of Computer Threads onto Heterogeneous Resources*, Oct. 14, 2015
U.S. Pat. 9,178,694 – *Securing Backing Storage Data Passed Through a Network*, November 3, 2015
U.S. Pat. 9,189,282 – *Thread-to-core mapping based on thread deadline, thread demand, and hardware characteristics data collected by a performance counter*, November 17, 2015
U.S. Pat. 9,189,448 – *Routing image data across on-chip networks*, November 17, 2015
U.S. Pat. 9,208,093 – *Allocation of memory space to individual processor cores*, December 8, 2015
U.S. Pat. 9,239,994 – *Data Centers Task Mapping,* January 19, 2016
ZL201080036611.1 – *Allocating Processor Cores with Cache Memory Associativity*, January 20, 2016
EP2228779 B1 – *Traffic flow model to provide traffic flow information*, Jan. 27, 2016
U.S. Pat. 9,262,628 – *Operating System Sandbox,* February 16, 2016
GB2485682 – *Mapping Of Computer Threads onto Heterogeneous Resources*, Sept. 28, 2016
U.S. Pat. 9,330,137 – *Cloud Data Backup Storage Manager*, May. 3, 2016
ZL201080035185.X – *Cache Prefill on Thread Migration*, Aug. 24, 2016
U.S. Pat. 9,519,305 – *Processor Core Clock Rate Selection*, December 13, 2016
U.S. Pat. 9,569,270 – *Mapping thread phases onto heterogeneous cores based on execution characteristics and cache line eviction count*, February 14, 2017
GB2485683 – *Thread Shift: Allocating Threads to Cores*, Oct. 18, 2017
U.S. Pat. 9,852,435 – *Telemetrics based location and tracking*, December 26, 2017.
U.S. Pat. 9,915,994 – *Power management for processor*, March 13, 2018
U.S. Pat. 9,927,254 – *Traffic flow model to provide traffic flow information*, March 27, 2018
EP2254048 B1 – *Thread Mapping in Multi-Core Processors*, August 29, 2018
U.S. Pat. 10,860,432 – *Cloud Data Backup Storage Manager*, December 8, 2020

Expert testimony by deposition or at trial – April 15, 2018 - -present

Case	Venue	Case Number
Joe Andrew Salazar v HTC Corporation	E.D. Texas	2:16-cv-010986-JRG-RSP
Hitachi Maxell, Ltd. v. ZTE Corporation and ZTE (USA), Inc.	E.D. Texas	5:16-cv-00179 5:16-cv-00178
INTER PARTES REVIEW OF U.S. PATENT NO. 7,663,506 (Mediatek v AMD)	PTAB	IPR2017-00101 IPR2017-00102
Papst Licensing GmbH & Co. KG v. Samsung Electronics Co.,Ltd. and Samsung Electronics America, Inc.	E.D. Texas	6:15-CV-1102
HTC Corporation v. Telefonaktiebolaget LM Ericsson	E.D. Texas	6: 18-cv-00243-JRG
Seven Networks, LLC v ZTE (USA) Inc and ZTE Corporation	N. D. Texas - Dallas	3:17-CV-1495
AGIS Software Development, LLC v. HTC Corporation	E. D. Texas	2:17-cv-514
Barbaro Technologies, LLC v. Niantic, Inc	N.D. CA	3:18-cv-02955-RS
Immersion Corp. v. Samsung Electronics America, Inc. et al	E.D. Texas	2:17-cv-00572
INTER PARTES REVIEW OF U.S. PATENT NO. 7,171,526 (Northstar Innovations – Micron)	PTAB	IPR2018-01004 IPR2018-01005
CISCO SYSTEMS, INC., vs. UNILOC USA, INC., UNILOC 2017 LLC and UNILOC LICENSING USA LLC	N.D. CA	3:18-cv-04991-SI
INTER PARTES REVIEW OF U.S. PATENT NO. 8,020,014 (Intel/VLSI)	PTAB	IPR2018-01661 IPR2018-01312
INTER PARTES REVIEW OF U.S. PATENT NO Patent 9,294,799 (Comcast/Rovi)	PTAB	IPR2019-00299
Inter Partes Review of U.S. Patent No 7720929 (Unified Patents v Datascape)	PTAB	IPR2019-01115
Solas OLED Ltd,, v. Samsung Display Co,, Ltd,, Samsung Electronics Co, Ltd,, and Samsung Electronics America, Inc,,	PTAB	IPR2019-01668
INTER PARTES REVIEW OF U.S. PATENT NO Patent 8,973,069 (Comcast/Rovi)	PTAB	IPR2019-01434
U.S. Patent No. 8,448,215 IPR (Comcast v Rovi)	PTAB	IPR2019-01353
U.S. PATENT NO. 8,847,898 IPR filing (Samsung v Neodron)	PTAB	IPR2020-00234
U.S. PATENT NO. 8,610,009 IPR filing (Samsung v Neodron)	PTAB	IPR2020-00225
U.S. PATENT NO. 10,365,747 IPR filing (Samsung v Neodron)	PTAB	IPR2020-00308
AGIS Software Development LLC v. Google LLC, AGIS Software Development LLC v. WAZE Mobile Limited,and AGIS Software Development LLC v. Samsung Elecs. Co., Ltd. et al	E.D. Texas	2:19-cv-00361-JRG 2:19-cv-00359-JRG 2:19-cv-00362-JRG
Inter Partes Review of U.S. Patent No 8,112,670 (Sony)	PTAB	IPR2020-00726
U.S. Patent 8,819,505 IPR (Intel v PACT)	PTAB	IPR2020-00525
Inter Partes Review of U.S. Patent No 8,078,540 (Sony)	PTAB	IPR2020-00922
Inter Partes Review of U.S. Patent No 9,037,807 (Intel v PACT)	PTAB	IPR2020-00540

HONG KONG UCLOUDLINK NETWORK TECHNOLOGY LIMITED AND UCLOUDLINK (AMERICA), LTD., vs. SIMO HOLDINGS INC. AND SKYROAM, INC.,	N.D. CA	3:18-cv-05031-EMC
Joe Andrew Salazar v AT&T Mobility, LLC et al.	E.D. Texas	2:20-cv-00004-JRG
Innovative Memory Solutions Inc. v Micron, Inc	Delaware	14-1480 RGA
Inter Partes Review of U.S. Patent No. 6,411,941	PTAB	IPR2021-01338 IPR2021-01406
Maxell, Ltd. et al v. VIZIO, Inc	C.D. CA	2-21-cv-06758
Inter Partes Review of U.S. Patent No. 7,619,912	PTAB	IPR2022-00615
Inter Partes Review of U.S. Patent Nos. 11,016,918 and 11,232,054	PTAB	IPR2022-00996 IPR2022-00999
In the Matter of Certain Smart Televisions, - Maxell, Ltd. et al v. VIZIO, Inc	ITC	337-TA-1338
INTER PARTES REVIEW OF U.S. PATENT NOS. 8,787,060 and 9,318,160	PTAB	IPR2022-01427 IPR2022-01428

UNITED STATES
PATENT AND TRADEMARK OFFICE

P.O. Box 1450
Alexandria, VA 22313 - 1450
www.uspto.gov

ELECTRONIC ACKNOWLEDGEMENT RECEIPT

APPLICATION #	RECEIPT DATE / TIME	ATTORNEY DOCKET #
90/019,243	**09/01/2023 06:38:04 PM ET**	**185945.012600**

Title of Invention

SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

Application Information

APPLICATION TYPE	Utility / ex parte reexam	PATENT #	7624302
CONFIRMATION #	1079	FILED BY	Judi Cordo
PATENT CENTER #	62727465	FILING DATE	-
CUSTOMER #	56436	FIRST NAMED INVENTOR	Scott L. Michaelis
CORRESPONDENCE ADDRESS	-	AUTHORIZED BY	Andrew Sommer

Documents

TOTAL DOCUMENTS: 29

DOCUMENT	PAGES	DESCRIPTION	SIZE (KB)
185945_012600_Ex_CC_Appendix_C.PDF	5	Reexam - Affidavit/Declaration/Exhibit Filed by 3rd Party	184 KB
185945_012600_Ex_AA_Appendix_A.PDF	11	Reexam - Affidavit/Declaration/Exhibit Filed by 3rd Party	239 KB
185945_012600_Ex_BB_Appendix_B.PDF	15	Reexam - Affidavit/Declaration/Exhibit Filed by 3rd Party	248 KB
185945_012600_Requesters_POA.PDF	1	Power of Attorney	106 KB

185945_012600_Transmittal_Form.pdf	4	Reexam Miscellaneous Incoming Letter	413 KB
185945_012600_Ex_1017_Marshall_US5915082.PDF	16	Reexam - Affidavit/Declaration/Exhibit Filed by 3rd Party	1083 KB
185945_012600_Certificate_of_ServiceUS7624302.PDF	3	Reexam Certificate of Service	214 KB
185945_012600_Ex_1015_The_958_IA-64_Manual.PDF	6	Reexam - Affidavit/Declaration/Exhibit Filed by 3rd Party	3032 KB
185945.012600_Transmittal_Form.PDF	4	Reexam Miscellaneous Incoming Letter	404 KB
185945_012600__IDS.PDF	5	Information Disclosure Statement (IDS) Form (SB08)	599 KB
185945_012600_Copy_of_Patent_for_Which_Reexamination_is_Requested.PDF	18	Reexam Miscellaneous Incoming Letter	2146 KB
185945_012600_Ex_1003_US7624302_Reexam_Declaration.PDF	67	Reexam - Affidavit/Declaration/Exhibit Filed by 3rd Party	586 KB
185945_012600_Reexam_Request_US7624302.PDF	79	Reexam Miscellaneous Incoming Letter	995 KB
185945_012600_Ex_1001_Michaelis_US7624302.PDF	18	Reexam - Affidavit/Declaration/Exhibit Filed by 3rd Party	2146 KB
185945_012600_Ex_1004_Nguyen_US20040123201A1.PDF	16	Reexam - Affidavit/Declaration/Exhibit Filed by 3rd Party	1123 KB
185945_012600_Ex_1007_S	8	Reexam -	556 KB

afford_US7085959.PDF		Affidavit/Declaration/Exhibit Filed by 3rd Party	
185945_012600_Ex_1005_Kalyanasundharam_US7328371.PDF	17	Reexam - Affidavit/Declaration/Exhibit Filed by 3rd Party	1630 KB
185945_012600_Ex_1006_Kennedy_US5450576.PDF	14	Reexam - Affidavit/Declaration/Exhibit Filed by 3rd Party	1655 KB
185945_012600_Ex_1009_Fox_US7251749.PDF	17	Reexam - Affidavit/Declaration/Exhibit Filed by 3rd Party	1122 KB
185945_012600_Ex_1008_Quach_US6625749.PDF	19	Reexam - Affidavit/Declaration/Exhibit Filed by 3rd Party	1575 KB
185945_012600_Ex_1010_Commercial_Fault_Tolerance.PDF	10	Reexam - Affidavit/Declaration/Exhibit Filed by 3rd Party	799 KB
185945_012600_Ex_1013_Unix_Administration_Guide.PDF	11	Reexam - Affidavit/Declaration/Exhibit Filed by 3rd Party	180 KB
185945_012600_Ex_1012_Hobson_US7191328.PDF	9	Reexam - Affidavit/Declaration/Exhibit Filed by 3rd Party	633 KB
185945_012600_Ex_1014_Cepulis_US5491788.PDF	23	Reexam - Affidavit/Declaration/Exhibit Filed by 3rd Party	1720 KB
185945_012600_Ex_1011_Intel_EFI_Specification.PDF	8	Reexam - Affidavit/Declaration/Exhibit Filed by 3rd Party	4847 KB
185945_012600_Ex_1002_U	310	Reexam -	12731 KB

S_File_History.PDF		Affidavit/Declaration/Exhibit Filed by 3rd Party	
185945_012600_Ex_1016_Intel_Multiprocessor_Specification_1_of_2.pdf	48	Reexam - Affidavit/Declaration/Exhibit Filed by 3rd Party	19306 KB
185945_012600_Ex_1016_Intel_Multiprocessor_Specification_2_of_2.PDF	49	Reexam - Affidavit/Declaration/Exhibit Filed by 3rd Party	22164 KB
185945_012600_Declaration_Appendix_A.PDF	8	Reexam - Affidavit/Declaration/Exhibit Filed by 3rd Party	185 KB

Digest

DOCUMENT	MESSAGE DIGEST(SHA-512)
185945_012600_Ex_CC_Appendix_C.PDF	CE000757F240BFB6D6D919A8049CA1C118DEE5D8F376A3DEF6F0D64886FBBDF8E015A96973E044BACBB803DD92DDFD825C2082A729B8A25AE248AA9D9E77E3D7
185945_012600_Ex_AA_Appendix_A.PDF	74A10592B4CB895B1701BC9C86EF9535BBF01A049DF60500A53D43F162E4BE541FEEA8E31C950975A1ECD63CCFF03157A99870E5ACA6A73A7E8C4DFE95A62064
185945_012600_Ex_BB_Appendix_B.PDF	0D790880FD167988860E14864A85F87653DD798D1F64220BC6817FE6961D52DE6EEAC83F2318107A285AF94C3483C900A62FA604407A1B8441FF299F0B1C385A
185945_012600_Requesters_POA.PDF	3220D0D28956FFEF9DFACFC4D9784A162DA6AF04F9E5255914BA49D7B4A27F61E5D8A31C140B5A5784BF2C356C91A472600224EA600E201BFBBB2EF9DE700937
185945_012600_Transmittal_Form.pdf	A15666C94DDA171D2B07729137D6B415DFE12C882B700D0A27AAFD4CACD361F28B21DFD310EDD5F2D32E09381F2AF7EFCD14BEDABECB970EB0B67F458E678BF7

185945_012600_Ex_1017_Marshall_US5915082.PDF	6B49F337E59E720D18388464C741843677D15451D2EAA036D58B55081798359F4298D98C63E799454B315E37810273300F89B8E3BC83548C751F9B90F4A04598
185945_012600_Certificate_of_ServiceUS7624302.PDF	1212A018FCA5DB78FC9446CF427E42CA483ABFE0232DB566245DA189BD95A08A203B02763B2C6F13D07EC5B81AE561AC5E88410E417C99362064B0F0FD02893D
185945_012600_Ex_1015_The_958_IA-64_Manual.PDF	BBB5BF7CADE5CEB892456BBEF9A60767A0612B6BC68A528C8313F2426E2DF4E322552F27FED4A5BC9B21D4C444BD1F7D5358303AC49E81B1A372428D25D61864
185945.012600_Transmittal_Form.PDF	7EB0A7878546C4AB4261792CDDE9CB7BD9BE6007351AB8AC0F82CE1E1545026ECDD11303F6A7D51B429BF62759EBC5823E6F977F123110B4C4B4C9DD9B6DADD1
185945_012600__IDS.PDF	15ED75DA7FBD4AD0240B39E9342198FDACA72467DA82366C4CA85D2AE30022EB0C2773C76BF031687AC6CD214CBB83D08B27A7CF8493EF977559C0819B2FFBAE
185945_012600_Copy_of_Patent_for_Which_Reexamination_is_Requested.PDF	4235F9002344BEEB943177BAA334123E62AACEC630990DB2A416981BFAADF54EB1EEEE648731D8ECD8D3D6133DE68025C93BAA63667F9017CB2EDAFEC3EC2F31
185945_012600_Ex_1003_US7624302_Reexam_Declaration.PDF	002EB0A4F68FA47F6BF8E963E7A0CDFA05464448DEA342AE60D771A1E1623F4AC264B5D1D3C77EA5E51734B5BF00CBF2E4FC063DEA0F3138CBBC2005BEE0B787
185945_012600_Reexam_Request_US7624302.PDF	0CA7F08E9DE433ADF046B06FD39788D33F4A70CFC7CEB617A9668200E60E7BC2FA17EA2A6DB31D3C02E1C7A84FF7F1F925A5B677897A00949B3157AEE64D6DA9
185945_012600_Ex_1001_Michaelis_US7624302.PDF	4235F9002344BEEB943177BAA334123E62AACEC630990DB2A416981BFAADF54EB1EEEE648731D8ECD8D3D6133DE68025C93BAA63667F9017CB2EDAFEC3EC2F31
185945_012600_Ex_1004_Nguyen_US20040123201A1.PDF	F45238816130024992B378EFE178E6D76D9F6031B5B552B627E21E5738212F2AA5F24A7E1D10C2F34930EA9E11F6725B4DD680218774BD0D1AED6E3437948F1B

185945_012600_Ex_1007_Safford_US7085959.PDF	974398271157DA02029F407327C8E28B1E813A590CB77A7A521CCD6D71C870744C59944CADD11C57C0BC49FD32BCBA8B7BFDDE1573CEA40FCD19D9343DC5AB22
185945_012600_Ex_1005_Kalyanasundharam_US7328371.PDF	6864A8FE34C09A0FE32B626BCC3197E16FF6C57FD915B62516DA68EA245A8449E5930961D0C42E097754EB19412EA9335E8CF83BAE5385BF43B1272E6DB1989
185945_012600_Ex_1006_Kennedy_US5450576.PDF	788315FB04066F273C70FE865DB0AE8AAFA8B81A8FE86AE5C5481AEE5A21B4F0AC07A3E8821612130A4E96CF398F139543205D34C09E7653CEE3A0FEEF0F5600
185945_012600_Ex_1009_Fox_US7251749.PDF	0F8936CFB0914C293E3BCE9D1E92A3C8559E6E25DFE52DFC69AD900CDAD191B2D634A20BEAE64C77EE31AB225DB0BB72E96E02AA5BAF92DAE70CEA3F70B6176F
185945_012600_Ex_1008_Quach_US6625749.PDF	86607E864C4D313CA044700E61594436E59FEC1AEDFAFC5D71534ED83D2DFEB4ADF79D939EBC172049951CE3E13574B1B8BA8A759E94027D5983285492C437D1
185945_012600_Ex_1010_Commercial_Fault_Tolerance.PDF	78CD06DDE7CA2CF7B193401F790503366D13410BC8375E322249156C9426F775243B6899430A91492503BB27573C9A5DB5B99D38DF23C7ED2AA725E4FE0A0DE3
185945_012600_Ex_1013_Unix_Administration_Guide.PDF	DDAE0D98E3CB3AD9301FAB3496041D6089544448BC89F65303CD04F31EB408E4563CC0CA3244398D5BE2E339F750C34A4D735A4E37E7113CD68263B1487D5420
185945_012600_Ex_1012_Hobson_US7191328.PDF	DD2A7C594D02FA76E0F2FA1A991DC8D41B14AD543C4C34D43ACAF2EF412B6E4E0127CD59FE11B60AEB86E9E7966E1B8894A369BEC6238728AB205302B1374A6A
185945_012600_Ex_1014_Cepulis_US5491788.PDF	2E515767F4888B5E145B291D6B69EEBA65C22668A3993FC7046B6C79E3B146F3BFBA7F9EDD6DE6522A51FD12384763D122DACAFCFD1A59BDE631E6706715B620
185945_012600_Ex_1011_Intel_EFI_Specification.PDF	824D1428E9B6D293095296CA4727A57E9FD072816CC9D1493161385514F86E9F0D93330D84B8D444463DD01B11CA9C5AD10EF4D273E7E61A638B7945E65618BA

185945_012600_Ex_1002_US_File_History.PDF	1D11A6A690C33C59F853E208EA2CE937BA3C008C297CF8EAF8B11B47442B94B5C18ADB0BB9DB2A27FC6758A950EFC85F12F329620603194512E8D29B9ABD443B
185945_012600_Ex_1016_Intel_Multiprocessor_Specification_1_of_2.pdf	BBB3A0C062A626C027BA181D1F57EEBE570868E2DB810717040A1A209B374DAC276F729853F974356985D3657E07C0177F51BFD200E9CA6630B803DC9B61F2D7
185945_012600_Ex_1016_Intel_Multiprocessor_Specification_2_of_2.PDF	B5422504042075225758B5296E52173E46854A890A28D1F02C115D691D0C36A05EA8823846E1356D38C60987174CF75B48E284A9AC7B7F7B2A86DDE6D40F3718
185945_012600_Declaration_Appendix_A.PDF	C5F55D877944119E50B66E283A0210DD8ABD75686EB2352EFED9CF883B942E0111ABE34F5B03DC29B3EB006572065E7553FEDDE7BDB10B1EE44E8183A9C8BB9F

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

BIB DATA SHEET

CONFIRMATION NO. 1079

SERIAL NUMBER	FILING or 371(c) DATE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.
90/019,243	09/01/2023 RULE	714	3992	185945.012600

APPLICANTS

L.P. HEWLETT-PACKARD DEVELOPMENT COMPANY, Residence Not Provided;
LLC UNIFIED PATENTS, CHEVY CHASE, MD;

INVENTORS

7624302, Residence Not Provided;
GREENBERG TRAURIG (NJ), Residence Not Provided;

**** CONTINUING DATA ***************************

This application is a REX of 10/972,588 10/25/2004 PAT 7624302

**** FOREIGN APPLICATIONS ***************************

**** IF REQUIRED, FOREIGN FILING LICENSE GRANTED ****

Foreign Priority claimed ❑ Yes ❑ No 35 USC 119(a-d) conditions met ❑ Yes ❑ No Verified and Acknowledged _____ Examiner's Signature	❑ Met after Allowance _____ Initials	STATE OR COUNTRY	SHEETS DRAWINGS	TOTAL CLAIMS	INDEPENDENT CLAIMS
				27	3

ADDRESS

Hewlett Packard Enterprise
3404 E. Harmony Road
Mail Stop 79
Fort Collins, CO 80528
UNITED STATES

TITLE

SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

FILING FEE RECEIVED		
0.00	FEES: Authority has been given in Paper No._____ to charge/credit DEPOSIT ACCOUNT No._____ for following:	❑ All Fees ❑ 1.16 Fees (Filing) ❑ 1.17 Fees (Processing Ext. of time) ❑ 1.18 Fees (Issue) ❑ Other _____ ❑ Credit

BIB (Rev. 05/07).

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

REEXAM CONTROL NUMBER	FILING OR 371 (c) DATE	PATENT NUMBER
90/019,243	09/01/2023	

CONFIRMATION NO. 1079
REEXAMINATION REQUEST NOTICE

GREENBERG TRAURIG (NJ)
500 CAMPUS DRIVE, SUITE 400
P.O. BOX 677
FLORHAM PARK, NEW JERSEY 07932



NOTICE OF REEXAMINATION REQUEST FILING DATE

(Third Party Requester)

Requester is hereby notified that the filing date of the request for reexamination is 09/01/2023, the date that the filing requirements of 37 CFR § 1.510 were received.

A decision on the request for reexamination will be mailed within three months from the filing date of the request for reexamination. (See 37 CFR 1.515(a)).

A copy of the Notice is being sent to the person identified by the requester as the patent owner. Further patent owner correspondence will be the latest attorney or agent of record in the patent file. (See 37 CFR 1.33). Any paper filed should include a reference to the present request for reexamination (by Reexamination Control Number).

cc: Patent Owner
56436
Hewlett Packard Enterprise
3404 E. Harmony Road
Mail Stop 79
Fort Collins, CO 80528

/pflewis/

Legal Instruments Examiner
Central Reexamination Unit 571-272-7705; FAX No. 571-273-9900

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

REEXAM CONTROL NUMBER	FILING OR 371 (c) DATE	PATENT NUMBER
90/019,243	09/01/2023	

CONFIRMATION NO. 1079

REEXAM ASSIGNMENT NOTICE

56436
Hewlett Packard Enterprise
3404 E. Harmony Road
Mail Stop 79
Fort Collins, CO 80528

"OC0000006165525"

Date Mailed: 09/11/2023

NOTICE OF ASSIGNMENT OF REEXAMINATION REQUEST

The above-identified request for reexamination has been assigned to Art Unit 3992. All future correspondence to the proceeding should be identified by the control number listed above and directed to the assigned Art Unit.

A copy of this Notice is being sent to the latest attorney or agent of record in the patent file or to all owners of record. (See 37 CFR 1.33(c)). If the addressee is not, or does not represent, the current owner, he or she is required to forward all communications regarding this proceeding to the current owner(s). An attorney or agent receiving this communication who does not represent the current owner(s) may wish to seek to withdraw pursuant to 37 CFR 1.36 in order to avoid receiving future communications. If the address of the current owner(s) is unknown, this communication should be returned within the request to withdraw pursuant to Section 1.36.

NOTICE OF USPTO EX PARTE REEXAMINATION PATENT OWNER STATEMENT WAIVER PROGRAM

The USPTO has implemented a pilot program where, after a reexamination proceeding has been granted a filing date and before the examiner begins his or her review, the patent owner may orally waive the right to file a patent owner's statement. See *"Pilot Program for Waiver of Patent Owner's Statement in Ex Parte Reexamination Proceedings,"* 75 FR 47269 (August 5, 2010). One goal of the pilot program is to reduce the pendency of reexamination proceedings and improve the efficiency of the reexamination process.

Ordinarily when ex parte reexamination is ordered, the USPTO must wait until after the receipt of the patent owner's statement and the third party requester's reply, or after the expiration of the time period for filing the statement and reply (a period that can be as long as 5 to 6 months), before mailing a first determination of patentability. The USPTO's first determination of patentability is usually a first Office action on the merits or a Notice of Intent to Issue Reexamination Certificate (NIRC).

Under the pilot program, the patent owner's oral waiver allows the USPTO to act on the first determination of patentability immediately after determining that reexamination will be ordered, and in a suitable case issue the reexamination order and the first determination of patentability (which could be a NIRC if the claims under reexamination are confirmed) at the same time.

Benefits to the Patent Owner for participating in this pilot program include reduction in pendency.

To participate in this pilot program, Patent Owners may contact the USPTO's Central Reexamination Unit (CRU) at 571-272-7705. The USPTO will make the oral waiver of record in the reexamination file in an interview summary and a copy will be mailed to the patent owner and any third party requester.

cc: Third Party Requester(if any)
GREENBERG TRAURIG (NJ)
500 CAMPUS DRIVE, SUITE 400
P.O. BOX 677
FLORHAM PARK, NEW JERSEY 07932

/pflewis/

Legal Instruments Examiner
Central Reexamination Unit 571-272-7705; FAX No. 571-273-9900

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
90/019,243	09/01/2023	7624302	185945.012600	1079

56436 7590 09/13/2023

Hewlett Packard Enterprise
3404 E. Harmony Road
Mail Stop 79
Fort Collins, CO 80528

EXAMINER	
TON, MY TRANG	

ART UNIT	PAPER NUMBER
3992	

MAIL DATE	DELIVERY MODE
09/13/2023	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

DO NOT USE IN PALM PRINTER

(THIRD PARTY REQUESTER'S CORRESPONDENCE ADDRESS)

GREENBERG TRAURIG (NJ)
500 CAMPUS DRIVE, SUITE 400
P.O. BOX 677
FLORHAM PARK NJ 07932

EX PARTE REEXAMINATION COMMUNICATION TRANSMITTAL FORM

REEXAMINATION CONTROL NO. *90/019,243* .

PATENT UNDER REEXAMINATION *7,624,302* .

ART UNIT *3992* .

Enclosed is a copy of the latest communication from the United States Patent and Trademark Office in the above identified *ex parte* reexamination proceeding (37 CFR 1.550(f)).

Where this copy is supplied after the reply by requester, 37 CFR 1.535, or the time for filing a reply has passed, no submission on behalf of the *ex parte* reexamination requester will be acknowledged or considered (37 CFR 1.550(g)).

Ex Parte Reexamination Interview Summary – Pilot Program for Waiver of Patent Owner's Statement	Control No. 90/019,243	Patent Under Reexamination is Requested 7,624,302	
	Examiner TON, MY-TRANG	Art Unit 3992	AIA (FITF) Status No

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address. --

All participants (USPTO official and patent owner):

(1) Hewlett Packard Enterprise (3) _____

(2) Renee Preston (4) _____

Date of Telephonic Interview: 11 September 2023.

A. The USPTO official requested waiver of the patent owner's statement pursuant to the pilot program for waiver of patent owner's statement in *ex parte* reexamination proceedings.*

☐ The patent owner **agreed** to waive its right to file a patent owner's statement under 35 U.S.C. 304 in the event reexamination is ordered for the above-identified patent.

☐ The patent owner **did not agree** to waive its right to file a patent owner's statement under 35 U.S.C. 304 at this time.

☑ USPTO personnel were unable to reach the patent owner.**

B. The Patent Owner of record telephoned the Office and indicated they would like to participate in the pilot program for waiver of patent owner's statement in *ex parte* reexamination proceedings.*

☐ The Patent owner of record telephoned the Office and **agreed** to waive its right to file a patent owner's statement under 35 U.S.C. 304 in the event reexamination is ordered for the above-identified patent.

The patent owner is not required to file a written statement of this telephone communication under 37 CFR 1.560(b) or otherwise. However, any disagreement as to this interview summary must be brought to the immediate attention of the USPTO, and no later than one month from the mailing date of this interview summary. Extensions of time are governed by 37 CFR 1.550(c).

*For more information regarding this pilot program, see *Pilot Program for Waiver of Patent Owner's Statement in Ex Parte Reexamination Proceedings, 75 Fed. Reg.* 47269 (August 5, 2010), available on the USPTO Web site at http://www.uspto.gov/patents/law/notices/2010.jsp.

**The patent owner may contact the USPTO personnel at (571) 272-7705 or at the telephone number provided below if the patent owner decides to waive the right to file a patent owner's statement under 35 U.S.C. 304.

/RENEE M PRESTON/
CRU Paralegal, OCRU (571)272-1607
Signature and telephone number of the USPTO official, who contacted, was contacted by, or attempted to contact the patent owner.

cc: Requester (if third party requester)

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

BIB DATA SHEET

CONFIRMATION NO. 1079

SERIAL NUMBER	FILING or 371(c) DATE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.
90/019,243	09/01/2023 RULE	714	3992	185945.012600

APPLICANTS

FORAS TECHNOLOGIES LTD(OWNER), Residence Not Provided;
LLC UNIFIED PATENTS, CHEVY CHASE, MD;

INVENTORS

7624302, Residence Not Provided;
GREENBERG TRAURIG (NJ), Residence Not Provided;

**** CONTINUING DATA ***************************

This application is a REX of 10/972,588 10/25/2004 PAT 7624302

**** FOREIGN APPLICATIONS ***************************

**** IF REQUIRED, FOREIGN FILING LICENSE GRANTED ****

Foreign Priority claimed ❑ Yes ❑ No 35 USC 119(a-d) conditions met ❑ Yes ❑ No Verified and Acknowledged Examiner's Signature	❑ Met after Allowance Initials	STATE OR COUNTRY	SHEETS DRAWINGS	TOTAL CLAIMS	INDEPENDENT CLAIMS
				27	3

ADDRESS

Hewlett Packard Enterprise
3404 E. Harmony Road
Mail Stop 79
Fort Collins, CO 80528
UNITED STATES

TITLE

SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

FILING FEE RECEIVED	FEES: Authority has been given in Paper	❑ All Fees
0.00	No._____ to charge/credit DEPOSIT ACCOUNT	❑ 1.16 Fees (Filing)
	No._____ for following:	❑ 1.17 Fees (Processing Ext. of time)
		❑ 1.18 Fees (Issue)
		❑ Other _____
		❑ Credit

BIB (Rev. 05/07).

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
90/019,243	09/01/2023	7624302	185945.012600	1079

56436 7590 10/17/2023

Hewlett Packard Enterprise
3404 E. Harmony Road
Mail Stop 79
Fort Collins, CO 80528

EXAMINER	
TON, MY TRANG	

ART UNIT	PAPER NUMBER
3992	

MAIL DATE	DELIVERY MODE
10/17/2023	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

DO NOT USE IN PALM PRINTER

(THIRD PARTY REQUESTER'S CORRESPONDENCE ADDRESS)

GREENBERG TRAURIG (NJ)
500 CAMPUS DRIVE, SUITE 400
P.O. BOX 677
FLORHAM PARK NJ 07932

EX PARTE REEXAMINATION COMMUNICATION TRANSMITTAL FORM

REEXAMINATION CONTROL NO. *90/019,243* .

PATENT UNDER REEXAMINATION *7624302* .

ART UNIT *3992* .

Enclosed is a copy of the latest communication from the United States Patent and Trademark Office in the above identified *ex parte* reexamination proceeding (37 CFR 1.550(f)).

Where this copy is supplied after the reply by requester, 37 CFR 1.535, or the time for filing a reply has passed, no submission on behalf of the *ex parte* reexamination requester will be acknowledged or considered (37 CFR 1.550(g)).

Order Granting Request For Ex Parte Reexamination	Control No. 90/019,243	Patent Under Reexamination 7624302	
	Examiner MY TRANG TON	Art Unit 3992	AIA (FITF) Status No

--The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

The request for *ex parte* reexamination filed 09/01/2023 has been considered and a determination has been made. An identification of the claims, the references relied upon, and the rationale supporting the determination are attached.

Attachments: a)☐ PTO-892, b)☑ PTO/SB/08, c)☐ Other: ____

1. ☑ The request for *ex parte* reexamination is GRANTED.

RESPONSE TIMES ARE SET AS FOLLOWS:

For Patent Owner's Statement (Optional): TWO MONTHS from the mailing date of this communication (37 CFR 1.530 (b)). **EXTENSIONS OF TIME ARE GOVERNED BY 37 CFR 1.550(c).**

For Requester's Reply (optional): TWO MONTHS from the **date of service** of any timely filed Patent Owner's Statement (37 CFR 1.535). **NO EXTENSION OF THIS TIME PERIOD IS PERMITTED.** If Patent Owner does not file a timely statement under 37 CFR 1.530(b), then no reply by requester is permitted.

cc:Requester (if third party requester)

ORDER GRANTING *EX PARTE* REEXAMINATION

Ex parte reexamination has been requested for claims 1-27 ("**Requested Claims**") of United States Patent No. 7,624,302, entitled "SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR" ("**302 Patent**"). Substantial new questions of patentability (SNQs) affecting the requested claims 1-27 of the 302 Patent are raised by the request for *ex parte* reexamination ("**Request**"). Accordingly, reexamination is **GRANTED**.

PRIORITY

After careful review of the prosecution history of the 302 Patent, Examiners find that the 302 Patent was filed on October 25, 2004 and assigned U.S. Patent Application No. 10/972,588 ("**588 Application**"). Examiners also find that the 302 Patent does not claim foreign priority. Thus, the effective U.S. filing date of this instant proceeding is the actual filing date.

LITIGATION

After reviewing the Request, performing a litigation search and reviewing the file history of the 302 Patent, Examiners are aware of the following related litigations:

- (Open) 1:23cv640 Foras Technologies Ltd. V. Nissan Motor Company, Ltd. Et Al.

- (Open) 2:23cv150 Foras Technologies Ltd. V. Toyota Motor North America, Inc. Et Al.

Additionally, Examiners have reviewed the USPTO's Patent Trial and Appeal Case Tracking System ("P-TACTS")[1] for current or previous Derivation Proceedings ("DER"), Post Grant Reviews ("PGR"), and Inter Parte Reviews ("IPR") (collectively as "AIA Trials"). As of the date of this Order and based upon this review, Examiners did not locate any current or previous AIA Trials in P-TACTS.

REFERENCES/DOCUMENTS CITED AS RAISING AN SNQ

In the request for reexamination, the Request alleges that the Requested Claims 1-27 raise SNQs in light of the following prior art references:

1. U.S. Patent No. 5,450,576 to Kennedy ("**Kennedy**").
2. U.S. Patent No. 7,085,959 to Safford ("**Safford**").
3. U.S. Patent No. 7,328,371 to Kalyanasundharam ("**Kalyanasundharam**").
4. U.S. App. No. 2004/0123201 to Nguyen ("**Nguyen**")

The above cited references have been asserted in the Request as providing disclosures and/or teachings relevant to the Requested Claims 1-27 of the 302 Patent. The Request, all of its accompanying documents, and supporting

[1] Patent Trial and Appeal Case Tracking System (P-TACTS) is available on the USPTO website at: https://ptacts.uspto.gov/ptacts/ui/home (last accessed, 05 May 2023).

evidence have been fully considered in the Examiners' determinations and findings discussed below.

1/Kalyanasundharam, Kennedy and Nguyen were not of record in the prosecution of the 588 Application, later issuing as the 302 Patent.

2/ Regarding Safford, a specific discussion follows:

The propose SNQs are based in part on patents and/or printed publications already cited/considered in an earlier concluded examination of the patent being reexamined. On November 2, 2002, Public Law 107-273 was enacted. Title III, Subtitle A, Section 13105, part (a) of the Act revised the reexamination statute by adding the following new last sentence to 35 U.S.C. 303(a) and 312(a):

> *On November 2, 2002, Public Law 107-273 was enacted. Title III, Subtitle A, Section 13105, part (a) of the Act revised the reexamination statute by adding the following new last sentence to 35 U.S.C. 303(a) and 312(a):*
>
> *The existence of a substantial new question of patentability is not precluded by the fact that a patent or printed publication was previously cited by or to the Office or considered by the Office.*

For any reexamination ordered on or after November 2, 2002, the effective date of the statutory revision, reliance on previously cited/considered art, i.e., "old art," does not necessarily preclude the existence of a substantial new question of patentability (SNQ) that is based exclusively on that old art. Rather, determinations on whether a SNQ exists

in such an instance shall be based upon a fact-specific inquiry done on a case-by-case basis.

In the present instance, there exist substantial new questions of patentability based on the previously cited patent to Safford.

With regard to Safford, this reference was the subject of extensive written discussion on the record of the 588 Application, it is clear that the request presents its teachings **in a new light or in a different way**, as compared with its use in the earlier concluded examination. In particular, Safford is used in combination with different arts (Kennedy, Nguyen, Cepulis). As noted above, Kennedy, Nguyen and Cepulis were not of record in the prosecution of the 588 Application, later issued as 302 Patent. Neither Kennedy, Nguyen nor Cepulis were previously combined with Safford. The combinations of these references are not deemed cumulative to the teachings of the prior art already of record. A reasonable examiner would view the teachings of the combination may be interpreted to provide, for example: "*changing a system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected, to a different value, which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor; and changing a system identifier of the spare processor to that of the boot processor..*" important in deciding to allow the Requested claims 1-27 being considered, thus raising SNQs regarding the Requested claims of the 302 Patent. In view of the materially new interpretation presented in the Request, Safford is available as prior art consistent with the

holdings of *Ex Parte Chicago Rawhide Mfg. Co.*, 223 USPQ 351 (Bd. Pat. App. & Inter. 1984).

RELEVANT PROSECUTION HISTORY

The followings are the Examiners' findings of fact related to the prosecution history of the 302 Patent, as evidenced in the documents of record in the prosecution history of the 588 Application:

10/25/2004 Patent Owner filed the 588 Application with 30 claims ("**examined claims**"), of which examined claims 1, 16, 24 and 29 were independent claims.

09/12/2007 The original Examiner issued a non-final Office action ("**September 2007 NF**") rejected examined claims 1-30 under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1-32 of U.S. application 10/973,077, and claims 1-26 of U.S. application 10/972,796; rejected examined claims 1-4, 7, 9-23, and 29 under 35 U.S.C. 102(e) as being anticipated over Safford; and rejected examined claims 5-6, 8, 24-28, and 30 under 35 U.S.C. 103(a) as being unpatentable over Safford in view of Marshall et al. (U.S. Patent 5,915,082).

12/12/2007 Patent Owner responded to the September 2007 NF ("**December 2007 Response**"). Therein, the Patent Owner inquired that the

double patenting rejection be held in abeyance until allowable subject matter had been reached, and traversed the 35 U.S.C 102(e) and 103(a) rejections. Patent Owner argued that "*Safford does not discuss boot processors, the system determining whether a processor is the boot processor, or switching the role of boot processor from one processor to another*" (page 9 of the December 2007 Response); "*Safford does not discuss transfer of system information (which processer is the boot processor) that remains permanent and survives a system reset. Safford discusses rebooting individual processors while a system is running (for example, column 4, lines 57-60, and claim3), but that is not the same as rebooting the entire system, which is initiated by the one designated boot processor after a system reset or power shutdown*" (page 10 of the December 2007 Response).

03/10/2008 The original Examiner issued a second non-final Office action ("**March 2008 NF**") rejected examined claims 1-30 under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1-32 of U.S. application 10/973,077, and claims 1-26 of U.S. application 10/972,796; rejected examined claims 1-4, 7, 9-23, and 29 under 35 U.S.C. 102(e) as being anticipated over Safford; rejected examined claims 1-23 and 29-30 under 35 U.S.C. 103(a) as being unpatentable over Quach (U.S. Patent 6,625,749) in view of Fox

et al (U.S. Patent 7,251,749); and indicated allowability for claims 24-28.

06/05/2008 Patent Owner responded to the March 2008 NF ("**June 2008 Response**"). In the June 2008 Response, the Patent Owner inquired that the double patenting rejection be held in abeyance until allowable subject matter had been reached, and traversed the 35 U.S.C 102(e) and 103(a) rejections. Patent Owner argued that "*1/ Safford does not discuss a boot processor. 2/ Safford does not teach or suggest switching the role of boot processor. 3/ Re-booting a single processor is not the same as switching the role of boot processor.*" (page 3 of the June 2008 Response); "... *Figure 1 and the cited text do not illustrate or discuss anything about boot processors*" (page 4, 2nd par. of the June 2008 Response); and "... *The renaming being discussed has to do with registers, not processors, and in particular does not teach or suggest anything about a boot processor or switching the role of boot processor... In column 14, claim 6 discusses initializing individual processors, which has nothing to do with a boot processor or switching the role of boot processor.*" (page 4, last paragraph of the June 2008 Response).

09/05/2008 Telephonic interview was conducted.

09/10/2008 Patent Owner filed an Amendment to the examined claims ("**September 2008 Amendment**"). Therein, examined claims 1,

16, 29 were amended, examined claims 2-12, 17-28, 30 were unchanged, and examined claims 13-15 were cancelled. Thus, examined claims 1-12 and 16-30 were remained. Of these, examined claims 1, 16, 24 and 29 were independent claims.

Claim 1 was amended as follow:

> 1. (currently amended) A method comprising:
>
> detecting loss of lockstep (LOL) for a processor in a multi-processor system;
>
> determining that the processor for which said LOL is detected is assigned a role of boot processor; and
>
> changing a system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected, to a different value, which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor; and
>
> changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to a spare processor without shutting down the system's operating system.

Claim 16 was amended as follow:

> 16. (currently amended) A method comprising:
>
> system firmware determining that loss of lockstep (LOL) is detected for a

lockstep pair of processors that are assigned the role of boot processor in a system;

determining one of said lockstep pair of processors that is not the cause of said LOL;

copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to a spare processor; and

changing a system identifier, used by the system's operating system to recognize the boot processor, of said lockstep pair of processors that are assigned the role of boot processor, to a different value, which causes the system's operating system to not recognize the lockstep pair of processors for which said LOL is detected as the boot processor; and

changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to said spare processor.

Claim 29 was amended as follow:

29. (currently amended) A system comprising:

system firmware means for detecting loss of lockstep (LOL) for a processor in a multi-processor system;

system firmware means for determining whether the processor for which said LOL is detected is assigned the role of boot processor; and

system firmware means for switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which said LOL is detected is assigned the role of boot processor.

Patent Owner noted that "... *In response to that interview, claim 1 has been amended to incorporate the limitations of claims 13-15*

and claims 13-15 have been canceled. Claim 16 has been amended to incorporate similar limitations as claim 1. In addition, claim 29 has been amended to add system firmware as additional structure for the means plus function claims" (page 7). Terminal Disclaimers were filed along with the September 2008 Amendment.

09/23/2008 The original Examiner issued a Notice of Allowability allowing examined claims 1-12 and 16-30 without a statement of reasons for allowance.

12/09/2008 Patent Owner filed a Request for continued examiner (**RCE**). Information Disclosure Statement (**IDS**) was filed along with the RCE.

02/12/2009 The original Examiner issued a Notice of Allowability allowing examined claims 1-12 and 16-30. No specific reasons for allowance were provided by the original Examiner. The examined claims 1-12 and 16-30 were renumbered as the Requested Claims 1-27. IDS was entered.

11/24/2009 The 588 Application issued as the 302 Patent.

SUBSTANTIAL NEW QUESTIONS OF PATENTABILITY

For a substantial new question of patentability (SNQ) to be present the MPEP states that it is only necessary that:

(A) the prior art patents and/or printed publications raise a substantial question of patentability regarding at least one claim, i.e., the teaching of the (prior art) patents and printed publications is such that a reasonable examiner would consider the teaching to be important in deciding whether or not the claim is patentable; and

(B) the same question of patentability as to the claim has not been decided by the Office in a previous examination or pending reexamination of the patent or in a final holding of invalidity by the Federal Courts in a decision on the merits involving the claim. See MPEP § 2242.

Furthermore, "[i]t must first be demonstrated that a patent or printed publication that is relied upon in a proposed rejection presents a new, non-cumulative technological teaching that was not previously considered and discussed on the record during the prosecution of the application that resulted in the patent for which reexamination is requested, and during the prosecution of any other prior proceeding involving the patent for which reexamination is requested." See MPEP §2216.

Based upon the prosecution history of the 588 Application, Examiners make the following findings:

1/ Examiners find that Patent Owner added new features to the examined claims 1, 16, 29 as noted above in the September 2008 Amendment to overcome the application of prior art, specifically those features shown above in the amendments to the examined claims 1, 16 and 29 of the 588 Application, which became the

Requested Claims 1, 13 and 26. Patent Owner also argued patentability based on the newly added features.

2/ Although there is no specific reasons for allowance were provided by the original Examiner. Examiners find that the original Examiner allowed the 588 Application on the basis of the Patent Owner's amendments and arguments to the examined claims 1, 16 and 29 in the September 2008 Amendment.

Based on the above findings, Examiners determine that the technological features that overcame the prior art cited during the prosecution of the 588 Application that became the 302 Patent were those features of amended into the examined claims 1, 16 and 29 : "*changing a system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected. to a different value, which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor: and changing a system identifier of the spare processor to that of the boot processor ...*" (claim 1), or "*changing a system identifier, used by the system's operating system to recognize the boot processor, of said lockstep pair of processors that are assigned the role of boot processor, to a different value. which causes the system's operating system to not recognize the lockstep pair of processors for which said LOL is detected as the boot processor; and changing a system identifier of the spare processor to that of the boot processor ...*" (claim 16), "*system firmware means for determining whether the processor*

for which said LOL is detected is assigned the role of boot processor; and system firmware means for switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which said LOL is detected is assigned the role of boot processor" (claim 29). Accordingly, Examiners conclude that these technological features would be important to Examiners because these features provided the basis for allowance of the examined claims 1, 16 and 29 and the issuance of the 302 Patent.

In view of these findings, Examiners conclude that if the Request presents a new reference or combination of references that discloses or teaches the technological features (i.e., the combination of elements specified in the claims) of:

[a] "*changing a system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected. to a different value, which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor; and changing a system identifier of the spare processor to that of the boot processor ...*" as recited in the Requested claim 1 and those Requested claims 2-12 depending therefrom.

[b] "*changing a system identifier, used by the system's operating system to recognize the boot processor, of said lockstep pair of processors that are assigned the role of boot processor, to a different value. which*

causes the system's operating system to not recognize the lockstep pair of processors for which said LOL is detected as the boot processor; and changing a system identifier of the spare processor to that of the boot processor ..." as recited in the Requested claim 13 and those Requested claims 14-20 depending therefrom.

[c] *"system firmware means for determining whether the processor for which said LOL is detected is assigned the role of boot processor; and system firmware means for switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which said LOL is detected is assigned the role of boot processor"* as recited in the Requested claim 26 and the Requested claim 27 depending therefrom.

Then that new reference or combination of references would be important to Examiners as such teachings would be directed to those technological features for which the claims were allowed and/or argued to overcome the applied prior art. Thus, the new reference or combination of references disclosing the feature **[a]**, **[b]** or **[c]** would raise a substantial new question (SNQ) of patentability for the respective claims.

Moreover, the examined claims 24-28 (renumbered as the Requested claims 21-25) were indicated as allowed claims in the March 2008 NF. However, no specific reasons were provided. Thus, the prosecution history does not shed light on the question of what limitations defined the examined

claims 24-28 over the prior art of record. In view of the ambiguity of the record, any reference or combination of references that includes a teaching recited in the examined claims 24-28 28 (renumbered as the Requested claims 21-25) raise a substantial new question of patentability where such teaching is not seen to be cumulative of the teaching of the prior arts of record.

PROPOSED SNQs

The Request proposes combinations of references to raise SNQs for the Requested Claims 1-27 in the 302 Patent (Request, pages 23, 33, 46):

Ground 1 – The Request submits Kalyanasundharam and Kennedy raise SNQ for the Requested claims 1, 5-8, 13, 17, 21, 22, 25-26.

Ground 2 – The Request submits Kennedy, Nguyen and Safford raise SNQ for the Requested claims 1-27.

Ground 3 – The Request submits Safford and Cepulis raise SNQ for the Requested claims 1-4.

DISCUSSION OF PROPOSED SNQs

Ground 1: Kalyanasundharam and Kennedy **raise** SNQ for the Requested claims 1, 5-8, 13, 17, 21, 22 and 25-26 (pages 47-57).

Examiners **agree** with the Requester that the consideration of Kalyanasundharam and Kennedy raise a substantial new question of patentability to the Requested claims 1, 5-8, 13, 17, 21, 22 and 25-26 of the 302 Patent. As presented in the Request and in view of the detailed explanation

in pages 47-57 offered by the Requester, a reasonable examiner would consider Kalyanasundharam and Kennedy important in making a decision as to the patentability of the Requested claims 1, 5-8, 13, 17, 21, 22 and 25-26.

Consider, for example, the following limitations from the Requested claim 1 of the 302 Patent which is representative:

Regarding claim 1: Kalyanasundharam discloses in Figs. 1-7 **a method comprising:**

detecting loss of lockstep (LOL) for a processor in a multi-processor system

Kalyanasundaram shows in Fig. 1 an error detecting mechanism that checks for errors within a pair of Processor Cores 18A-18B that execute in lockstep: "*In embodiments in which the processor cores 18A- 18B execute in lockstep, the redundancy check circuit 26A may check that corresponding communications are concurrently received from the processor cores 18A-18B (e.g. received on the same clock cycle). For example, if a valid communication is received from one of the processor cores 18A-18B and a valid communication is not concurrently received from the other processor core 18A-18B, an execution error may be detected* (4:27-35);

determining that the processor for which said LOL is detected

Kalyanasundharam teaches "*an execution error may be any indication that an error has occurred in the execution of the software code, detected via the redundant execution.*" (3:63-65), "*Diagnostic software may be used to determine which processor core 18A-18B experienced the error and what corrective*

measures are to be taken." (3:67-4:3), and "*If the masked comparison results indicate that the communication from the primary processor core mismatches with all auxiliary processor core (that are enabled)(decision block 100, "yes" leg), the control circuit 84 may disable the primary processor core and select one of the auxiliary processor cores to be the primary processor core (block 102)*" (13:16-22),

is assigned a role of boot processor

While Kalyanasundharam determines, upon detecting a loss of lockstep, whether a processor is a primary processor core, Kalyanasundharam fails to specifically indicate that the processor is a "boot processor." However, Kennedy shows a multiprocessor system that identifies default boot processors and alternative boot processors should the default boot processor fail to boot (7:5-13). A POSITA would have been motivated to combine the known prior art elements of processor failover from a primary processor core to an auxiliary processor core, as taught by Kalyanasundharam, with the teachings of boot processor failover, as described by Kennedy, to increase the reliability and availability of the underlying system.

changing a system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected, to a different value

Kalyanasundharam teaches "*As with the discussion above for disabling the auxiliary processor core, the control circuit 84 may simply disable the primary processor core from comparison by clearing the corresponding enable bit*

in the configuration register 42, or may disable the primary processor core from executing at all. The control circuit 84 may also update the PC field of the configuration register 42 to reflect the newly selected primary processor core" (13:22-29).

Kennedy also teaches "*If the CPU1 120 in slot 1 does not boot (proper boot is indicated by a change in the level of the SLOT SELECT signal line 193 from high to low) before the timer 184 in slot 2 times-out, then the CPU2 132 exits the reset state and attempts to boot the system as an alternate boot CPU.*" (15:9-14) and "*If the CPU1 120 does not complete boot operations within the time allotted, then the CPU2 132 takes over default boot operations as an alternative boot processor instead of CPU1 120. If the CPU1 120 does boot properly, then the CPU2 132 checks in as an additional processor rather than taking over as a boot processor.*" (13:38-44),

which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor

Kennedy teaches "*If the CPU1 120 does not complete boot operations within the time allotted, then the CPU2 132 takes over default boot operations as an alternative boot processor instead of CPUI1 120. If the CPU1 120 does boot properly, then the CPU2 132 checks in as an additional processor rather than taking over as a boot processor.*" (13:38-44);

changing a system identifier of the spare processor to that of the boot processor

Kalyanasundharam teaches "... *As with the discussion above for disabling the auxiliary processor core, the control circuit 84 may simply disable the primary processor core from comparison by clearing the corresponding enable bit in the configuration register 42, or may disable the primary processor core from executing at all. The control circuit 84 may also update the PC field of the configuration register 42 to reflect the newly selected primary processor core.*" (13:22-29). Kennedy also teaches "... *provision needs to be made to determine which processor will boot the system, and if this 'default' boot processor fails to boot the system, an alternative processor, if one is installed, should take over boot operations to enhance the availability of the system.*" (7:8-13),

thereby switching the role of boot processor to a spare processor without shutting down the system's operating system.

Kalyanasundharam teaches a control circuit 84 that can "*update the PC field of the configuration register 42 to reflect the newly selected primary processor core.*" (13:27-29).

Thus, Kalyanasundharam in combination with Kennedy teach relevant element of claim 1.

These references therefore provides a new technical teaching that were not present during the original prosecution, and a reasonable Examiner would have found these references important in considering the patentability of the Requested claim 1. Thus, a SNQ is raised.

Examiners find the teachings of these references as discussed are not cumulative to any written discussion on the record of the teachings of the prior

art, were not previously considered nor addressed in this manner during a prior examination, and the same questions were not the subject of a final holding of invalidity in the Federal Courts.

In regard to claims 5-8 such claims are dependent on independent claim 1. Dependent claims 5-8 are construed to include all the limitations of the independent claim 1. Thus, Examiners conclude Kalyanasundharam in combination with Kennedy would raise a SNQ for claims 5-8. Claims 13, 17, 21, 22, 25-26 are similar to claims 1, 5-8, thus, Examiners conclude Kalyanasundharam in combination with Kennedy would also raise a SNQ for these claims.

More particularly, the detail explanation in pages 23-33, 47-57 and Appendix A offered by the Requester is deemed plausible to the degree that further consideration is warranted.

Ground 2 – Kennedy, Nguyen and Safford **do not raise** SNQ for the Requested claims 1-27 (pages 57-71).

The Request submits that Kennedy, Nguyen and Safford raise SNQs for the Requested claims 1-27. Examiners **disagree**.

As noted above, Safford is available as prior art in a new light or in a different way because Safford is used in combination with different arts, i.e., Kennedy and Nguyen, as compared with its use in the earlier concluded examination. However, the Request relies on Safford for the teaching of "**[a]** *changing a system identifier, used by the system's operating system to recognize*

the boot processor, of said boot processor for which said LOL is detected. to a different value, which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor; and changing a system identifier of the spare processor to that of the boot processor ...";

[b] "*changing a system identifier, used by the system's operating system to recognize the boot processor, of said lockstep pair of processors that are assigned the role of boot processor, to a different value. which causes the system's operating system to not recognize the lockstep pair of processors for which said LOL is detected as the boot processor; and changing a system identifier of the spare processor to that of the boot processor ...*"; and ***[c]*** "*system firmware means for determining whether the processor for which said LOL is detected is assigned the role of boot processor; and system firmware means for switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which said LOL is detected is assigned the role of boot processor*".

Nevertheless, these precise matters [a], [b], [c] were already determined in the original prosecution of the 588 Application, where the original Examiner had before her Stafford that ultimately became the 302 Patent. The original Examiner allowed the 588 Application on the basis of the Patent Owner's amendments (adding [a], [b], or [c] to the claims to overcome Safford) and arguments to the claims in the September 2008 Amendment. While Safford might be considered to raise a substantially question of patentability, nonetheless, these precise matters [a], [b], [c] were squarely before the original

Examiner during the prior examination, and the original Examiner unequivocally decided the matter, therefore, there is nothing new to make this an SNQ.

Moreover, Examiners have further independently reviewed Safford and do not believe that Safford discloses features [a], [b] or [c]. Stanford does not teach switching the role of boot processor: "*changing a system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected. to a different value, which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor; and changing a system identifier of the spare processor to that of the boot processor ...*". Rather, Stanford appears to teach rebooting individual processor while a system is running. Accordingly, Examiners find that the combination of Kennedy, Nguyen and Safford would not raise an SNQ.

Ground 3 – Safford in view of Cepulis **do not** raise SNQ for the Requested claims 1-4 (pages 71-76).

The Request submits that Safford in view of Cepulis raise SNQ for the Requested claims 1-4. Examiners **disagree**.

As noted above, Safford is available as prior art in a new light or in a different way because in the Request, Safford is used in combination with different art (Cepulis), as compared with its use in the 588 Application. However, the Request relies on Safford for the teaching of "**[a]** *changing a*

system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected. to a different value, which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor; and changing a system identifier of the spare processor to that of the boot processor ...".

Nevertheless, this precise matter [a] was already determined in the original prosecution of the 588 Application, where the original Examiner had before her Safford that ultimately became the 302 Patent. The original Examiner allowed the 588 Application on the basis of the Patent Owner's amendments (adding [a] to claim 1 to overcome Safford) and arguments to claim 1 in the September 2008 Amendment. While Safford might be considered to raise a substantially question of patentability, nonetheless, this precise matter [a] was squarely before the original Examiner during the prior examination, and the original Examiner unequivocally decided the matter, therefore, there is nothing new to make this an SNQ.

Moreover, Examiners have further independently reviewed Safford and do not believe that Safford discloses feature [a]. Stanford does not teach switching the role of boot processor: "*changing a system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected. to a different value, which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor; and changing a system identifier of the spare processor to that of the boot processor ...*". Rather, Stanford appears to teach rebooting

individual processor while a system is running. Accordingly, Examiners find that the combination of Safford in view of Cepulis would not raise an SNQ.

SCOPE OF REEXAMINATION

On the basis of the findings above:

1/ Examiners **agree** that Ground 1 raises SNQ for the Requested claims 1, 5-8, 13, 17, 21, 22 and 25-26 in the 302 Patent.

2/ Examiners **do not agree** Ground 2 raises SNQ for the Requested claims 1-27 or Ground 3 raises SNQ for the Requested claims 1-4 in the 302 Patent.

3/ It is noted that since SNQs for all independent claims 1, 13, 21, 26 in Ground 1 then same SNQs would apply to dependent claims 2-4, 9-12, 14-16, 18-20, 23, 24 and 27.

INFORMATION DISCLOSURE STATEMENT

Regarding Information Disclosure Statement (IDS) submissions on September 1, 2023 (“**2023 IDS**”), MPEP 2256 recites the following: “*Where patents, publications, and other such items of information are submitted by a party (patent owner or requester) in compliance with the requirements of the rules, the requisite degree of consideration to be given to such information will be normally limited by the degree to which the party filing the information citation has explained the content and relevance of the information. The initials of the*

examiner placed adjacent to the citations on the form PTO/SB/08A and 08B or its equivalent, without an indication to the contrary in the record, do not signify that the information has been considered by the examiner any further than to the extent noted above." Accordingly, the 2023 IDS has been considered by Examiners only with the scope required by MPEP 2256, unless otherwise noted.

DUTY TO DISCLOSE

The Patent Owner is reminded of the continuing responsibility under 37 C.F.R. § 1.565(a) to apprise the Office of any litigation activity, or other prior or concurrent proceeding, throughout the course of this reexamination proceeding involving the 302 Patent. The Requester is also reminded of the ability to similarly apprise the Office of any such activity or proceeding throughout the course of this reexamination proceeding. See MPEP §§ 2207, 2282 and 2286.

SERVICE OF PAPERS

After the filing of a request for reexamination by the Requester, any document filed by either the Patent Owner or the Requester must be served on the other party (or parties where two or more third party requester proceedings are merged) in the reexamination proceeding in the manner provided in 37 C.F.R. § 1.248 (See also 37 C.F.R. § 1.550).

EXTENSIONS OF TIME

Extensions of time under 37 C.F.R. §1.136(a) will not be permitted in these proceedings because the provisions of 37 C.F.R. §1.136 apply only to "an applicant" and not to parties in a reexamination proceeding. Additionally, 35 U.S.C. §305 requires that reexamination proceedings "will be conducted with special dispatch" (37 C.F.R. §1.550(a)). Extension of time in reexamination proceedings are provided for in 37 C.F.R. §1.550(c). After the filing of a request for reexamination by the Requester, any document filed by either the Patent Owner or the Requester must be served on the other party (or parties where two or more third-party-requester proceedings are merged) in the reexamination proceeding in the manner provided in 37 C.F.R. §1.248. See 37 C.F.R. §1.550(f).

35 U.S.C. § 325(d)

A review of the post grant history for the 302 Patent indicates that there have been no other Office post grant challenges involving the patent (Reexamination Proceedings or *Inter Partes* Review, Post Grant Review, Covered Business Method trials). Accordingly, a determination of whether to exercise discretion to reject the request pursuant to 35 U.S.C. § 325(d) is not applicable.

CONCLUSION

Reexamination of the Requested Claims (*i.e.* claims 1-27) is hereby ORDERED.

All correspondence relating to this ex-parte reexamination proceeding should be directed:

By EFS: Registered users may submit via the electronic filing system (EFS-Web, at https://efs.uspto.gov/efile/myportal/efs-registered)

By mail to: Mail Stop Ex Parte Reexam
Central Reexamination Unit
Commissioner for Patents
United States Patent & Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

By Fax to: (571) 273-9900
Central Reexamination Unit

By hand: Customer Service Window
Randolph Building
401 Dulany Street
Alexandria, VA 22314

For EFS-Web transmissions 37 CFR 1.8(a)(1)(i) (C) and (ii) states that the correspondence (except for a request for reexamination and a corrected or replacement request for reexamination) will be considered timely filed if (a) it is transmitted via the Office's electronic filing system in accordance with 37 CFR 1.6(a)(4), and (b) includes a certificate of transmission for each piece of correspondence stating the date of transmission, which is prior to the expiration of the set period of time in the Office action.

Any inquiry concerning this communication or earlier communications from the Reexamination Legal Advisor or Examiner, or as to the status of this proceeding, should be directed to the Central Reexamination Unit at telephone number (571) 272-7705. The CRU's fax number is 571-273-9900.

Signed:

/MY TRANG TON/
Primary Examiner, Art Unit 3992

Conferees:

/KENNETH WHITTINGTON/
Primary Examiner, Art Unit 3992

/ANDREW J. FISCHER/
Supervisory Patent Reexamination Specialist, Art Unit 3992

Reexamination	Application/Control No.	Applicant(s)/Patent Under Reexamination
	90/019,243	7624302
	Certificate Date	Certificate Number

Requester Correspondence Address: ☐ Patent Owner ☑ Third Party
GREENBERG TRAURIG (NJ) 500 CAMPUS DRIVE, SUITE 400 P.O. BOX 677 FLORHAM PARK NJ 07932

LITIGATION REVIEW ☑	mt (examiner initials)	18 September 2023 (date)
Case Name		Director Initials
1:23cv640 | Foras Technologies Ltd. V. Nissan Motor Company, Ltd. Et Al (Open)		
2:23cv150 Foras Technologies Ltd. V. Toyota Motor North America. Inc. Et Al (Open)		

COPENDING OFFICE PROCEEDINGS	
TYPE OF PROCEEDING	NUMBER
None	

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	90/019,243	7624302
	Examiner	Art Unit
	MY TRANG TON	3992

CPC - Searched*		
Symbol	Date	Examiner

CPC Combination Sets - Searched*		
Symbol	Date	Examiner

US Classification - Searched*			
Class	Subclass	Date	Examiner

* See search history printout included with this form or the SEARCH NOTES box below to determine the scope of the search.

Search Notes		
Search Notes	Date	Examiner
P-TACTS searched	09/18/2023	mt
PDP searched	09/18/2023	mt
Reviewed US Pat. No. 7624302 & App. No. 10/972588 file history	09/18/2023	mt
Litigation reviewed	09/18/2023	mt

Interference Search			
US Class/CPC Symbol	US Subclass/CPC Group	Date	Examiner

PTO/SB/08a (01-10)
Approved for use through 07/31/2012. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Not for submission under 37 CFR 1.99)

Application Number	90/019243
Filing Date	2023-09-01
First Named Inventor	Scott L. Michaelis
Art Unit	~~3992~~
Examiner Name	~~Unassigned~~ Ton
Attorney Docket Number	185945.012600

U.S.PATENTS Remove

Examiner Initial*	Cite No	Patent Number	Kind Code[1]	Issue Date	Name of Patentee or Applicant of cited Document	Pages,Columns,Lines where Relevant Passages or Relevant Figures Appear
/M.T/	1	7624302		2009-11-24	Michaelis et al.	
	2	7085959		2022-06-17	Safford	
	3	7328371		2008-02-05	Kalyanasundharam et al.	
	4	5450576		1995-09-12	Kennedy et al.	
	5	6625749		2019-12-25	Quach	
	6	7191328		2007-03-13	Hobson	
	7	5491788		2014-05-14	Cepulis et al.	
/M.T/	8	5915082		2016-05-11	Marshall	

If you wish to add additional U.S. Patent citation information please click the Add button. Add

INFORMATION DISCLOSURE STATEMENT BY APPLICANT (Not for submission under 37 CFR 1.99)		
	Application Number	
	Filing Date	2023-09-01
	First Named Inventor	Scott L. Michaelis
	Art Unit	
	Examiner Name	Unassigned
	Attorney Docket Number	185945.012600

U.S.PATENT APPLICATION PUBLICATIONS							Remove
Examiner Initial*	Cite No	Publication Number	Kind Code[1]	Publication Date	Name of Patentee or Applicant of cited Document	Pages,Columns,Lines where Relevant Passages or Relevant Figures Appear	
/M.T/	1	20040123201		2006-05-30	Nguyen et al.		
/M.T/	2	20050172164		2007-07-31	Fox et al.		
	3						

If you wish to add additional U.S. Published Application citation information please click the Add button. Add

FOREIGN PATENT DOCUMENTS								Remove
Examiner Initial*	Cite No	Foreign Document Number[3]	Country Code[2] ¡	Kind Code[4]	Publication Date	Name of Patentee or Applicant of cited Document	Pages,Columns,Lines where Relevant Passages or Relevant Figures Appear	T[5]
	1							☐

If you wish to add additional Foreign Patent Document citation information please click the Add button Add

NON-PATENT LITERATURE DOCUMENTS			Remove
Examiner Initials*	Cite No	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc), date, pages(s), volume-issue number(s), publisher, city and/or country where published.	T[5]
/M.T/	1	U.S. File History of the '302 Patent ("U.S. File History")	☐
/M.T/	2	Declaration of Andrew Wolfe ("Wolfe Decl.")	☐

INFORMATION DISCLOSURE STATEMENT BY APPLICANT (Not for submission under 37 CFR 1.99)		
	Application Number	
	Filing Date	2023-09-01
	First Named Inventor	Scott L. Michaelis
	Art Unit	
	Examiner Name	Unassigned
	Attorney Docket Number	185945.012600

/M.T/	3	Wendy Bartlett, et. al., Commercial Fault Tolerance: A Tale of Two Systems, IEEE Transactions on Dependable and Secure Computing, Vol. 1, No. 1 (January-March 2004) ("Bartlett")	☐
	4	Intel, Extensible Firmware Interface Specification, Version 1.10 (Dec. 1, 2002) ("Intel EFI Specification")	☐
	5	Bozidar Levi, UNIX Administration, A Comprehensive Sourcebook for Effective Systems and Network Management (2002) ("Levi")	☐
	6	Intel IA-64 Architecture Software Developer's Manual, Revision 1.1 (July 2000) ("Intel Itanium Manual")	☐
/M.T/	7	Intel Multiprocessor Specification, Version 1.4 (May 1997) ("Intel Multiprocessor Specification")	☐

If you wish to add additional non-patent literature document citation information please click the Add button Add

EXAMINER SIGNATURE

Examiner Signature	/MY TRANG TON/	Date Considered	09/21/2023

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through a citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

[1] See Kind Codes of USPTO Patent Documents at www.USPTO.GOV or MPEP 901.04. [2] Enter office that issued the document, by the two-letter code (WIPO Standard ST.3). [3] For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [4] Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [5] Applicant is to place a check mark here if English language translation is attached.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT (**Not for submission under 37 CFR 1.99)**		
	Application Number	
	Filing Date	2023-09-01
	First Named Inventor	Scott L. Michaelis
	Art Unit	
	Examiner Name	Unassigned
	Attorney Docket Number	185945.012600

CERTIFICATION STATEMENT

Please see 37 CFR 1.97 and 1.98 to make the appropriate selection(s):

☐ That each item of information contained in the information disclosure statement was first cited in any communication from a foreign patent office in a counterpart foreign application not more than three months prior to the filing of the information disclosure statement. See 37 CFR 1.97(e)(1).

OR

☐ That no item of information contained in the information disclosure statement was cited in a communication from a foreign patent office in a counterpart foreign application, and, to the knowledge of the person signing the certification after making reasonable inquiry, no item of information contained in the information disclosure statement was known to any individual designated in 37 CFR 1.56(c) more than three months prior to the filing of the information disclosure statement. See 37 CFR 1.97(e)(2).

☐ See attached certification statement.

☐ The fee set forth in 37 CFR 1.17 (p) has been submitted herewith.

☒ A certification statement is not submitted herewith.

SIGNATURE

A signature of the applicant or representative is required in accordance with CFR 1.33, 10.18. Please see CFR 1.4(d) for the form of the signature.

Signature	/Greg Lunt/	Date (YYYY-MM-DD)	2023-09-01
Name/Print	Greg Lunt	Registration Number	57354

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 1 hour to complete, including gathering, preparing and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C. 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether the Freedom of Information Act requires disclosure of these record s.

2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.

3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.

4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).

5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.

6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).

7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (i.e., GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.

8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspections or an issued patent.

9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
90/019,243	09/01/2023	7624302	185945.012600	1079

56436 7590 02/23/2024

Hewlett Packard Enterprise
3404 E. Harmony Road
Mail Stop 79
Fort Collins, CO 80528

EXAMINER	
TON, MY TRANG	

ART UNIT	PAPER NUMBER
3992	

MAIL DATE	DELIVERY MODE
02/23/2024	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

DO NOT USE IN PALM PRINTER

(THIRD PARTY REQUESTER'S CORRESPONDENCE ADDRESS)

GREENBERG TRAURIG (NJ)
500 CAMPUS DRIVE, SUITE 400
P.O. BOX 677
FLORHAM PARK NJ 07932

EX PARTE REEXAMINATION COMMUNICATION TRANSMITTAL FORM

REEXAMINATION CONTROL NO. *90/019,243* .

PATENT UNDER REEXAMINATION *7624302* .

ART UNIT *3992* .

Enclosed is a copy of the latest communication from the United States Patent and Trademark Office in the above identified *ex parte* reexamination proceeding (37 CFR 1.550(f)).

Where this copy is supplied after the reply by requester, 37 CFR 1.535, or the time for filing a reply has passed, no submission on behalf of the *ex parte* reexamination requester will be acknowledged or considered (37 CFR 1.550(g)).

Office Action in Ex Parte Reexamination	**Control No.** 90/019,243	**Patent Under Reexamination** 7624302	
	Examiner MY TRANG TON	**Art Unit** 3992	**AIA (FITF) Status** No

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

a. ☐ Responsive to the communication(s) filed on _____.
☐ A declaration(s)/affidavit(s) under **37 CFR 1.130(b)** was/were filed on _____.

b. ☐ This action is made FINAL.

c. ☑ A statement under 37 CFR 1.530 has not been received from the patent owner.

A shortened statutory period for response to this action is set to expire 2 month(s) from the mailing date of this letter. Failure to respond within the period for response will result in termination of the proceeding and issuance of an *ex parte* reexamination certificate in accordance with this action. 37 CFR 1.550(d). **EXTENSIONS OF TIME ARE GOVERNED BY 37 CFR 1.550(c).** If the period for response specified above is less than thirty (30) days, a response within the statutory minimum of thirty (30) days will be considered timely.

Part I THE FOLLOWING ATTACHMENT(S) ARE PART OF THIS ACTION:

1. ☐ Notice of References Cited by Examiner, PTO-892.
2. ☐ Information Disclosure Statement, PTO/SB/08.
3. ☐ Interview Summary, PTO-474.
4. ☐ _____.

Part II SUMMARY OF ACTION

1a. ☑ Claims 1,5-8,13,17,21-22 and 25-26 are subject to reexamination.

1b. ☑ Claims 2-4,9-12,14-16,18-20,23-24 and 27 are not subject to reexamination.

2. ☐ Claims _____ have been canceled in the present reexamination proceeding.

3. ☑ Claims 21-22 and 25-26 are patentable and/or confirmed.

4. ☑ Claims 1,5-8,13 and 17 are rejected.

5. ☐ Claims _____ are objected to.

6. ☐ The drawings, filed on _____ are acceptable.

7. ☐ The proposed drawing correction, filed on _____ has been (7a) ☐ approved (7b) ☐ disapproved.

8. ☐ Acknowledgment is made of the priority claim under 35 U.S.C. 119(a)-(d) or (f).

a) ☐ All b) ☐ Some* c) ☐ None of the certified copies have

1 ☐ been received.

2 ☐ not been received.

3 ☐ been filed in Application No. _____.

4 ☐ been filed in reexamination Control No. _____.

5 ☐ been received by the International Bureau in PCT application No. _____.

* See the attached detailed Office action for a list of the certified copies not received.

9. ☐ Since the proceeding appears to be in condition for issuance of an *ex parte* reexamination certificate except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte* Quayle, 1935 C.D. 11, 453 O.G. 213.

10. ☐ Other: _____

cc: Requester (if third party requester)

EX PARTE REEXAMINATION NON-FINAL OFFICE ACTION

This action addresses *Ex Parte* reexamination of claims 1, 5-8, 13, 17, 21-22 and 25-26 of U.S. Patent No. 7,624,302 (the "**302 Patent**"). The 302 Patent issued on November 24, 2009 to Michaelis et al, and was titled "SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR".

PRIORITY

After careful review of the prosecution history of the 302 Patent, Examiners find that the 302 Patent was filed on October 25, 2004 and assigned U.S. Patent Application No. 10/972,588 (the "**588 Application**"). The Examiners find that the 302 Patent does not claim foreign priority.

LITIGATION

After reviewing a Request, performing a litigation search and reviewing the file history of the 302 Patent, the Examiners are aware of the following related proceeding and litigations:

- (Open) 1:23cv640 Foras Technologies Ltd. V. Nissan Motor Company, Ltd. Et Al;
- (Open) 2:23cv150 Foras Technologies Ltd. V. Toyota Motor North America, Inc. Et Al.

SUMMARY OF THE PROCEEDING

Below are the pertinent Examiners' findings of fact relevant to this proceeding:

09/01/2023 A Third Party Requester ("**Requester**") filed a request for ex parte reexamination of claims 1-27 of the 302 Patent ("**the Request**"). The Request was assigned reexamination control number 90/019,243 ("**Instant Reexamination Proceeding**").

10/17/2023 The Office issued an order ("**Order**") granting reexamination of the requested claims 1-27. The Order stated that there was a substantial new question of patentability affecting claims 1-27 of the 302 Patent.

SCOPE OF REEXAMINATION

Although the Order stated there was SNQ affecting claims 1-27. However, the Office's determination in this Non-Final Office action will be limited **solely to a review of claims 1, 5-8, 13, 17, 21-22 and 25-26** for which reexamination was requested based on Kalyanasundharam and Kennedy (Ground 1). Grounds 2-3 proposed including dependent claims 2-4, 9-12, 14-16, 18-20, 23-24 and 27 were not accepted in the Order, and the Request did not assert the existence of a substantial new question of patentability (SNQP) for dependent claims 2-4, 9-12, 14-16, 18-20, 23-24 and 27 based on Kalyanasundharam and Kennedy. See: MPEP 2240 & 2243. If the Request fails to set forth the pertinency and manner of applying the cited art to a claim

as required by 37 CFR 1.510(b), that claim will generally not be reexamined. This matter was squarely addressed in Sony Computer Entertainment America Inc. v. Dudas, 85 USPQ2d 1594 (E.D. Va 2006). Thus, **claims 2-4, 9-12, 14-16, 18-20, 23-24 and 27 will not be reexamined in this *ex parte* reexamination proceeding**.

STATUS OF CLAIMS

Claims 1, 5-8, 13, 17, 21-22 and 25-26 are original. Of these, claims 1, 13, 21 and 26 are independent claims.

CLAIM INTERPRETATION

During examination, claims are given the broadest reasonable interpretation consistent with the specification and limitations in the specification are not read into the claims. See MPEP §2111, MPEP §2111.01 and *In re Yamamoto et al.*, 222 USPQ 934 (Fed. Cir. 1984). Under a broadest reasonable interpretation, words of the claim must be given their plain meaning, unless such meaning is inconsistent with the specification. See MPEP §2111.01(I). It is further noted it is improper to import claim limitations from the specification, i.e., a particular embodiment appearing in the written description may not be read into a claim when the claim language is broader than the embodiment. See MPEP §2111.01(II). Therefore, unless one of the exceptions applies below, the Examiners will interpret the limitations of the pending and examined claims using the broadest reasonable interpretation.

A. Lexicographic Definitions

A first exception to the prohibition of reading limitations from the specification into the claims is when the Applicant for patent has provided a lexicographic definition for the term. See MPEP §2111.01(IV). Following an independent review of the claims in view of the specification herein, the Examiners find that Patent Owner has not provided any lexicographic definitions related to claim terms with any reasonable clarity, deliberateness and precision.

B. Claim Interpretation Under 35 U.S.C. §112(6th ¶)

A second exception to the prohibition of reading limitations from the specification into the claims is when the claimed feature is written as a means-plus-function or a step-plus-function. See MPEP §2181-2183. As noted in MPEP §2181, a three prong test is used to determine the scope of a means-plus-function or step-plus-function limitation in a claim:

(A) the claim limitation uses the term "means" or "step" or a term used as a substitute for "means" that is a generic placeholder (also called a nonce term or a non-structural term having no specific structural meaning) for performing the claimed function

(B) the term "means" or "step" or the generic placeholder is modified by functional language, typically, but not always linked by the transition word "for" (e.g., "means for") or another linking word or phrase, such as "configured to" or "so that"

(C) the term "means" or "step" or the generic placeholder is not modified by sufficient structure, material, or acts for performing the claimed function.

B1. FL #1: "system firmware means for detecting ..." (Claim 26)

A first means-plus-function phrase is recited in claim 26, which recites "system firmware means for detecting..." or hereinafter FL #1. The Examiners determine herein that FL #1 meets the three prong test and thus will be interpreted as a means-plus-function limitation under 35 U.S.C. §112 ¶6th.

The Examiners find that FL #1 in claim 26 recites:

"*system firmware means for detecting loss of lockstep (LOL) for a processor in a multi-processor system*".

(B1)(a) 3-Prong Analysis: Prong (A)

FL #1 meets invocation prong (A) because "means ... for" language is recited and thus there is a presumption that interpretation under §112 ¶6th is invoked.

(B1)(b) 3-Prong Analysis: Prong (B)

FL #1 meets invocation prong (B) because it recites the function "*for detecting loss of lockstep (LOL) for a processor in a multi-processor system*".

(B1)(c) 3-Prong Analysis: Prong (C)

FL #1 meets invocation prong (C) because FL #1 does *not* recite sufficient structure for performing the claimed function. Based upon a review of claim 26, the Examiners find that FL #1 recites very little structure for performing the function as set forth of FL #1.

In view of the Examiners findings above that FL #1 meets invocation prongs (A)-(C), the Examiners conclude FL #1 invokes interpretation under 35 U.S.C. §112 ¶6th.

(B1)(d) Corresponding Structure

After a claimed phrase has been shown to invoke 35 U.S.C. §112 ¶6th as found above, the next step is to determine the corresponding structure or material as described in the specification for performing the recited function. See MPEP §2181(II).

Based upon a review of the specification of the 302 Patent, the Examiners find that the closest corresponding disclosed structure for performing the claimed function is a master processor 12A, a slave processor 12B and an error detect 13C with supporting firmware (18A) including a Processor Abstraction Layer (201), a System Abstraction Layer (202) shown in FIG. 2 of the 302 Patent, and an Extended Firmware Interface disclose in col. 8, line 40 of the 302 Patent, reprinted below. The software modules for performing the recite functions in association with this hardware are described in Figs. 5-6 in the 302 Patent.

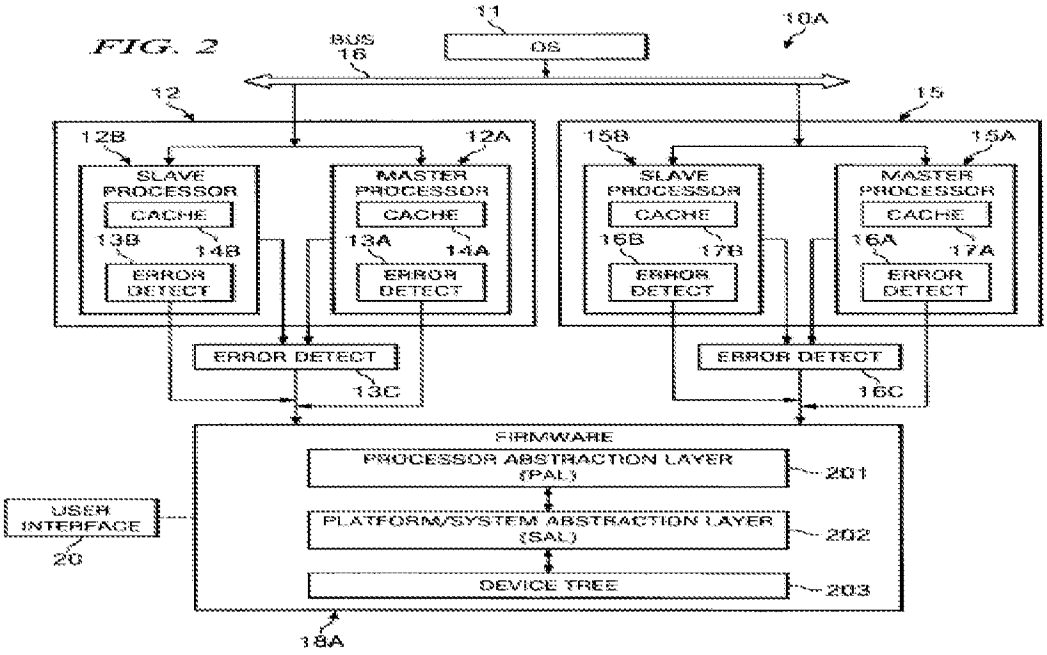
The 302 Patent states:

> *" In this example, master processor 12A includes error detect logic 13A, and slave processor 12B includes error detect logic 13B. While shown as included in each of the processors 12A and 12B in this example, in certain embodiments the error detect logic 13A and 13B may be implemented external to processors 12A and 12B. Error detect logic 13A and 13B*

include logic for detecting errors, such as data cache errors, present in their respective processors 12A and 12B. Examples of error detect logic 13A and 13B include known parity-based mechanisms and ECC mechanisms. Error detect logic 13C is also included, which may include an XOR (exclusive OR) gate, for detecting a lockstep mismatch between master processor 12A and slave processor 12B. As mentioned above, a lockstep mismatch refers to the output of master processor 12A and slave processor 12B failing to match. While shown as external to the lockstep processor pair 12 in this example, in certain embodiments error detect logic 13C may be implemented on a common silicon chip with processors 12A and 12B. Lockstep mismatch is one way of detecting a LOL between the master processor 12A and slave processor 12B. A detection of an error by either of error detect logic 13A and 13B also provides detection of LOL in the processors 12A and 12B. Because the detection of LOL by error detect logic 13A and 13B may occur before an actual lockstep mismatch occurs, the detection of LOL by error detect logic 13A and 13B may be referred to as a detection of a "precursor to lockstep mismatch". In other words, once an error (e.g., corrupt data) is detected by error detect logic 13A or 13B, such error may eventually propagate to a lockstep mismatch error that is detectable by error detect logic 13C." (col. 7:6-36)

"*Processor architecture generally comprises corresponding supporting firmware, such as firmware 18 of system 10. For example, as described further below in conjunction with the specific example of FIG. 2, the IPF processor architecture comprises such Supporting firmware as Processor Abstraction Layer (PAL), System Abstraction Layer (SAL), and Extended Firmware Interface (EFI). Such supporting firmware may enable, for example, the OS to access a particular function implemented for the processor... Accordingly, the Supporting firmware. Such as the PAL, is implemented to provide an interface to the processor(s) for accessing the functionality provided by such processor(s). Each of those interfaces*

provide standard, published procedure calls that are Supported." (col. 8:34-59)



302 Patent FIG. 2

In view of these findings, pursuant to 35 U.S.C. §112 ¶ 6th, FL #1 in claim 26 will be limited to the corresponding structures discussed above in conjunction with the appropriate software modules or programming noted above for detecting loss of lockstep (LOL) for a processor in the multi-processor system.

If applicant does not intend to have this limitation, FL #1 in claim 26 interpreted under 35 U.S.C. §112 ¶6th applicant may: (1) amend the claim limitation(s) to avoid it/them being interpreted under 35 U.S.C. §112 ¶6th (e.g.,

by reciting sufficient structure to perform the claimed function); or (2) present a sufficient showing that the claim limitation recites sufficient structure to perform the claimed function so as to avoid it/them being interpreted under 35 U.S.C. §112 ¶6th.

B2. FL #2: "system firmware means for determining ..." (Claim 26)

A second means-plus-function phrase is recited in claim 26, which recites "system firmware means for determining ..." or hereinafter FL #2. The Examiners determine herein that FL #2 meets the three prong test and thus will be interpreted as a means-plus-function limitation under 35 U.S.C. §112 ¶6th.

The Examiners find that FL #2 in claim 26 recites:

"*system firmware means for determining whether the processor for which said LOL is detected is assigned the role of boot processor*".

(B2)(a) 3-Prong Analysis: Prong (A)

FL #2 meets invocation prong (A) because "means ... for" language is recited and thus there is a presumption that interpretation under §112 ¶6th is invoked.

(B2)(b) 3-Prong Analysis: Prong (B)

FL #2 meets invocation prong (B) because it recites the function "*for determining whether the processor for which said LOL is detected is assigned the role of boot processor*".

(B2)(c) 3-Prong Analysis: Prong (C)

FL #2 meets invocation prong (C) because FL #2 does *not* recite sufficient structure for performing the claimed function. Based upon a review of claim 26, the Examiners find that FL #2 recites very little structure for performing the function as set forth of FL #2.

In view of the Examiners findings above that FL #2 meets invocation prongs (A)-(C), the Examiners conclude FL #2 invokes interpretation under 35 U.S.C. §112 ¶6th.

(B2)(d) Corresponding Structure

After a claimed phrase has been shown to invoke 35 U.S.C. §112 ¶6th as found above, the next step is to determine the corresponding structure or material as described in the specification for performing the recited function. See MPEP §2181(II).

Based upon a review of the specification of the 302 Patent, the Examiners find that the closest corresponding disclosed structure for performing the claimed function is a device tree (203) in the firmware 18A, a Boot Processor 12 and a Spare Processor 15 for accessing the processor

designators in the device tree (203) is shown in FIG. 2 of the 302 Patent, reprinted below. The software modules for performing the recite functions in association with this hardware are described in Figs. 5-6 in the 302 Patent.

The 302 Patent states:

> "*As described further herein, in certain embodiments, a device tree is stored to a data storage device (e.g., SRAM) such that it is accessible to the firmware 18, wherein the device tree includes information identifying processor 12 as the system's boot processor and identifying processor 15 as the spare. The information specifying the boot processor and/or square may be received as input from a user in certain embodiment.*" (col. 9:4-11)
>
> "*Further, device tree 203 includes information designating which processor is the boot processor, as well as a spare processor for the boot processor.*" (col. 13:21-23)

The 302 Patent further discloses:

> "*In operational : block 501, the system firmware, responsive to such detection of LOL for the boot processor, examines the device tree to determine which processor module is assigned the role of active spare. In operational block 502 the system firmware (e.g., SAL) asserts a General Purpose Event Interrupt (GPE) by writing to the appropriate GPE register in the ACPI register space. This causes the OS to execute its interrupt handler, which will run an AML Method associated with this particular interrupt. In this example, this AML method that is executed responsive to the interrupt generated by the GPE includes an AML instruction called "Notify". This "Notify" operation indicates the determined active spare processor object and that an Eject is being requested as arguments. Accordingly, an AML method is used to request the OS to eject the active*

spare that was determined in block 501. In response, the OS idles the active spare processor and returns control over the active spare processor to the system firmware, in block 503." (col. 17:15-33)

"In operational block 601, LOL is detected for a processor in a multi-processor system. In operational block 602, the system (e.g., system firmware) determines whether the processor for which LOL is detected is the system's boot processor. As described above, this is determined in certain embodiments by accessing the node of the device tree corresponding to the processor for which LOL is detected and determining if the information in that node of the device tree indicates that the processor is the system's boot processor. (col. 18: 50-58)

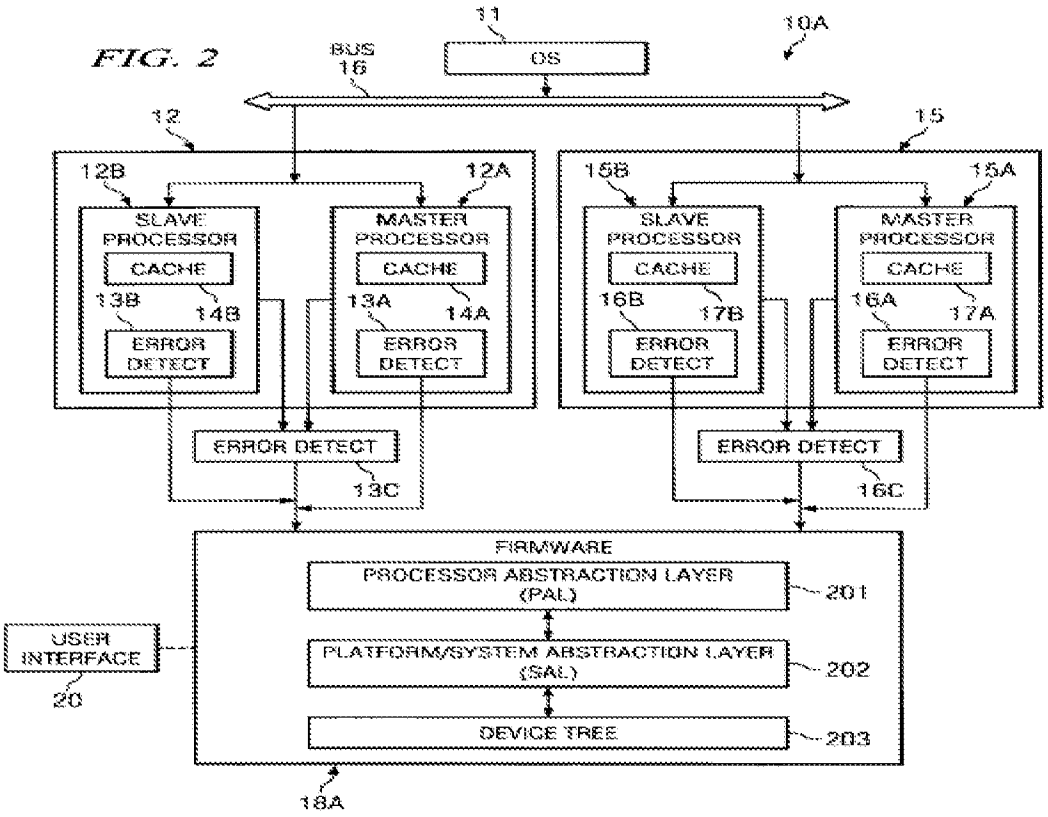


Fig. 2 of the 302 Patent

FIG. 5

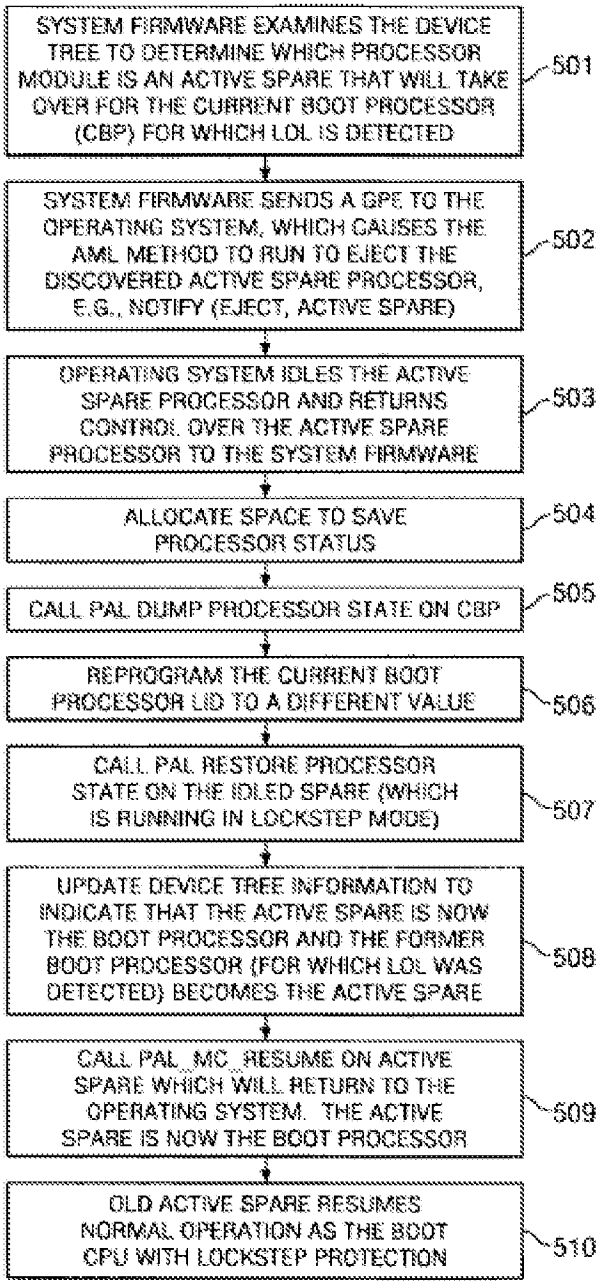
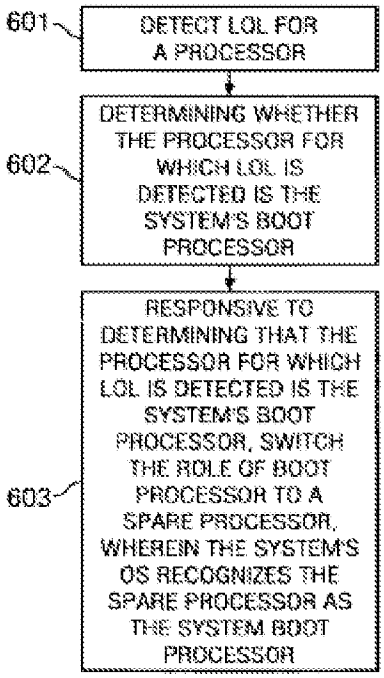


FIG. 6



Figs. 5-6 of the 302 Patent

5

In view of these findings, pursuant to 35 U.S.C. §112 ¶ 6th, FL #2 in claim 26 will be limited to the corresponding structures discussed above in conjunction with the appropriate software modules or programming noted above for determining whether the processor for which the LOL is detected and assigned the role of boot processor.

If applicant does not intend to have this limitation, FL #2 in claim 26 interpreted under 35 U.S.C. §112 ¶6th applicant may: (1) amend the claim limitation(s) to avoid it/them being interpreted under 35 U.S.C. §112 ¶6th (e.g., by reciting sufficient structure to perform the claimed function); or (2) present a sufficient showing that the claim limitation recites sufficient structure to perform the claimed function so as to avoid it/them being interpreted under 35 U.S.C. §112 ¶6th.

B3. FL #3: "system firmware means for switching ..." (Claim 26)

A third means-plus-function phrase is recited in claim 26, which recites "system firmware means for switching ..." or hereinafter FL #3. The Examiners determine herein that FL #3 meets the three prong test and thus will be interpreted as a means-plus-function limitation under 35 U.S.C. §112 ¶6th.

The Examiners find that FL #3 in claim 26 recites:

"*system firmware means for switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which said LOL is detected is assigned the role of boot processor.*"

(B3)(a) 3-Prong Analysis: Prong (A)

FL #3 meets invocation prong (A) because "means ... for" language is recited and thus there is a presumption that interpretation under §112 ¶6th is invoked.

(B3)(b) 3-Prong Analysis: Prong (B)

FL #3 meets invocation prong (B) because it recites the function "*for switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which said LOL is detected is assigned the role of boot processor.*"

(B3)(c) 3-Prong Analysis: Prong (C)

FL #3 meets invocation prong (C) because FL #3 does *not* recite sufficient structure for performing the claimed function. Based upon a review of claim 26, the Examiners find that FL #3 recites very little structure for performing the function as set forth of FL #3.

In view of the Examiners findings above that FL #3 meets invocation prongs (A)-(C), the Examiners conclude FL #3 invokes interpretation under 35 U.S.C. §112 ¶6th.

(B3)(d) Corresponding Structure

After a claimed phrase has been shown to invoke 35 U.S.C. §112 ¶6th as found above, the next step is to determine the corresponding structure or material as described in the specification for performing the recited function. See MPEP §2181(II).

Based upon a review of the specification of the 302 Patent, the Examiners find that the corresponding disclosed structure for performing the claimed function is the Boot Processor (12), the Spare Processor (15), the error detects 13C, 16C and supporting firmware 18A including a Processor Abstraction Layer (201), a System Abstraction Layer (202) shown in FIG. 2 of the 302 Patent, and an Extended Firmware Interface discloses in col. 8, line 40 of the 302 Patent, reprinted below. The software modules for performing the recite functions in association with the hardware are described in FIGS. 1 and 4-5 in the 302 Patent.

The 302 Patent states:

> *"Processor architecture generally comprises corresponding supporting firmware, such as firmware 18 of system 10. For example, as described further below in conjunction with the specific example of FIG. 2, the IPF processor architecture comprises such supporting firmware as Processor Abstraction Layer (PAL), System Abstraction Layer (SAL), and Extended Firmware Interface (EFI). Such supporting firmware may enable, for example, the OS to access a particular function implemented for the processor. For instance, the OS may query the PAL as to the size of the cache implemented for the processor, etc. Other well-known functions provided by the supporting firmware (SAL, EFT) include, for example: (a) performing I/O configuration accesses to discover and program the I/O Hardware (SAL_PCILCONFIG_READ and SAL_PCI_CONFIG_WRITE); (b)*

retrieving error log data from the platform following a Machine Check Abort (MCA) event (SAL_GET_STATE_INFO); (c) accessing persistent store configuration data stored in non-volatile memory (EFI variable services: GetNextVariableName, GetVariable and SetVariable); and accessing the battery-backed real-time clock/calendar (EFI GetTime and SetTime). Accordingly, the supporting firmware, such as the PAL, is implemented to provide an interface to the processor(s) for accessing the functionality provided by such processor(s). Each of those interfaces provide standard, published procedure calls that are supported." (col. 8:34-59)

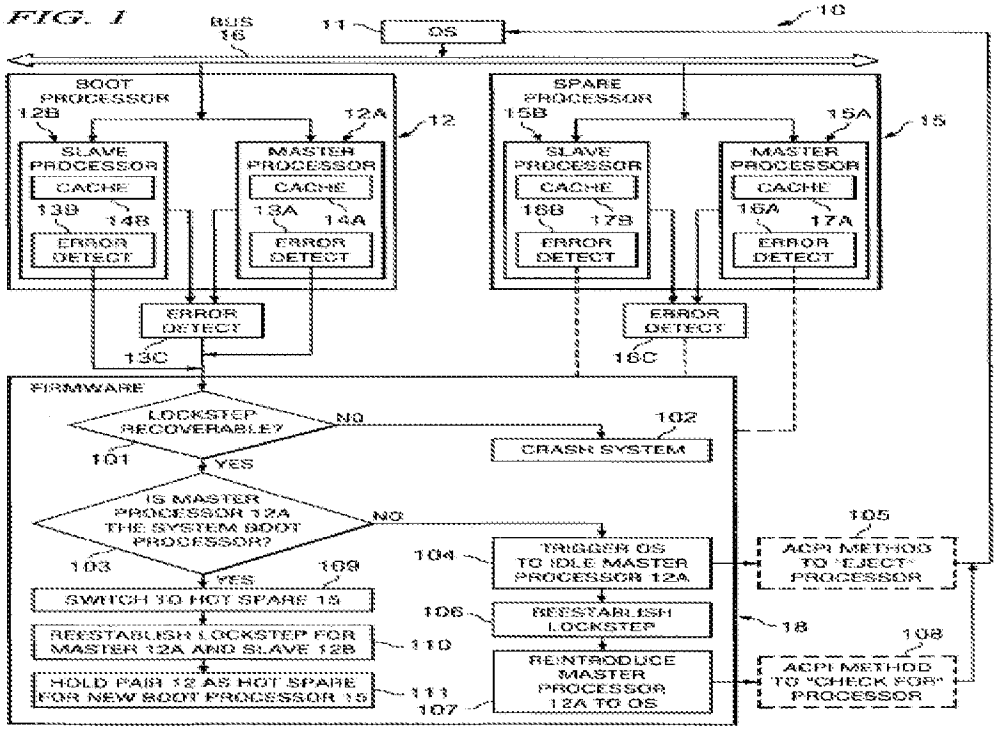
"For instance, in the example of FIG. 1, if determined in block 103 that the master processor 12A is the system's boot processor, operation advances to block 109 whereat the state of the "good" processor in the lockstep processor pair 12 is copied over to each of processors 15A and 15B in the spare lockstep processor pair 15. Exemplary techniques that may be utilized by firmware 18 for switching the role of boot processor to the hot spare are described further herein below." (col. 11:11-18)

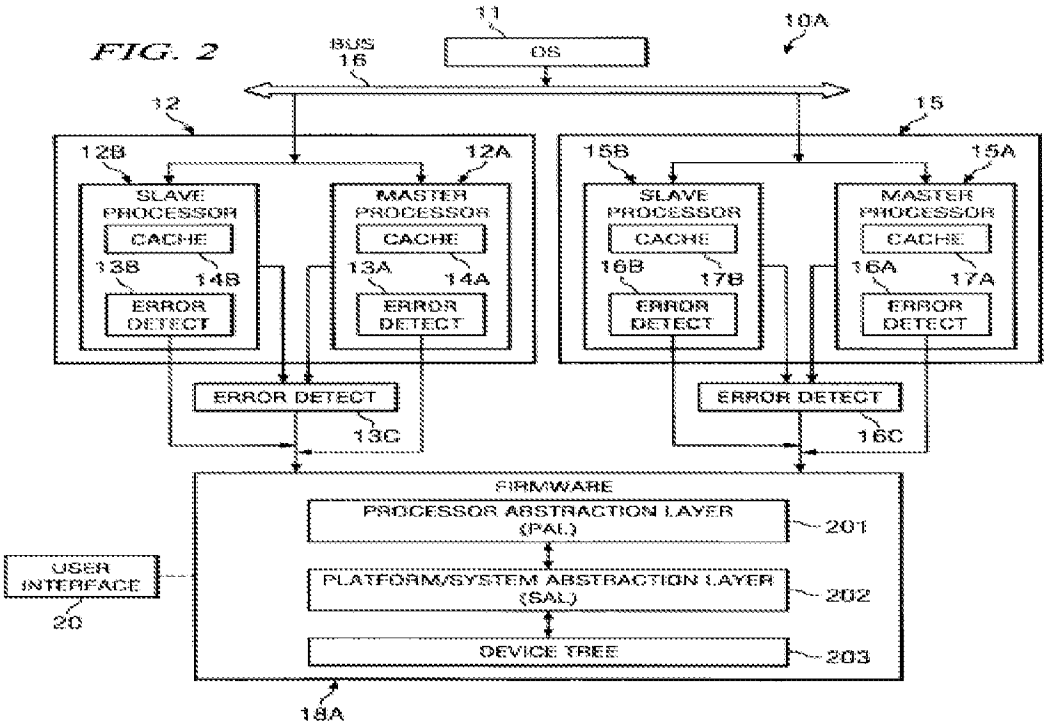
"In operational block 403, the LID of the current boot processor (for which LOL was detected) is reprogrammed to a different value. For instance, the current boot processor may be assigned a LID that is reserved just for this situation (i.e., for switching the role of boot processor). Accordingly, the boot processor for which LOL was detected will not respond to the interrupts addressed for its original LID because the boot processor's LID value is reprogrammed to a different value." (col. 16, 27-35)

"Turning to FIG. 5, an exemplary operational flow diagram for switching the role of boot processor to an active spare processor according to one embodiment is shown. Again, an LOL has been detected for the boot processor. In operational block 501, the system firmware, responsive to

such detection of LOL for the boot processor, examines the device tree to determine which processor module is assigned the role of active spare. In operational block 502 the system firmware (e.g., SAL) asserts a General Purpose Event Interrupt (GPE) by writing to the appropriate GPE register in the ACPI register space. This causes the OS to execute its interrupt handler, which will run. an AML Method associated with this particular interrupt. In this example, this AML method that is executed responsive to the interrupt generated by the GPE includes an AML instruction called "Notify". This "Notify" operation indicates the determined active spare processor object and that an Eject is being requested as arguments. Accordingly, an AML method is used to request the OS to eject the active spare that was determined in block 501. In response, the OS idles the active spare processor and returns control over the active spare processor to the system firmware, in block 503." (col. 17:12-33)

"In operational block 603, responsive to determining that the processor for which LOL is detected is the system's boot processor, the system's firmware switches the role of boot processor to a spare processor, which may be either a hot spare or an active spare, as described above. Accordingly, the system's OS recognizes the spare processor as the system boot processor. In view of the above, embodiments hereof enable the role of boot processor to be dynamically switched, during system runtime, to a spare processor responsive to a detected LOL for the boot processor, thus enabling recovery from the LOL without requiring that the system be crashed or re-booted." (col. 18:59- col. 19:3)





FIGs. 1-2 of the 302 Patent.

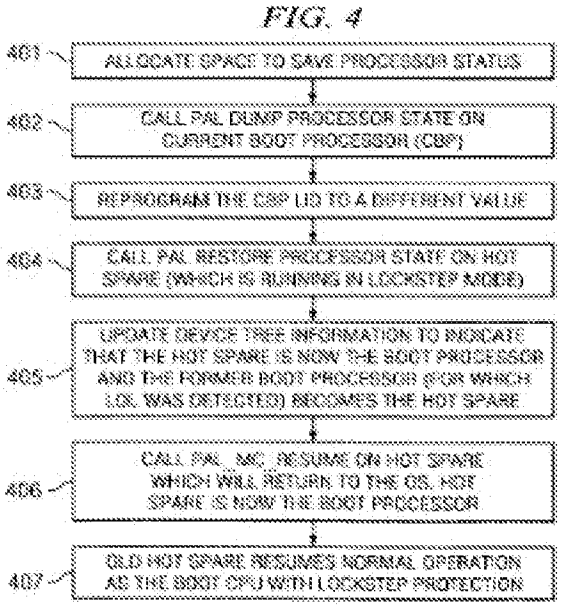


Fig. 4 of the 302 Patent

FIG. 5

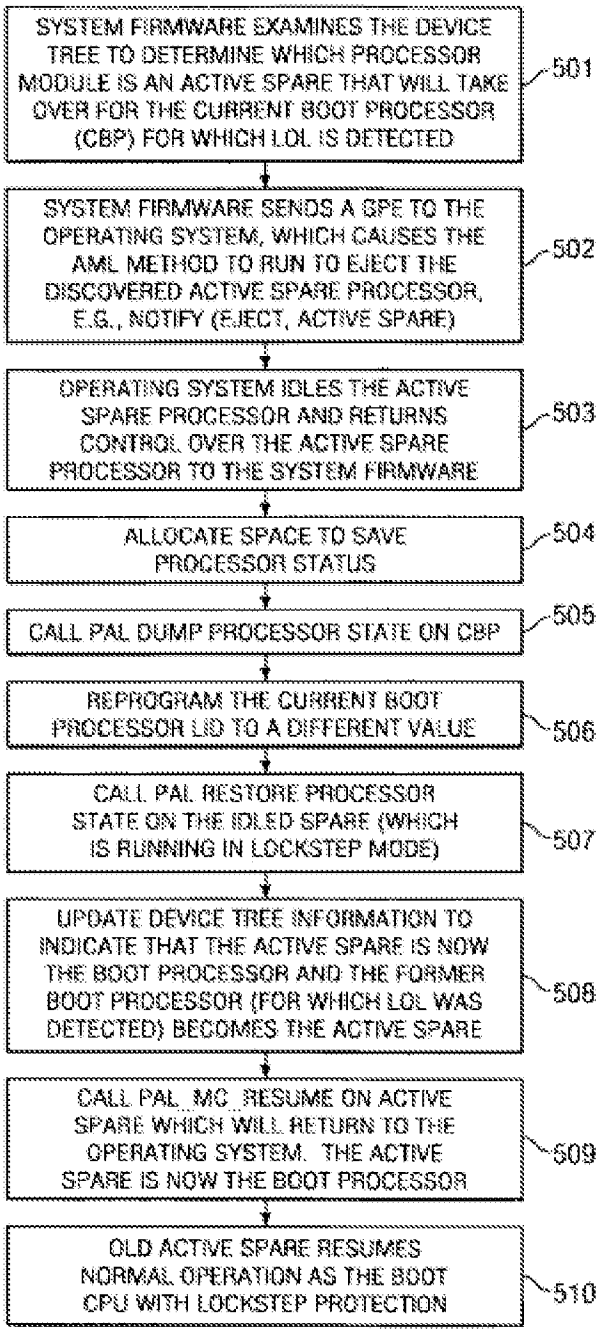
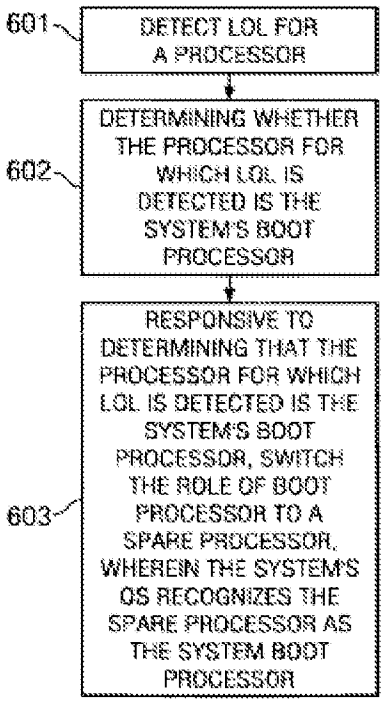


FIG. 6



Figs. 5-6 of the 302 Patent

In view of these findings, pursuant to 35 U.S.C. §112 ¶ 6th, FL #3 in claim 26 will be limited to the corresponding structures discussed above in conjunction with the appropriate software modules or programming noted above for switching the role of boot processor to the spare processor without shutting down the system's operating system if determined that the processor for which the LOL is detected and assigned the role of boot processor.

If applicant does not intend to have this limitation, FL #3 in claim 26 interpreted under 35 U.S.C. §112 ¶6th applicant may: (1) amend the claim limitation(s) to avoid it/them being interpreted under 35 U.S.C. §112 ¶6th (e.g., by reciting sufficient structure to perform the claimed function); or (2) present a sufficient showing that the claim limitation recites sufficient structure to perform the claimed function so as to avoid it/them being interpreted under 35 U.S.C. §112 ¶6th.

Furthermore, Examiners find herein that claim 21 include one or more claim limitations that do not use the word "means," but are nonetheless being interpreted under 35 U.S.C. §112 ¶6th because the claim limitations uses a generic placeholder that is coupled with functional language without reciting sufficient structure to perform the recited function and the generic placeholder is not preceded by a structural modifier. Each such limitation will be discussed in turn as follows:

B4. FL #4: "error detection logic" (Claim 21)

A fourth means-plus-function phrase is recited in claim 21 (and dependent claims 22 and 25), which recites "error detection logic" or hereinafter FL #4. The Examiners determine herein that FL #4 meets the three prong test and thus will be interpreted as a means-plus-function limitation under 35 U.S.C. §112 ¶6th.

The Examiners find that FL #4 in claim 21 recites:

"*error detection logic operable to detect loss of lockstep (LOL) for at least one of said processor modules*".

(B4)(a) 3-Prong Analysis: Prong (A)

FL #4 meets invocation prong (A) because "means ... for" language is recited. The Examiners find that "error detection logic" as used in claim 21 is a generic placeholder or nonce term equivalent to "means" because the phrase does not convey any particular sufficient structure for performing the recited functions.

Furthermore, there is no disclosure or suggestion from the prior art or the 302 Patent that a "error detection logic" is sufficient structure to perform the functions recited in FL #4. Rather, the prior art uses the phrase "error detection logic" in association with numerous structures and perform numerous functions which are distinct from those functions in FL #4. For example, U.S. Patent No. 7,370,243, claim 15 recites "error detection logic" comprising various means and functions; U.S. Publication No. 2006/0020850

claim 1 recites "*error detection logic configured to scan the memory regions for a latent processing error maintained in a memory region as an erroneous data output from a processor element*"; U.S. Patent No. 6,882,620, claim 2 recites "*error detection logic configured to monitor at least one of the plurality of clients for an anomaly in the operation thereof*". The Examiners further find these error detection logics have different structures and functions than those recited in FL #4. Accordingly, based on the diverse functional uses of "error detection logic" which would have diverse structures to perform these functions, the Examiners find that "error detection logic" does not imply or require a specific structure across these references and FL #4.

Accordingly, the Examiners find nothing in the specification, prosecution history or the prior art to construe "error detection logic" in FL #4 as the name of a sufficiently definite structure for performing the functions recited in FL #4 so as to take the overall claim limitation out of the ambit of §112(f). *See Williamson v. Citrix Online, L.L.C.,* 115 USPQ2d 1105, 1112 (Fed. Cir. 2015).

In light of the above findings, the Examiners conclude that the term "error detection logic" is a generic placeholder having no specific structure associated therewith. Because "error detection logic" is merely a generic placeholder having no specific structure associated therewith, the Examiners conclude that FL #4 meets invocation Prong (A).

(B4)(b) 3-Prong Analysis: Prong (B)

FL #4 meets invocation prong (B) because it recites the function "*operable to detect loss of lockstep (LOL) for at least one of said processor modules.*"

(B4)(c) 3-Prong Analysis: Prong (C)

FL #4 meets invocation prong (C) because FL #4 does *not* recite sufficient structure for performing the claimed function. Based upon a review of claim 21, the Examiners find that FL #4 recites very little structure, if any, for performing the function as set forth of FL #4.

In view of the Examiners findings above that FL #4 meets invocation prongs (A)-(C), the Examiners conclude FL #4 invokes interpretation under 35 U.S.C. §112 ¶6th.

(B4)(d) Corresponding Structure

After a claimed phrase has been shown to invoke 35 U.S.C. §112 ¶6th as found above, the next step is to determine the corresponding structure or material as described in the specification for performing the recited function. See MPEP §2181(II).

Based upon a review of the specification of the 302 Patent, the Examiners find that the corresponding hardware structure for the FL#4, i.e. the error detection logic, is the error detection logics 13A-13C, 16A-16C shown in Fig. 1 of the 302 Patent.

The 302 Patent discloses at col. 7:6-44:

"*In this example, master processor 12A includes error detect logic 13A, and slave processor 12B includes error detect logic 13B. While shown as included in each of the processors 12A and 12B in this example, in certain embodiments the error detect logic 13A and 13B may be implemented external to processors 12A and 12B. Error detect logic 13A and 13B include logic for detecting errors, such as data cache errors, present in their respective processors 12A and 12B. Examples of error detect logic 13A and 13B include known parity-based mechanisms and ECC mechanisms. Error detect logic 13C is also included, which may include an XOR (exclusive OR) gate, for detecting a lockstep mismatch between master processor 12A and slave processor 12B. As mentioned above, a lockstep mismatch refers to the output of master processor 12A and slave processor 12B failing to match. While shown as external to the lockstep processor pair 12 in this example, in certain embodiments error detect logic 13C may be implemented on a common silicon chip with processors 12A and 12B.*

Lockstep mismatch is one way of detecting a LOL between the master processor 12A and slave processor 12B. A detection of an error by either of error detect logic 13A and 13B also provides detection of LOL in the processors 12A and 12B. Because the detection of LOL by error detect logic 13A and 13B may occur before an actual lockstep mismatch occurs, the detection of LOL by error detect logic 13A and 13B may be referred to as a detection of a "precursor to lockstep mismatch". In other words, once an error (e.g., corrupt data) is detected by error detect logic 13A or 13B, such error may eventually propagate to a lockstep mismatch error that is detectable by error detect logic 13C.

System 10 further includes a second lockstep processor pair 15 that is communicatively coupled to bus 16. Lockstep processor pair 15 includes master processor 15A and slave processor 15B. Master processor 15A includes cache 17A, and slave processor 15B includes cache 17B. As with error detect logic 13A-13C implemented for lockstep processor pair 12, lockstep processor pair 15 has error detect logic 16A-16C."

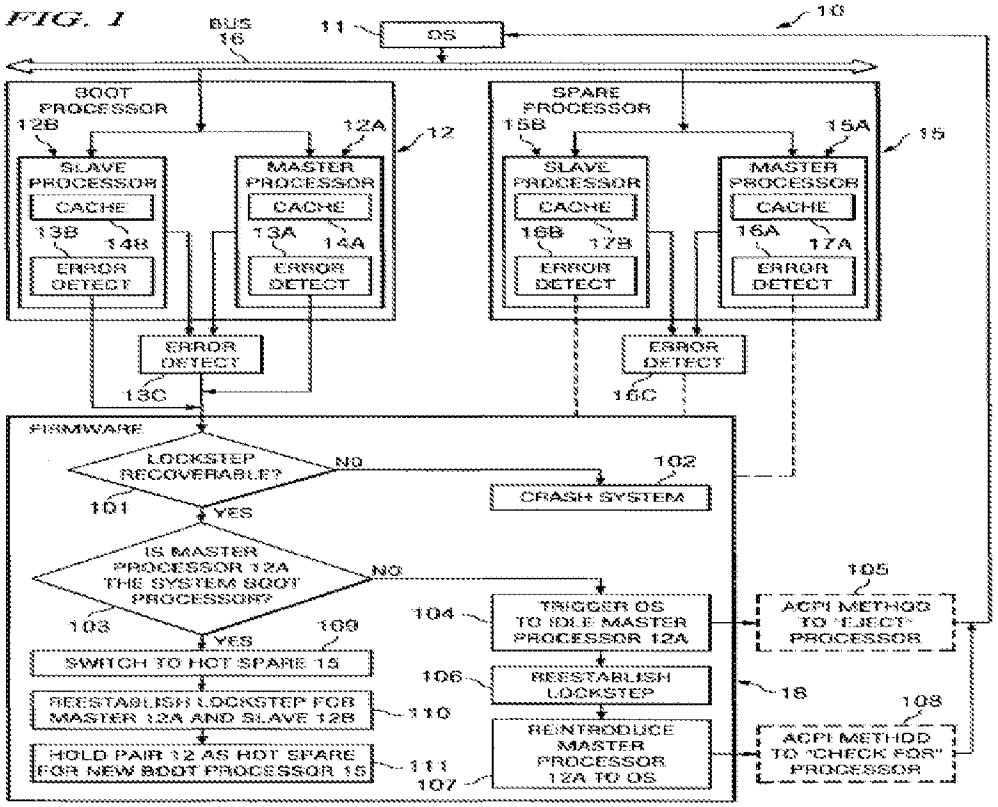


Fig. 1 of 302 Patent

In view of these findings, pursuant to 35 U.S.C. §112 ¶ 6th, FL #4 in claim 21 will be limited to the corresponding structures discussed above.

If applicant does not intend to have this limitation, FL #4 in claim 21 interpreted under 35 U.S.C. §112 ¶6th applicant may: (1) amend the claim limitation(s) to avoid it/them being interpreted under 35 U.S.C. §112 ¶6th (e.g., by reciting sufficient structure to perform the claimed function); or (2) present a sufficient showing that the claim limitation recites sufficient structure to perform the claimed function so as to avoid it/them being interpreted under 35 U.S.C. §112 ¶6th.

B5. FL #5: "system firmware" (Claim 21)

A fifth means-plus-function phrase is recited in claim 21, which recites "system firmware" or hereinafter FL #5. The Examiners determine herein that FL #5 meets the three prong test and thus will be interpreted as a means-plus-function limitation under 35 U.S.C. §112 ¶6th.

The Examiners find that FL #5 in claim 21 recites:

"*system firmware operable, responsive to the detection of LOL for a first of said processor modules, to determine whether said first processor module is assigned a role of system boot processor; wherein if determined that said first processor module is assigned the role of system boot processor, said system firmware is further operable to cause said operating system to recognize another processor module as system boot processor without shutting down said operating system.*"

(B5)(a) 3-Prong Analysis: Prong (A)

FL #5 meets invocation prong (A) because "means ... for" language is recited. The Examiners find that "system firmware" as used in claim 21 is a generic placeholder or nonce term equivalent to "means" because the phrase does not convey any particular sufficient structure for performing the recited functions.

Furthermore, there is no disclosure or suggestion from the prior art or the 302 Patent that the "system firmware" is sufficient structure to perform the functions recited in FL #5. Rather, the prior art uses the phrase "system

firmware" in association with numerous structures and perform numerous functions which are distinct from those functions in FL #5. For example, U.S. Patent No. 5,524,245 discloses in col. 1:29-34 "*While manufacturers have designed components such as the CPU and I/O controllers that can be configured to operate in either byte-order mode, firmware is normally one mode or the other. Thus, the configurable components are normally statically hard-wired by the manufacturers to operate in the same byte order as the computer firmware*"; U.S. Patent No. 4,413,322 discloses in col. 12:56-66 "Firmware stored in memory 34" comprising various functions. The Examiners further find these firmwares have different structures and functions than those recited in FL #5. Accordingly, based on the diverse functional uses of "system firmware" which would have diverse structures to perform these functions, the Examiners find that "system firmware" does not imply or require a specific structure across these references and FL #5.

Accordingly, the Examiners find nothing in the specification, prosecution history or the prior art to construe "system firmware" in FL #5 as the name of a sufficiently definite structure for performing the functions recited in FL #5 so as to take the overall claim limitation out of the ambit of §112(f). *See Williamson v. Citrix Online, L.L.C.,* 115 USPQ2d 1105, 1112 (Fed. Cir. 2015).

In light of the above findings, the Examiners conclude that the term "system firmware" is a generic placeholder having no specific structure associated therewith. Because "system firmware" is merely a generic

placeholder having no specific structure associated therewith, the Examiners conclude that FL #5 meets invocation Prong (A).

(B5)(b) 3-Prong Analysis: Prong (B)

FL #5 meets invocation prong (B) because it recites the function: "*operable, responsive to the detection of LOL for a first of said processor modules, to determine whether said first processor module is assigned a role of system boot processor; wherein if determined that said first processor module is assigned the role of system boot processor, said system firmware is further operable to cause said operating system to recognize another processor module as system boot processor without shutting down said operating system.*"

(B5)(c) 3-Prong Analysis: Prong (C)

FL #5 meets invocation prong (C) because FL #5 does *not* recite sufficient structure for performing the claimed function. Based upon a review of claim 21, the Examiners find that FL #5 recites very little structure for performing the function as set forth of FL #5.

In view of the Examiners findings above that FL #5 meets invocation prongs (A)-(C), the Examiners conclude FL #5 invokes interpretation under 35 U.S.C. §112 ¶6th.

(B5)(d) Corresponding Structure

After a claimed phrase has been shown to invoke 35 U.S.C. §112 ¶6th as found above, the next step is to determine the corresponding structure or material as described in the specification for performing the recited function. See MPEP §2181(II).

Based upon a review of the specification of the 302 Patent, the Examiners find that the corresponding disclosed structure for performing the claimed function is the firmware 18A including a Processor Abstraction Layer (201), a System Abstraction Layer (202), a device tree (203) shown in FIG. 2 of the 302 Patent, reprinted below, and an Extended Firmware Interface discloses in col. 8, line 40 of the 302 Patent,. The software modules for performing the recite functions in association with the hardware are described in FIGS. 1 and 4-5 in the 302 Patent.

The 302 Patent states:

"Processor architecture generally comprises corresponding supporting firmware, such as firmware 18 of system 10. For example, as described further below in conjunction with the specific example of FIG. 2, the IPF processor architecture comprises such supporting firmware as Processor Abstraction Layer (PAL), System Abstraction Layer (SAL), and Extended Firmware Interface (EFI). Such supporting firmware may enable, for example, the OS to access a particular function implemented for the processor. For instance, the OS may query the PAL as to the size of the cache implemented for the processor, etc. Other well-known functions provided by the supporting firmware (SAL, EFT) include, for example: (a) performing I/O configuration accesses to discover and program the I/O Hardware (SAL_PCILCONFIG_READ and SAL_PCI_CONFIG_WRITE); (b) retrieving error log data from the platform following a Machine Check Abort

(MCA) event (SAL_GET_STATE_INFO); (c) accessing persistent store configuration data stored in non-volatile memory (EFI variable services: GetNextVariableName, GetVariable and SetVariable); and accessing the battery-backed real-time clock/calendar (EFI GetTime and SetTime). Accordingly, the supporting firmware, such as the PAL, is implemented to provide an interface to the processor(s) for accessing the functionality provided by such processor(s). Each of those interfaces provide standard, published procedure calls that are supported." (col. 8:34-59)

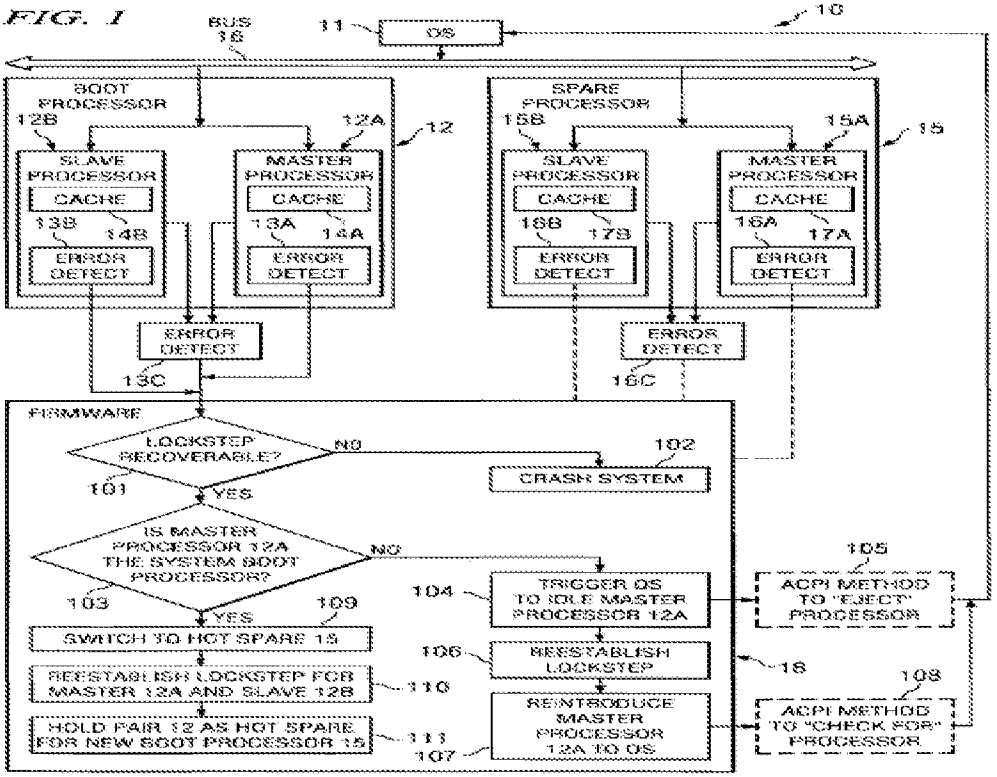
"For instance, in the example of FIG. 1, if determined in block 103 that the master processor 12A is the system's boot processor, operation advances to block 109 whereat the state of the "good" processor in the lockstep processor pair 12 is copied over to each of processors 15A and 15B in the spare lockstep processor pair 15. Exemplary techniques that may be utilized by firmware 18 for switching the role of boot processor to the hot spare are described further herein below." (col. 11:11-18)

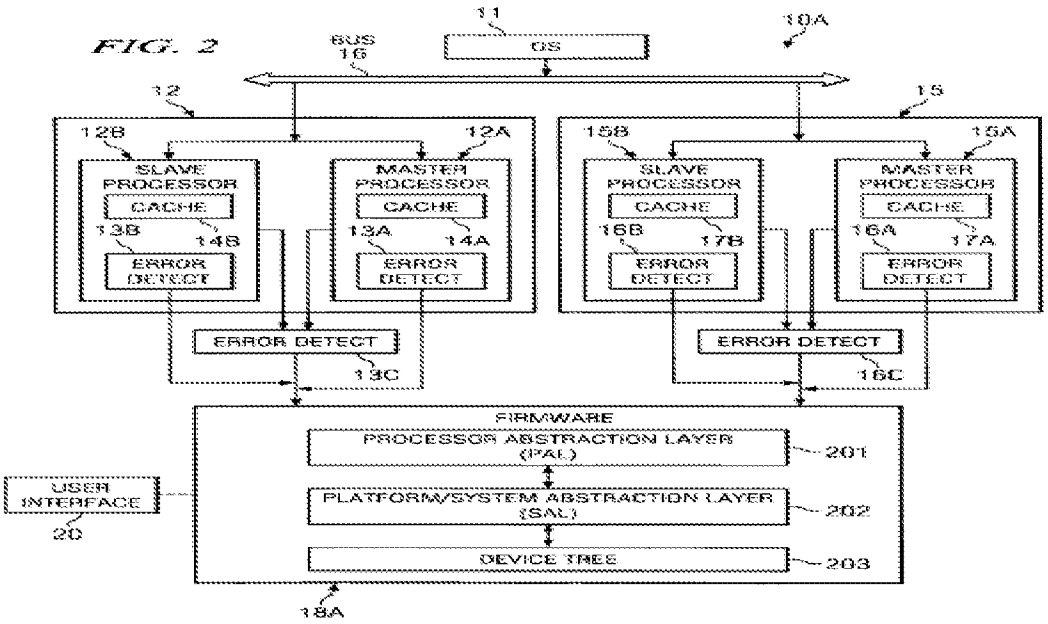
"In operational block 403, the LID of the current boot processor (for which LOL was detected) is reprogrammed to a different value. For instance, the current boot processor may be assigned a LID that is reserved just for this situation (i.e., for switching the role of boot processor). Accordingly, the boot processor for which LOL was detected will not respond to the interrupts addressed for its original LID because the boot processor's LID value is reprogrammed to a different value." (col. 16, 27-35)

"Turning to FIG. 5, an exemplary operational flow diagram for switching the role of boot processor to an active spare processor according to one embodiment is shown. Again, an LOL has been detected for the boot processor. In operational block 501, the system firmware, responsive to such detection of LOL for the boot processor, examines the device tree to

determine which processor module is assigned the role of active spare. In operational block 502 the system firmware (e.g., SAL) asserts a General Purpose Event Interrupt (GPE) by writing to the appropriate GPE register in the ACPI register space. This causes the OS to execute its interrupt handler, which will run. an AML Method associated with this particular interrupt. In this example, this AML method that is executed responsive to the interrupt generated by the GPE includes an AML instruction called "Notify". This "Notify" operation indicates the determined active spare processor object and that an Eject is being requested as arguments. Accordingly, an AML method is used to request the OS to eject the active spare that was determined in block 501. In response, the OS idles the active spare processor and returns control over the active spare processor to the system firmware, in block 503" (col. 17:12-33).

"In operational block 603, responsive to determining that the processor for which LOL is detected is the system's boot processor, the system's firmware switches the role of boot processor to a spare processor, which may be either a hot spare or an active spare, as described above. Accordingly, the system's OS recognizes the spare processor as the system boot processor. In view of the above, embodiments hereof enable the role of boot processor to be dynamically switched, during system runtime, to a spare processor responsive to a detected LOL for the boot processor, thus enabling recovery from the LOL without requiring that the system be crashed or re-booted." (col. 18:59- col. 19:3)





FIGs. 1-2 of the 302 Patent.

FIG. 5

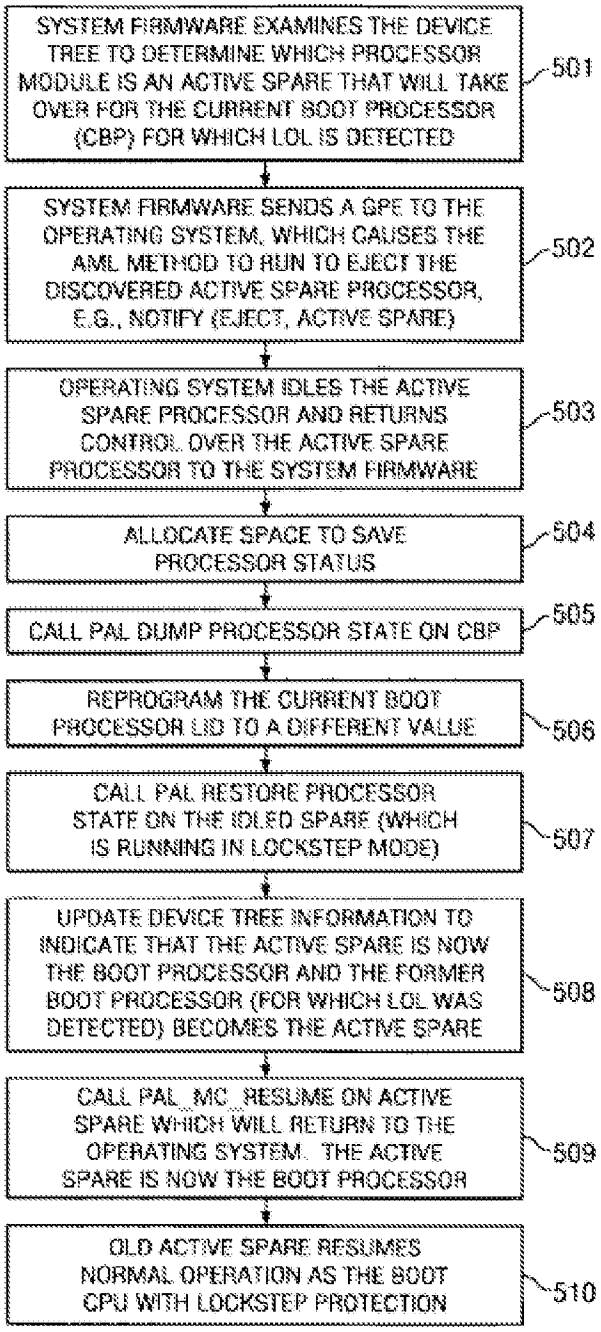
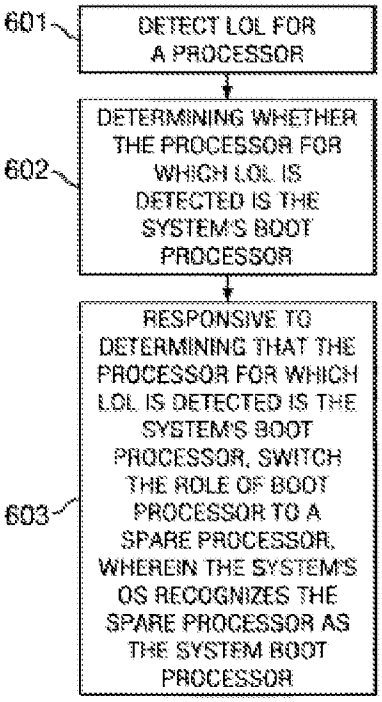


FIG. 6



Figs. 5-6 of the 302 Patent

In view of these findings, pursuant to 35 U.S.C. §112 ¶ 6th, FL #5 in claim 21 will be limited to the corresponding structures discussed above in conjunction with the appropriate software modules or programming noted above for switching the role of boot processor to the spare processor without shutting down the system if determined that the processor for which the LOL is detected and assigned the role of boot processor.

If applicant does not intend to have this limitation, FL #5 in claim 21 interpreted under 35 U.S.C. §112 ¶6th applicant may: (1) amend the claim limitation(s) to avoid it/them being interpreted under 35 U.S.C. §112 ¶6th (e.g., by reciting sufficient structure to perform the claimed function); or (2) present a sufficient showing that the claim limitation recites sufficient structure to perform the claimed function so as to avoid it/them being interpreted under 35 U.S.C. §112 ¶6th.

REFERENCES RELIED UPON OR DISCUSSED

In the request for reexamination, the Requester alleges that the 302 Patent reexamined claims 1, 5-8, 13, 17, 21, 22 and 25-26 are unpatentable in light of the following prior art references:

1. U.S. Patent No. 5,450,576 to Kennedy ("Kennedy").
2. U.S. Patent No. 7,328,371 to Kalyanasundharam ("Kalyanasundharam").

- It is noted that Kalyanasundharam and Kennedy were not of record in the prosecution of the 588 Application, later issuing as the 302 Patent.
- It is also noted that Safford (U.S. Patent No. 7,085,959) and Nguyen (U.S. App. No. 2004/0123201) proposed in the Request, Grounds 2 and 3, were not accepted in the Order. Therefore, Grounds 2 and 3 include Stafford and Nguyen will not listed and/or discussed further.

SUMMARY OF THE REQUESTER'S PROPOSED REJECTIONS

The following proposed rejection was accepted in the Order as raising SNQs for reexamined claims 1, 5-8, 13, 17, 21, 22 and 25-26 (Request, pages 23-33):

Ground 1 – Claims 1, 5-8, 13, 17, 21, 22 and 25-26 are obvious over Kalyanasundharam in view of Kennedy.

CLAIM REJECTIONS – RELEVANT STATUTE

In the event the determination of the status of the application as subject to AIA 35 U.S.C. 102 and 103 (or as subject to pre-AIA 35 U.S.C. 102 and 103) is incorrect, any correction of the statutory basis for the rejection will not be considered a new ground of rejection if the prior art relied upon, and the rationale supporting the rejection, would be the same under either status.

The following is a quotation of pre-AIA 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) *A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.*

DISCUSSION CLAIMS REJECTION

Ground 1: Kalyanasundharam and Kennedy for claims 1, 5-8, 13, 17, 21, 22 and 25-26 (pages 23-33 and 47-57). Of these:

- The rejection of claims 1, 5-8, 13 and 17 were proposed by the Requester in the request for reexamination, pages 23-33, 47-54 and are **ACCEPTED.**
- The rejection of claims 21-22 and 25-26 were proposed by the Requester in the request for reexamination, pages 54-57 and are **NOT ACCEPTED**.

Claims 1, 5-8, 13 and 17 are rejected under pre-AIA 35 U.S.C. 103(a) as being unpatentable over Kalyanasundharam in combination with Kennedy.

Regarding claim 1:

A method comprising:

Kalyanasundharam teaches in claims 16-19 and Fig. 6 a method of processor: "*FIG. 6 is a flowchart illustrating operation of one embodiment of the node controller including the redundancy check circuit shown in FIG. 5 in*

response to receiving a communication from a processor core."

(Kalyanasundharam at col. 2:7-10):

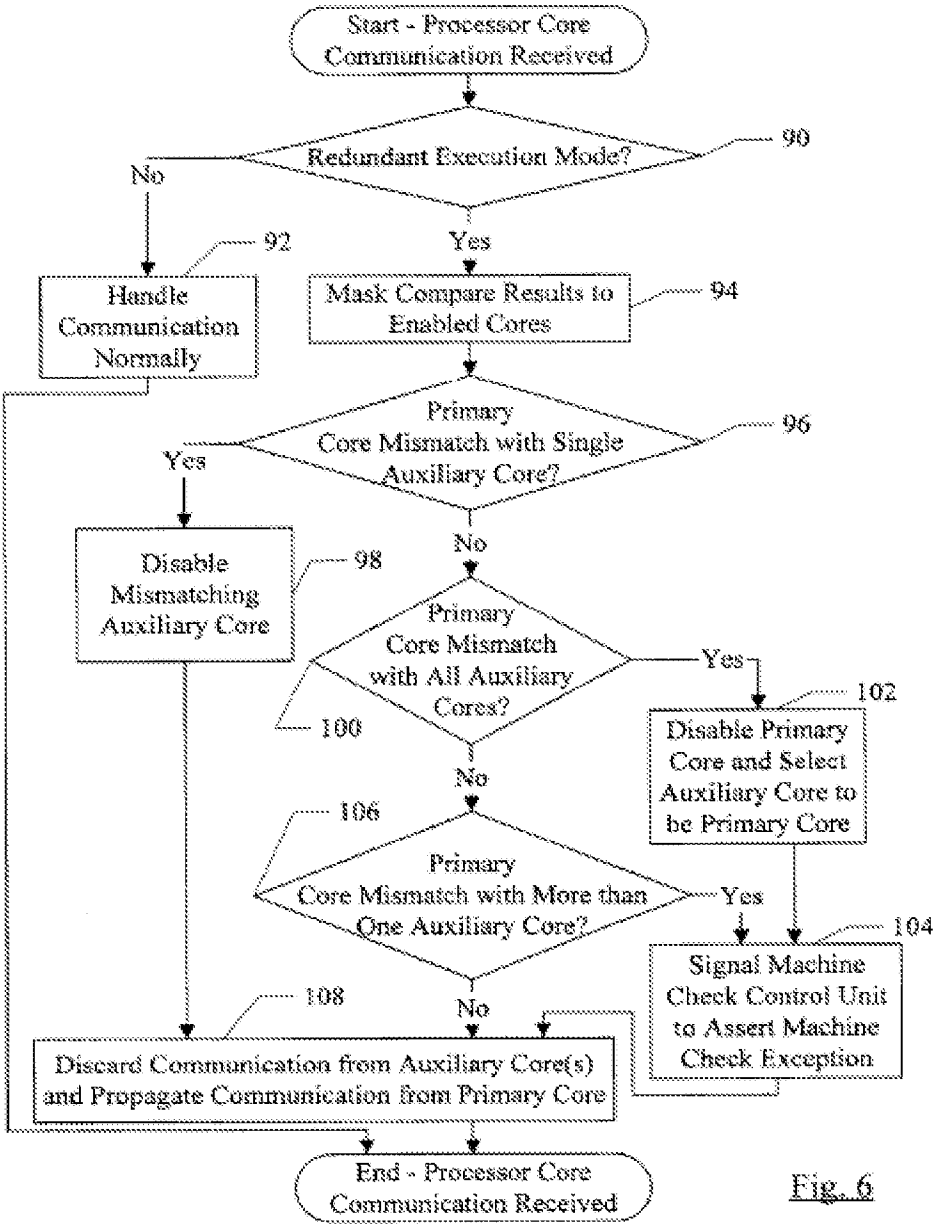


Fig. 6

Fig. 6 of Kalyanasundharam

5

detecting loss of lockstep (LOL) for a processor in a multi-processor system

Kalyanasundharam teaches in Figs. 1, 2 and 7 an error detecting mechanism (26A) that checks for errors within processors 18A-18B that execute in lockstep in a multi-processor system (Fig. 1): "*In embodiments in which the processor cores 18A-18B execute in lockstep, the redundancy check circuit 26A may check that corresponding communications are concurrently received from the processor cores 18A-18B (e.g. received on the same clock cycle). For example, if a valid communication is received from one of the processor cores 18A-18B and a valid communication is not concurrently received from the other processor core 18A- 18B, an execution error may be detected*" (Kalyanasundharam at col. 4:27-35). The multi-processor system is describe in col. 1:7-11 of Kalyanasundharam: "*This invention is related to the field of multiprocessors and, more particularly, to chip multiprocessors (CMPs) and implementing redundant execution for high reliability in CMPs.*"

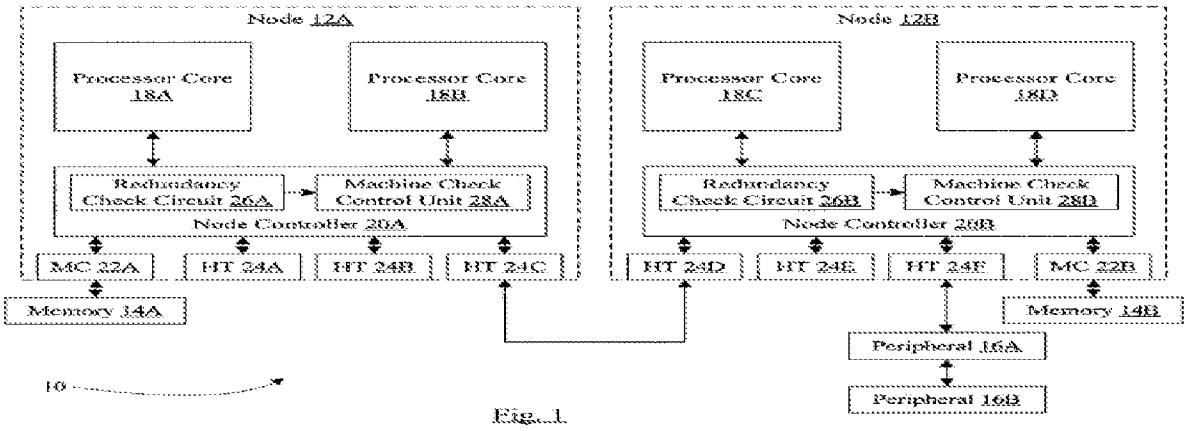


Fig. 1

Fig. 1 of Kalyanasundharam

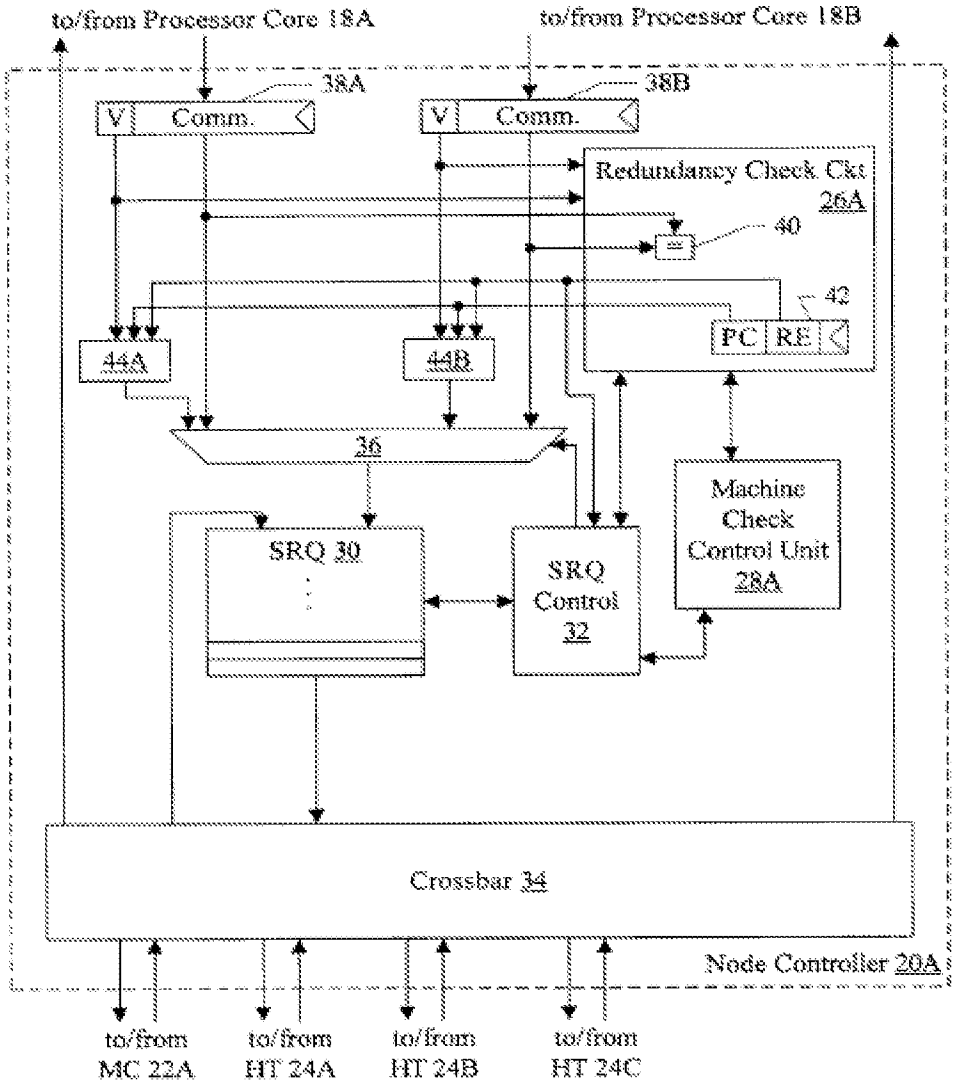


Fig. 2

Fig. 2 of Kalyanasundharam

determining that the processor for which said LOL is detected is assigned a role of boot processor,

Kalyanasundharam describes use of multiple pairs of processors (18A-18B) that operate in lockstep: "*If the masked comparison results indicate that the communication from the primary processor core mismatches with all auxiliary processor cores (that are enabled) (decision block 100, "yes" leg), the control circuit 84 may disable the primary processor core and select one of the auxiliary processor cores to be the primary processor core (block 102)*" (Kalyanasundharam at col. 13:16-22); and "*The processer cores may be configured as N/2 pairs of redundant processor cores. Each pair may redundantly execute different software code from other pairs. In some embodiments, the processor cores may be preassigned info pairs for redundant execution purposes" (*Kalyanasundharam at col. 13:60-64). While Kalyanasundharam determines, upon detecting a loss of lockstep, whether a processor is a primary processor core, Kalyanasundharam fails to specifically indicate that the processor is a "boot processor" as required in claim 1.

Nevertheless, Kennedy shows in Fig. 1 a multiprocessor system that identifies default boot processors and alternative boot processors should the default boot processor fail to boot: "*The CPU1 module 120 could be any microprocessor chip set running at any speed. In the present embodiment, at least one CPU is based upon an INTEL 80486 or compatible microprocessor. Accordingly, throughout the remainder of this description, references to the CPU1 120 assume an INTEL 80486-based CPU with supporting resources and on-board crystal oscillators for independent timing. Other CPUs in the system need not be 080486-based as explained in more detail herein. CPUs installed in the*

bus 104 may have independent asynchronous timing with respect to the bus 104" (Kennedy at 7:5-14). Kennedy teaches determining whether a processor module is assigned a role of system boot processor by disclosing "*In the multiprocessor system 100 which utilizes a system bus 102, a number of problems can arise if more than one processor type is installed on the 64-bit bus 104. For instance, provision needs to be made to determine which processor will boot the system, and if this "default" boot processor fails to boot the system, an alternative processor, if one is installed, should take over boot operations to enhance the availability of the system.*" (Kennedy at col. 7:5-13). Kennedy further describes the process of failing over from a failed boot processor to an alternative processor that takes over the functions of the failed boot processor "*The general function of the slot select logic 148 in- stalled in any of the slots 2-8 is to hold the corresponding CPU in reset until CPUs in previous slots have had sufficient (pre-determined) time to boot. In other words, for the CPU2 132 installed in slot 2, the slot select logic 148B holds the CPU 132 in a reset state until sufficient time has elapsed for the CPU1 120 to boot. If the CPU1 120 does not complete boot operations within the time allotted, then the CPU2 132 takes over default boot operations as an alternative boot processor instead of CPU1 120. If the CPU1 120 does boot properly, then the CPU2 132 checks in as an additional processor rather than taking over as a boot processor*" (Kennedy at col. 13:38-44).

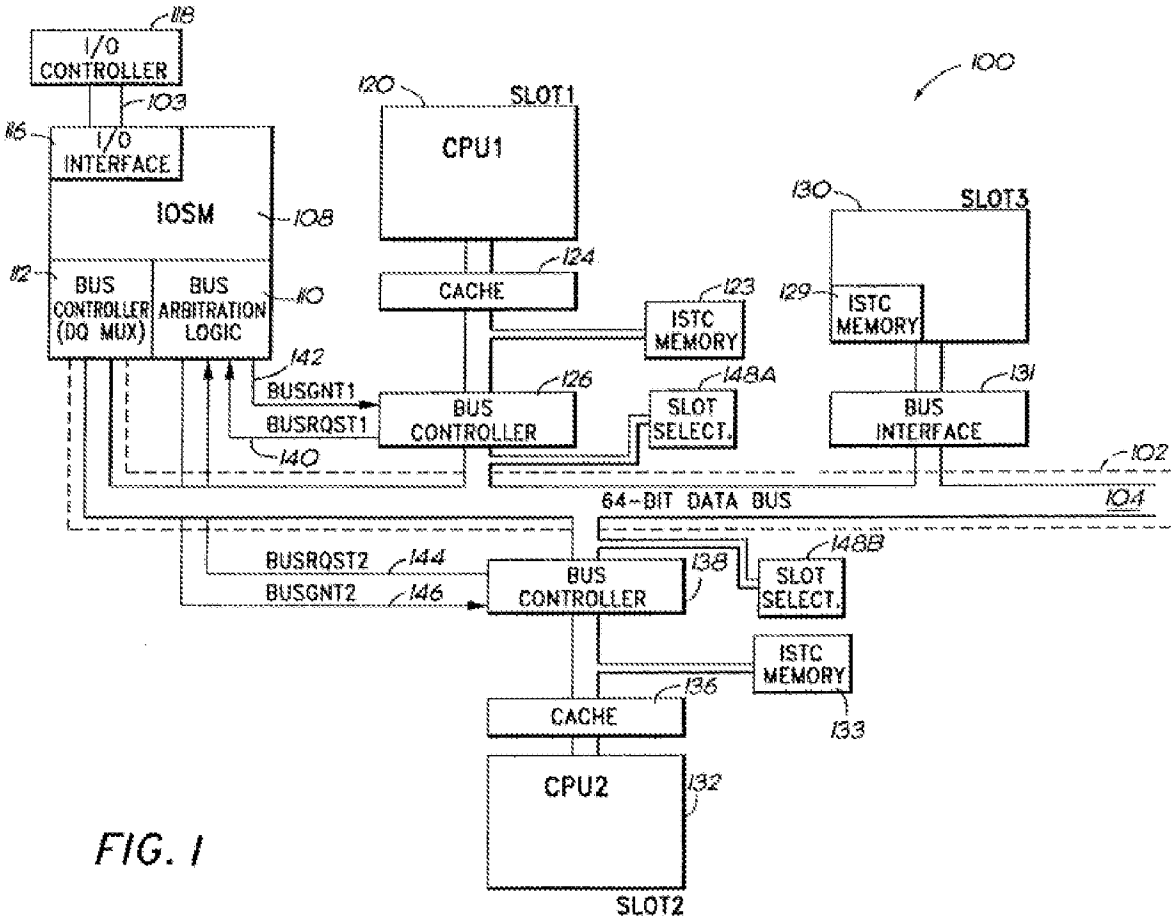


FIG. 1

Fig. 1 of Kennedy

It would have been obvious to combine Kalyanasundharam with Kennedy to expand Kalyanasundharam's ability to fail over to many different processors, i.e. boot processors as taught by Kennedy. Specifically, combining the boot processor of Kennedy to the primary and secondary processor cores of Kalyanasundharam for the purpose of increase the reliability and availability of the underlying system. A POSITA would also have been motivated to add the boot processor and spare processor described by Kennedy to extend its failover

functionality to a broader number of processor types. This would provide loss of lockstep failover not just between primary and secondary processor cores, but also between boot and spare processors. Because the Kalyanasundharam and Kennedy systems both describe a similar computing architecture in which processors fail over to backup processors upon the occurrence of a failure, the POSITA would reasonably expect to succeed when combining these teachings. Even when combining the processor failover elements of Kalyanasundharam with the boot processor specific failover elements of Kennedy, each element would perform the same function, and the resulting combination of elements would provide a predictable outcome of a highly reliable system that would continue to perform boot processes even after a failure in one processor has occurred. Wolfe Decl. (EX. 1003, ¶¶63-70, 75).

changing a system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected, to a different value

Kalyanasundharam discloses in Fig. 7 a control circuit 84 that can changing a system identifier "... *disable the primary processor core and select one of the auxiliary processor cores to be the primary processor core (block 102). As with the discussion above for disabling the auxiliary processor core, the control circuit 84 may simply disable the primary processor core from comparison by clearing the corresponding enable bit in the configuration register 42, or may disable the primary processor core from executing at all. The control circuit 84*

may also update the PC field of the configuration register 42 to reflect the newly selected primary processor core" (Kalyanasundharam at col. 13:19-29).

Kennedy also states, "*If the CPU1 120 in slot 1 does not boot (proper boot is indicated by a change in the level of the SLOT_ SELECT signal line 193 from high to low) before the timer 184 in slot 2 times-out, then the CPU2 132 exits the reset state and attempts to boot the system as an alternate boot CPU.*" (Kennedy at 15:9-14). "*If the CPU1 120 does not complete boot operations within the time allotted, then the CPU2 132 takes over default boot operations as an alternative boot processor instead of CPU1 120. If the CPU1 120 does boot properly, then the CPU2 132 checks in as an additional processor rather than taking over as a boot processor.*" (Kennedy at col. 13:38-44).

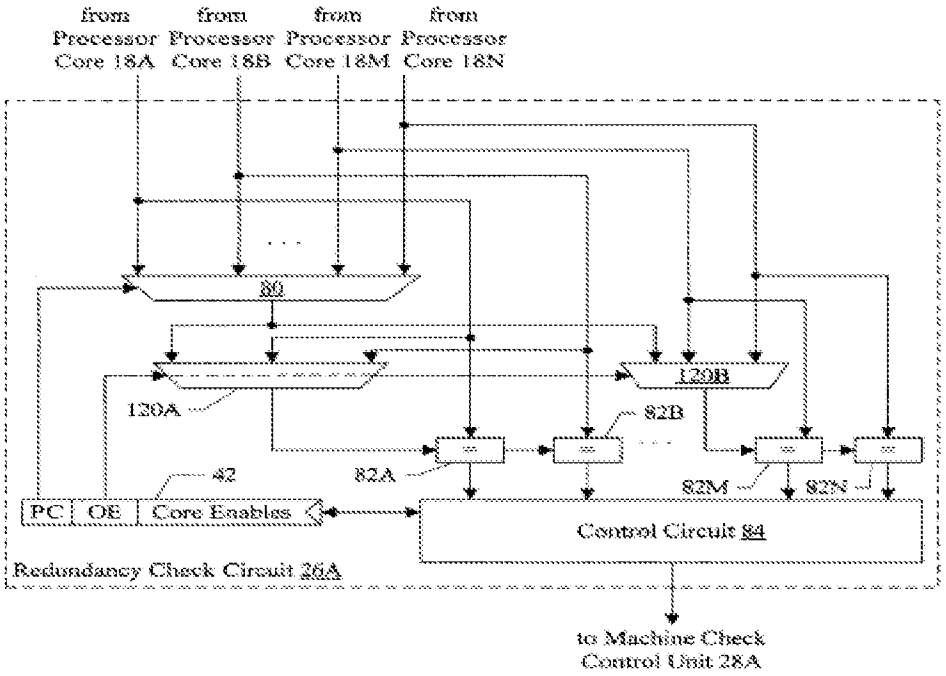


Fig. 7 of the Kalyanasundharam

which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor

Kennedy teach "*If the CPU1 120 does not complete boot operations within the time allotted, then the CPU2 132 takes over default boot operations as an alternative boot processor instead of CPU1 120. If the CPU1 120 does boot properly, then the CPU2 132 checks in as an additional processor rather than taking over as a boot processor.*" (Kennedy at col. 13:38-44). Thus, Kalyanasundharam as modified by Kennedy teach when one CPU fails to complete boot operations, another CPU will be used by the operating system as the boot processor. Thus, would cause the system's operating system to not recognize the processor for which the LOL is detected as the boot processor by indicating that slot select signals used by the operating system for communication would be changed in the event of a failure to boot,

changing a system identifier of the spare processor to that of the boot processor,

Kalyanasundharam teaches the control circuit 84 that can changing the system identifier: "... *disable the primary processor core and select one of the auxiliary processor cores to be the primary processor core (block 102). As with the discussion above for disabling the auxiliary processor core, the control circuit 84 may simply disable the primary processor core from comparison by clearing the corresponding enable bit in the configuration register 42, or may disable the primary processor core from executing at all. The control circuit 84 may also*

update the PC field of the configuration register 42 to reflect the newly selected primary processor core" (Kalyanasundharam at col. 13:19-29).

Kennedy also teaches "*If the CPU1 120 in slot 1 does not boot (proper boot is indicated by a change in the level of the SLOT SELECT signal line 193 from high to low) before the timer 184 in slot 2 times-out, then the CPU2 132 exits the reset state and attempts to boot the system as an alternate boot CPU.*" (Kennedy at col. 15:9-14)

thereby switching the role of boot processor to a spare processor without shutting down the system's operating system.

Kalyanasundharam discloses the control circuit 84 that can switching the role "... *disable the primary processor core and select one of the auxiliary processor cores to be the primary processor core (block 102). As with the discussion above for disabling the auxiliary processor core, the control circuit 84 may simply disable the primary processor core from comparison by clearing the corresponding enable bit in the configuration register 42, or may disable the primary processor core from executing at all. The control circuit 84 may also update the PC field of the configuration register 42 to reflect the newly selected primary processor core*" (Kalyanasundharam at col. 13:19-29). Since the configuration register is part of a redundancy checking circuit, changes to the configuration register would not require an operating system shutdown to implement. Kalyanasundharam as modified by Kennedy, would cause the system's operating system to switch the role of the boot processor to the spare

processor: "*If the CPU1 120 does not complete boot operations within the time allotted, then the CPU2 132 takes over default boot operations as an alternative boot processor instead of CPU1 120. If the CPU1 120 does boot properly, then the CPU2 132 checks in as an additional processor rather than taking over as a boot processor.*" (Kennedy at col. 13:38-44)

Regarding claim 5: **The method of claim 1 wherein the spare processor is a processor module that includes a master and slave processors that operate in lockstep.**

Kalyanasundharam discloses: "*The processor cores may be configured as N/2 pairs of redundant processor cores. Each pair may redundantly execute different software code from other pairs. In some embodiments, the processor cores may be preassigned into pairs for redundant execution purposes.*" (Kalyanasundharam at col. 13-60-64) and "*Generally, one of the processor cares 18A-18B may be designated as a primary processor core and the other processor core 18A-18B is referred to as the auxiliary processor core. The primary processor core may be the processor core whose execution results will be used as the result of executing the redundantly executed software code. The auxiliary processor care is used to verify correct execution by the primary processor core*". (Kalyanasundharam at col. 4:58-65). The primary (18A-18B) and auxiliary processor cores (the other processor core 18A-18B) can be seen as the master and slave processors.

Regarding claim 6: **The method of claim 1 wherein said determining that the processor for which said LOL is detected is assigned the role of boot processor comprises accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor.**

Kennedy teaches "*The CPU1 120 has a corresponding non-volatile memory 123 which contains configuration information and CPU specific initialize and self-test code (ISTC) for the CPU1 120...*" (Kennedy at col. 5:15-18). It would have been obvious to the POSITA to combine Kalyanasundharam with Kennedy by storing and accessing CPU-specific information from non-volatile memory as taught by Kennedy for the purpose of increasing the reliability of the system.

Regarding claim 7: **The method of claim 1 further comprising: determining the processor that is assigned the role of spare processor.**

Kennedy teaches "*For instance, provision needs to be made to determine which processor will boot the system, and if this "default" boot processor fails to boot the system, an alternative processor, if one is installed, should take over boot operations to enhance the availability of the system.*" (Kennedy at col. 7:7-13). It would have been obvious to the POSITA to combine Kalyanasundharam with Kennedy by identifying which processors are assigned certain roles, as taught by Kennedy for the purpose of increasing the reliability of the system.

Regarding claim 8: **The method of claim 7 wherein said determining the processor that is assigned the role of spare processor comprises: accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of spare processor.**

Kennedy teaches "*The CPU1 120 has a corresponding non-volatile memory 123 which contains configuration information and CPU specific initialize and self-test code (ISTC) for the CPU1 120...*" (Kennedy at 5:15-18) and "*For instance, provision needs to be made to determine which processor will boot the system, and if this "default" boot processor fails to boot the system, an alternative processor, if one is installed, should take over boot operations to enhance the availability of the system.*" (Kennedy at col. 7:5-13). It would have been obvious to the POSITA to combine Kalyanasundharam with Kennedy by storing and accessing CPU-specific information from non-volatile memory, and identifying which processors are assigned certain roles, as taught by Kennedy for the purpose of avoiding system errors, thus, increasing the reliability of the system.

Regarding claim 13: **A method comprising:**

Kalyanasundharam teaches in Fig. 6 a method of processor: "*FIG. 6 is a flowchart illustrating operation of one embodiment of the node controller including the redundancy check circuit shown in FIG. 5 in response to receiving a communication from a processor core.*" (Kalyanasundharam at col. 2:7-10):

system firmware determining that loss of lockstep (LOL) is detected for a lockstep pair of processors

Kalyanasundharam teaches in Figs. 1, 2 and 7 the error detecting mechanism (26A) that checks for errors within processors 18A-18B that execute in lockstep in the multi-processor system (Fig. 1): "*In embodiments in which the processor cores 18A-18B execute in lockstep, the redundancy check circuit 26A may check that corresponding communications are concurrently received from the processor cores 18A-18B (e.g. received on the same clock cycle). For example, if a valid communication is received from one of the processor cores 18A-18B and a valid communication is not concurrently received from the other processor core 18A- 18B, an execution error may be detected*" (Kalyanasundharam at col. 4:27-35). The multi-processor system is describe in col. 1:7-11 of Kalyanasundharam: "*This invention is related to the field of multiprocessors and, more particularly, to chip multiprocessors (CMPs) and implementing redundant execution for high reliability in CMPs.*"

that are assigned the role of boot processor in a system;

Similarly as explained in claim 1, while Kalyanasundharam determines, upon detecting the loss of lockstep, whether the processor is the primary processor core, Kalyanasundharam fails to specifically indicate that the processor is the "boot processor" as required in claim 13.

Nevertheless, Kennedy shows in Fig. 1 a multiprocessor system that identifies default boot processors and alternative boot processors should the default boot processor fail to boot: "*The CPU1 module 120 could be any microprocessor chip set running at any speed. In the present embodiment, at least one CPU is based upon an INTEL 80486 or compatible microprocessor. Accordingly, throughout the remainder of this description, references to the CPU1 120 assume an INTEL 80486-based CPU with supporting resources and on-board crystal oscillators for independent timing. Other CPUs in the system need not be 080486-based as explained in more detail herein. CPUs installed in the bus 104 may have independent asynchronous timing with respect to the bus 104*" (Kennedy at col. 7:5-14). Kennedy teaches determining whether the processor module is assigned the role of system boot processor by disclosing "*In the multiprocessor system 100 which utilizes a system bus 102, a number of problems can arise if more than one processor type is installed on the 64-bit bus 104. For instance, provision needs to be made to determine which processor will boot the system, and if this "default" boot processor fails to boot the system, an alternative processor, if one is installed, should take over boot operations to enhance the availability of the system.*" (Kennedy at col. 7:5-13). Kennedy further describes the process of failing over from the failed boot processor to the alternative processor that takes over the functions of the failed boot processor "*The general function of the slot select logic 148 in- stalled in any of the slots 2-8 is to hold the corresponding CPU in reset until CPUs in previous slots have had sufficient (pre-determined) time to boot. In other words, for the*

CPU2 132 installed in slot 2, the slot select logic 148B holds the CPU 132 in a reset state until sufficient time has elapsed for the CPU1 120 to boot. If the CPU1 120 does not complete boot operations within the time allotted, then the CPU2 132 takes over default boot operations as an alternative boot processor instead of CPU1 120. If the CPU1 120 does boot properly, then the CPU2 132 checks in as an additional processor rather than taking over as a boot processor" (Kennedy at col. 13:38-44).

It would have been obvious to combine Kalyanasundharam with Kennedy to expand Kalyanasundharam's ability to fail over to many different processors, i.e. boot processors as taught by Kennedy. Specifically, combining the boot processor of Kennedy to the primary and secondary processor cores of Kalyanasundharam for the purpose of increase the reliability and availability of the underlying system. The POSITA would also have been motivated to add the boot processor and spare processor described by Kennedy to extend its failover functionality to a broader number of processor types. This would provide loss of lockstep failover not just between primary and secondary processor cores, but also between boot and spare processors. Because the Kalyanasundharam and Kennedy systems both describe the similar computing architecture in which processors fail over to backup processors upon the occurrence of the failure, the POSITA would reasonably expect to succeed when combining these teachings. Even when combining the processor failover elements of Kalyanasundharam with the boot processor specific failover elements of Kennedy, each element would perform the same function, and the resulting

combination of elements would provide the predictable outcome of the highly reliable system that would continue to perform boot processes even after the failure in one processor has occurred. Wolfe Decl. (EX. 1003, ¶¶63-70, 75).

determining one of said lockstep pair of processors that is not the cause of said LOL;

Kalyanasundharam discloses "*The processer cores may be configured as N/2 pairs of redundant processor cores. Each pair may redundantly execute different software code from other pairs. In some embodiments, the processor cores may be preassigned info pairs for redundant execution purposes*" (Kalyanasundharam at col. 13:60-64). Kalyanasundharam further notes: "*If the masked comparison results indicate that the communication from the primary processor core mismatches with all auxiliary processor cores (that are enabled) (decision block 100, "yes" leg), the control circuit 84 may disable the primary processor core and select one of the auxiliary processor cores to be the primary processor core (block 102)*" (Kalyanasundharam at col. 13:16-22).

Kennedy also teaches "*In the multiprocessor system 100 which utilizes a system bus 102, a number of problems can arise if more than one processor type is installed on the 64-bit bus 104. For instance, provision needs to be made to determine which processor will boot the system, and if this "default" boot processor fails to boot the system, an alternative processor, if one is installed, should take over boot operations to enhance the availability of the system*" (Kennedy at col. 7:5-13); and "*The general function of the slot select logic 148 in-*

stalled in any of the slots 2-8 is to hold the corresponding CPU in reset until CPUs in previous slots have had sufficient (pre-determined) time to boot. In other words, for the CPU2 132 installed in slot 2, the slot select logic 148B holds the CPU 132 in a reset state until sufficient time has elapsed for the CPU1 120 to boot. If the CPU1 120 does not complete boot operations within the time allotted, then the CPU2 132 takes over default boot operations as an alternative boot processor instead of CPU1 120. If the CPU1 120 does boot properly, then the CPU2 132 checks in as an additional processor rather than taking over as a boot processor" (Kennedy at col. 13:38-44).

copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to a spare processor;

Kalyanasundharam discloses "*Additionally, the SRO control unit 32 may be configured to update state in the SRO 30 for a given communication in response to completion of various internal activities. For example, if a probe is in the SRO 30 and the corresponding responses from the processors 18A-18B are received for the probe, the SRO control unit 32 may update the state in the entry storing the probe to record the receipt of the response and to record the coherency State reported in the response*" (Kalyanasundharam at col. 7:34-43).

changing a system identifier, used by the system's operating system to recognize the boot processor, of said lockstep pair of processors that are assigned the role of boot processor, to a different value,

Kalyanasundharam discloses the control circuit 84 that can changing the system identifier "... *disable the primary processor core and select one of the auxiliary processor cores to be the primary processor core (block 102). As with the discussion above for disabling the auxiliary processor core, the control circuit 84 may simply disable the primary processor core from comparison by clearing the corresponding enable bit in the configuration register 42, or may disable the primary processor core from executing at all. The control circuit 84 may also update the PC field of the configuration register 42 to reflect the newly selected primary processor core*" (Kalyanasundharam at col. 13:19-29).

Kennedy also teaches "*If the CPU1 120 in slot 1 does not boot (proper boot is indicated by a change in the level of the SLOT_ SELECT signal line 193 from high to low) before the timer 184 in slot 2 times-out, then the CPU2 132 exits the reset state and attempts to boot the system as an alternate boot CPU.*" (Kennedy at 15:9-14). "*If the CPU1 120 does not complete boot operations within the time allotted, then the CPU2 132 takes over default boot operations as an alternative boot processor instead of CPU1 120. If the CPU1 120 does boot properly, then the CPU2 132 checks in as an additional processor rather than taking over as a boot processor.*" (Kennedy at col. 13:38-44).

which causes the system's operating system to not recognize the lock step pair of processors for which said LOL is detected as the processor; and

Kennedy teaches "*If the CPU1 120 does not complete boot operations within the time allotted, then the CPU2 132 takes over default boot operations as an alternative boot processor instead of CPU1 120. If the CPU1 120 does boot properly, then the CPU2 132 checks in as an additional processor rather than taking over as a boot processor.*" (Kennedy at col. 13:38-44).

Therefore, Kalyanasundharam as modified by Kennedy teach when one CPU fails to complete boot operations, another CPU will be used by the operating system as the processor. Thus, would cause the system's operating system to not recognize the processor for which the LOL is detected as the processor by indicating that slot select signals used by the operating system for communication would be changed in the event of the failure to boot,

changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to said spare processor.

Kalyanasundharam teaches the control circuit 84 that can changing the system identifier: "... *disable the primary processor core and select one of the auxiliary processor cores to be the primary processor core (block 102). As with the discussion above for disabling the auxiliary processor core, the control circuit 84 may simply disable the primary processor core from comparison by clearing the corresponding enable bit in the configuration register 42, or may disable the primary processor core from executing at all. The control circuit 84 may also*

update the PC field of the configuration register 42 to reflect the newly selected primary processor core" (Kalyanasundharam at col. 13:19-29).

Kennedy also teaches "*If the CPU1 120 in slot 1 does not boot (proper boot is indicated by a change in the level of the SLOT SELECT signal line 193 from high to low) before the timer 184 in slot 2 times-out, then the CPU2 132 exits the reset state and attempts to boot the system as an alternate boot CPU.*" (Kennedy at col. 15:9-14)

Kalyanasundharam further discloses the control circuit 84 that can switching the role "*... disable the primary processor core and select one of the auxiliary processor cores to be the primary processor core (block 102). As with the discussion above for disabling the auxiliary processor core, the control circuit 84 may simply disable the primary processor core from comparison by clearing the corresponding enable bit in the configuration register 42, or may disable the primary processor core from executing at all. The control circuit 84 may also update the PC field of the configuration register 42 to reflect the newly selected primary processor core*" (Kalyanasundharam at col. 13:19-29).

Kalyanasundharam as modified by Kennedy, would cause the system's operating system to switch the role of the boot processor to the spare processor: "*If the CPU1 120 does not complete boot operations within the time allotted, then the CPU2 132 takes over default boot operations as an alternative boot processor instead of CPU1 120. If the CPU1 120 does boot properly, then the CPU2 132 checks in as an additional processor rather than taking over as a boot processor.*" (Kennedy at col. 13:38-44).

Regarding claim 17: **The method of claim 13 wherein said switching causes the system's operating system to recognize said spare processor as said boot processor without shutting down said operating system.**

Kalyanasundharam as modified by Kennedy, would cause the system's operating system to switch the role of the boot processor to the spare processor: "*If the CPU1 120 does not complete boot operations within the time allotted, then the CPU2 132 takes over default boot operations as an alternative boot processor instead of CPU1 120. If the CPU1 120 does boot properly, then the CPU2 132 checks in as an additional processor rather than taking over as a boot processor.*" (Kennedy at col. 13:38-44). Since the configuration register is part of the redundancy checking circuit 26A of Kalyanasundharam, changes to the configuration register would not require the operating system shutdown to implement.

ALLOWABLE SUBJECT MATTER

Regarding claim 21: The prior art of record fails to specifically suggest or disclose the corresponding structure for FL #4 – FL#5 as discussed above in the Claim Interpretation section: "*error detection logic operable to detect loss of lockstep (LOL) for at least one of said processor modules*" in combination with "*system firmware operable, responsive to the detection of LOL for a first of said processor modules, to determine whether said first processor module is assigned a role of system boot processor; wherein if determined that said first processor*

module is assigned the role of system boot processor, said system firmware is further operable to cause said operating system to recognize another processor module as system boot processor without shutting down said operating system."

Claims 22 and 25 contain allowable subject matter at least by virtue of their dependency from independent claim 21.

Regarding claim 26: The prior art of record fails to specifically suggest or disclose the corresponding structure for FL #1 – FL#3 as discussed above in the Claim Interpretation section: "*system firmware means for detecting loss of lockstep (LOL) for a processor in a multi-processor system*"; "*means for determining whether the processor for which said LOL is detected is assigned the role of boot processor*" in combination with "*system firmware means for switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which said LOL is detected is assigned the role of boot processor*".

Regarding dependent claims 2-4, 9-12, 14-16, 18-20, 23, 24 and 27: Insofar as these claims are not subject to reexamination based on Kalyanasundharam and Kennedy, no opinion is offered regarding whether these claims are patentable.

DUTY TO DISCLOSE

The Patent Owner is reminded of the continuing responsibility under 37 C.F.R. § 1.565(a) to apprise the Office of any litigation activity, or other prior or concurrent proceeding, throughout the course of this reexamination proceeding involving the 302 Patent. The Requester is also reminded of the ability to similarly apprise the Office of any such activity or proceeding throughout the course of this reexamination proceeding. See MPEP §§ 2207, 2282 and 2286.

SERVICE OF PAPERS

After the filing of a request for reexamination by the Requester, any document filed by either the Patent Owner or the Requester must be served on the other party (or parties where two or more third party requester proceedings are merged) in the reexamination proceeding in the manner provided in 37 C.F.R. § 1.248 (See also 37 C.F.R. § 1.550).

EXTENSIONS OF TIME

Extensions of time under 37 C.F.R. §1.136(a) will not be permitted in these proceedings because the provisions of 37 C.F.R. §1.136 apply only to "an applicant" and not to parties in a reexamination proceeding. Additionally, 35 U.S.C. §305 requires that reexamination proceedings "will be conducted with special dispatch" (37 C.F.R. §1.550(a)). Extension of time in reexamination proceedings are provided for in 37 C.F.R. §1.550(c). After the filing of a request for reexamination by the Requester, any document filed by either the Patent

Owner or the Requester must be served on the other party (or parties where two or more third-party-requester proceedings are merged) in the reexamination proceeding in the manner provided in 37 C.F.R. §1.248. See 37 C.F.R. §1.550(f).

CONCLUSION

Information regarding the status of reexamination proceedings may be obtained from the USPTO's Patent Center. To file and manage patent submissions in Patent Center, visit: https://patentcenter.uspto.gov.

Visit https://www.uspto.gov/patents/apply/patent-center for more information about Patent Center and https://www.uspto.gov/patents/docx for information about filing in DOCX format. For additional questions, contact the Electronic Business Center (EBC) at (866) 217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative, call (800) 786-9199 (IN USA OR CANADA) or (571) 272-1000.

Any inquiry concerning this communication or earlier communications from the Reexamination Legal Advisor or Examiner, or as to the status of this proceeding, should be directed to the Central Reexamination Unit at telephone number (571) 272-7705.

Signed:

/MY TRANG TON/
Primary Examiner, Art Unit 3992

Conferees:

/KENNETH WHITTINGTON/
Primary Examiner, Art Unit 3992

/ANDREW J. FISCHER/
Supervisory Patent Reexamination Specialist, Art Unit 3992

Reexamination	Application/Control No.	Applicant(s)/Patent Under Reexamination
	90/019,243	7624302
	Certificate Date	Certificate Number

Requester Correspondence Address: ☐ Patent Owner ☑ Third Party
GREENBERG TRAURIG (NJ) 500 CAMPUS DRIVE, SUITE 400 P.O. BOX 677 FLORHAM PARK NJ 07932

LITIGATION REVIEW ☑	mt (examiner initials)	18 September 2023 (date)
Case Name		Director Initials
1:23cv640 | Foras Technologies Ltd. V. Nissan Motor Company, Ltd. Et Al (Open)		
2:23cv150 Foras Technologies Ltd. V. Toyota Motor North America. Inc. Et Al (Open)		

COPENDING OFFICE PROCEEDINGS	
TYPE OF PROCEEDING	NUMBER
None	

PTO/AIA/96 (11-23)
Approved for use through 05/31/2024. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

STATEMENT UNDER 37 CFR 3.73(c)

First Named Inventor/Patent Owner: Foras Technologies Ltd.

Application No./Patent No.: 7,624,302 Filed/Issue Date: November 24, 2009

Titled: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

Foras Technologies Ltd., a corporation

(Name of Assignee) (Type of Assignee, e.g., corporation, partnership, university, government agency, etc.)

states that, for the patent application/patent identified above, it is (choose **one** of options 1, 2, 3 or 4 below):

1. ☑ The assignee of the entire right, title, and interest.

2. ☐ An assignee of less than the entire right, title, and interest (check applicable box):

   ☐ The extent (by percentage) of its ownership interest is _____%. Additional Statement(s) by the owners holding the balance of the interest must be submitted to account for 100% of the ownership interest.

   ☐ There are unspecified percentages of ownership. The other parties, including inventors, who together own the entire right, title and interest are:

   Additional Statement(s) by the owner(s) holding the balance of the interest must be submitted to account for the entire right, title, and interest.

3. ☐ The assignee of an undivided interest in the entirety (a complete assignment from one of the joint inventors was made). The other parties, including inventors, who together own the entire right, title, and interest are:

   Additional Statement(s) by the owner(s) holding the balance of the interest must be submitted to account for the entire right, title, and interest.

4. ☐ The recipient, via a court proceeding or the like (*e.g.*, bankruptcy, probate), of an undivided interest in the entirety (a complete transfer of ownership interest was made). The certified document(s) showing the transfer is attached.

The interest identified in option 1, 2 or 3 above (not option 4) is evidenced by either (choose **one** of options A or B below):

A. ☐ An assignment from the inventor(s) of the patent application/patent identified above. The assignment was recorded in the United States Patent and Trademark Office at Reel _____, Frame _____, or for which a copy thereof is attached*.

B. ☑ A chain of title from the inventor(s), of the patent application/patent identified above, to the current assignee as follows:

1. From: Scott L. Michaelis, Anurupa Rajkumari & William B. McHardy To: Hewlett-Packard Development Company, L.P.

   The document was recorded in the United States Patent and Trademark Office at
   Reel 015932, Frame 0117, or for which a copy thereof is attached*.

2. From: Hewlett-Packard Development Company, L.P. To: Hewlett Packard Enterprise Development LP

   The document was recorded in the United States Patent and Trademark Office at
   Reel 037079, Frame 0001, or for which a copy thereof is attached*.

A Federal agency may not conduct or sponsor, and a person is not required to respond to, nor shall a person be subject to a penalty for failure to comply with an information collection subject to the requirements of the Paperwork Reduction Act of 1995, unless the information collection has a currently valid OMB Control Number. The OMB Control Number for this information collection is 0651-0031. Public burden for this form is estimated to average 12 minutes per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the information collection. Send comments regarding this burden estimate or any other aspect of this information collection, including suggestions for reducing this burden to the Chief Administrative Officer, United States Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450 or email InformationCollection@uspto.gov. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

STATEMENT UNDER 37 CFR 3.73(c)

3. From: Hewlett Packard Enterprise Development LP, Hewlett Packard Enterprise Company To: Sonraí Memory, Ltd.

The document was recorded in the United States Patent and Trademark Office at
Reel 052567, Frame 0734, or for which a copy thereof is attached*.

4. From: Sonraí Memory, Ltd. To: Foras Technologies Ltd.

The document was recorded in the United States Patent and Trademark Office at
Reel 058992, Frame 0571, or for which a copy thereof is attached*.

5. From: _____ To: _____

The document was recorded in the United States Patent and Trademark Office at
Reel _____, Frame _____, or for which a copy thereof is attached*.

6. From: _____ To: _____

The document was recorded in the United States Patent and Trademark Office at
Reel _____, Frame _____, or for which a copy thereof is attached*.

☐ Additional documents in the chain of title are listed on a supplemental sheet(s).

☑ *As required by 37 CFR 3.73(c)(1)(i), the documentary evidence of the chain of title from the original owner to the assignee was, or concurrently is being, submitted for recordation pursuant to 37 CFR 3.11.

[NOTE: A separate copy (i.e., a true copy of the original assignment document(s)) must be submitted to Assignment Division in accordance with 37 CFR Part 3, to record the assignment in the records of the USPTO. See MPEP 302.08]

The undersigned (whose title is supplied below) is authorized to act on behalf of the assignee.

/Antonio Papageorgiou/	April 23, 2024
Signature	Date
Antonio Papageorgiou	53431
Printed or Typed Name	Title or Registration Number

[Page 2 of 2]

Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. The United States Patent and Trademark Office (USPTO) collects the information in this record under authority of 35 U.S.C. 2. The USPTO's system of records is used to manage all applicant and owner information including name, citizenship, residence, post office address, and other information with respect to inventors and their legal representatives pertaining to the applicant's/owner's activities in connection with the invention for which a patent is sought or has been granted. The applicable Privacy Act System of Records Notice for the information collected in this form is COMMERCE/PAT-TM-7 Patent Application Files, available in the Federal Register at 78 FR 19243 (March 29, 2013). https://www.govinfo.gov/content/pkg/FR-2013-03-29/pdf/2013-07341.pdf

Routine uses of the information in this record may include disclosure to:

1) law enforcement, in the event that the system of records indicates a violation or potential violation of law;

2) a federal, state, local, or international agency, in response to its request;

3) a contractor of the USPTO having need for the information in order to perform a contract;

4) the Department of Justice for determination of whether the Freedom of Information Act (FOIA) requires disclosure of the record;

5) a Member of Congress submitting a request involving an individual to whom the record pertains, when the individual has requested the Member's assistance with respect to the subject matter of the record;

6) a court, magistrate, or administrative tribunal, in the course of presenting evidence, including disclosures to opposing counsel in the course of settlement negotiations;

7) the Administrator, General Services Administration (GSA), or their designee, during an inspection of records conducted by GSA under authority of 44 U.S.C. 2904 and 2906, in accordance with the GSA regulations and any other relevant (i.e., GSA or Commerce) directive, where such disclosure shall not be used to make determinations about individuals;

8) another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c));

9) the Office of Personnel Management (OPM) for personnel research purposes; and

10) the Office of Management and Budget (OMB) for legislative coordination and clearance.

If you do not furnish the information requested on this form, the USPTO may not be able to process and/or examine your submission, which may result in termination of proceedings, abandonment of the application, and/or expiration of the patent.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re patent of	:	Michaelis, Scott L, et al.		
Application No.	:	90/019,243	Confirmation No. :	1079
Patent No.	:	7,624,302		
Filing Date	:	October 25, 2004	Issue Date:	November 24, 2009
Title	:	SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR		

Request for an Extension of Time Pursuant to 37 C.F.R. 1.550(c)

A non-final office action was issued in the above referenced reexamination of the on February 23, 2024, with a two-month response period making a reply from the patent owner due on April 23, 2024.

The patent owner is requesting a one month extension to file a reply to the non-final office action pursuant to 37 C.F.R. 1.550(c). The patent owner submits that this extension is reasonable.

The fee pursuant to 1.17(g) accompanies this petition.

As noted in the declaration in support of this petition by Mr. Brett Cooper, the patent owner has sufficient cause for the Office to grant its request. Specifically, the third party requester, Unified Patents LLC, served the request for reexamination using the customer number associated with the previous assignor, Hewlett Packard Enterprises. The previous assignor was under no obligation to forward notices to the patent owner, and although it would often forward communications to the patent owner, this time it did not. As a result, the patent owner nor its representatives received notice of the non-final office action until April 22, 2024, only one day

prior to the due date for a response. Specifically, Mr. Cooper became aware of the reexamination in connection with a pre-trial conference that was held on April 22, 2024, one day prior to the due date. (See Cooper Decl. at ¶ 7). Because of the late notice of the Office Action, FORAS does not have the ability to meaningfully reply to the Office Action.

Accordingly, grant of this request is respectfully requested.

Date: April 23, 2024

Respectfully submitted,

/Antonio Papageorgiou/
Antonio Papageorgiou

Reg. No. 53,431

UNITED STATES
PATENT AND TRADEMARK OFFICE

P.O. Box 1450
Alexandria, VA 22313 - 1450
www.uspto.gov

ELECTRONIC ACKNOWLEDGEMENT RECEIPT

APPLICATION #	RECEIPT DATE / TIME	ATTORNEY DOCKET #
90/019,243	**04/23/2024 12:21:23 PM Z ET**	**185945.012600**

Title of Invention

SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

Application Information

APPLICATION TYPE	Utility / ex parte reexam	PATENT #	7624302
CONFIRMATION #	1079	FILED BY	Laura Alto
PATENT CENTER #	65206026	FILING DATE	09/01/2023
CUSTOMER #	56436	FIRST NAMED INVENTOR	- 7624302
CORRESPONDENCE ADDRESS	-	AUTHORIZED BY	Antonio Papageorgiou

Documents

TOTAL DOCUMENTS: 4

DOCUMENT	PAGES	DESCRIPTION	SIZE (KB)
FTL-REX243_POA_04-23-2024.pdf	2	Power of Attorney	782 KB
FTL-REX243_Statement of Assignee_04-23-2024.pdf	3	Assignee showing of ownership per 37 CFR 3.73	232 KB
FTL-REX243_Petition for Extension Reexamination_04-23-2024.pdf	2	Reexam Request for Extension of Time	105 KB
FTL-REX243_Declaration_in_support_extension_04-22-	14	Oath or Declaration filed	7570 KB

2024.pdf

Digest

DOCUMENT	MESSAGE DIGEST(SHA-512)
FTL-REX243_POA_04-23-2024.pdf	440A720A0401D2547B47371C199AB513F1948DB522B7678DC9C5EE44B7F39DA8747DCA02291606BF006138A7248020391360B7E03D0028F2276B5FF0413B8FB1
FTL-REX243_Statement of Assignee_04-23-2024.pdf	813F35C1819D2F138543AA23F312FFDCE28B5D878061592D3625CB7F55198DD7F6B16EED4DEB4918201C99F432C72146C016BA8B3926CA93F680945FD3113677
FTL-REX243_Petition for Extension Reexamination_04-23-2024.pdf	9EDFBE79DD9E053A44546CBEEDE815B5BEDC6283C3141F4C8DB75C5EDDBBB0CF0482675EE4CAA310C32AA136290B94417229BDED82B5C707E53A40F2452970F5
FTL-REX243_Declaration_in_support_extension_04-22-2024.pdf	5DF2AEE02CA8E4A947C335E1AE223C47AE1C2C5237DFB66E70E8CAE32D73D37D59D857A700FD03048758BE6445E95BD2F20C22EC90AFE362BD270839418 6FE86

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PTO/AIA/81B (07-13)
Approved for use through 09/30/2025. OMB 0651-0035
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995 no persons are required to respond to a collection of information unless it displays a valid OMB control number

REEXAMINATION OR SUPPLEMENTAL EXAMINATION – PATENT OWNER POWER OF ATTORNEY OR REVOCATION OF POWER OF ATTORNEY WITH A NEW POWER OF ATTORNEY AND CHANGE OF CORRESPONDENCE ADDRESS FOR REEXAMINATION OR SUPPLEMENTAL EXAMINATION AND PATENT		
	Control Number(s)	90/019,243
	Filing Date(s)	September 1, 2023
	First Named Inventor	Scott L. Michaelis
	Title	System And Method For Switching The Role Of Boot Processor To A Spa
	Patent Number	7,624,302
	Examiner Name	My Trang Ton
	Attorney Docket No(s)	FTL-REX243

I. Power of Attorney. This form may be used to change the Power of Attorney in a reexamination or supplemental examination proceeding (or multiple proceedings where merged). This form may also be used to change the Power of Attorney in the patent file; in such a case, a copy of this form will be placed in both the patent file and the reexamination or supplemental examination proceeding.

A. Revocation of Previous Power of Attorney. I hereby revoke all previous patent owner powers of attorney, if any, given:

■ in the above-identified reexamination or supplemental examination proceeding control number(s) (more than one may be changed only if the proceedings are merged).

■ in the file of the above-identified patent.

(check BOTH boxes if change in BOTH the patent file and the reexamination or supplemental examination proceeding is requested).

B. Designation of Power of Attorney.

☐ A Power of Attorney is submitted herewith.

OR

■ I hereby appoint Practitioner(s) associated with the Customer Number identified in the box at right as my/our attorney(s) or agent(s) to prosecute the proceeding(s)/patent identified above and selected in section I(A), and to transact all business in the United States Patent and Trademark Office connected therewith:

89220

OR

☐ I hereby appoint Practitioner(s) named below as my/our attorney(s) or agent(s) to prosecute the proceeding(s) identified above, and to transact all business in the United States Patent and Trademark Office connected therewith:

Practitioner(s) Name	Registration Number

Authorization for the Power of Attorney is provided by the signature on page 2 of this form.

A Federal agency may not conduct or sponsor, and a person is not required to respond to, nor shall a person be subject to a penalty for failure to comply with an information collection subject to the requirements of the Paperwork Reduction Act of 1995, unless the information collection has a currently valid OMB Control Number. The OMB Control Number for this information collection is 0651-0035. Public burden for this form is estimated to average 15 minutes hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the information collection. Send comments regarding this burden estimate or any other aspect of this information collection, including suggestions for reducing this burden to the Chief Administrative Officer, United States Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450 or email InformationCollection@uspto.gov. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **If filing this completed form by mail, send to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

II. Change of Correspondence Address

Please recognize or change the correspondence address for the above-identified reexamination or supplemental examination proceeding control number(s) (more than one may be changed only if they are merged proceedings) and for the file of the above-identified patent to be:

■ The address associated with the above-identified Customer Number.
OR
☐ The address associated with the Customer Number identified in the box at right:
OR

☐ Firm or Individual Name					
Address					
City		State		Zip	
Country					
Telephone		Email			

NOTE: THE CORRESPONDENCE ADDRESS FOR THE REEXAMINATION OR SUPPLEMENTAL EXAMINATION PROCEEDING CONTROL NUMBER(S) MUST BE THE SAME AS THAT FOR THE PATENT. SEE 37 CFR 1.33.

III. Authorization for Power of Attorney and (if selected) Change of Correspondence Address

I am the:
☐ Inventor, having ownership of the patent being reexamined.
OR
■ Patent owner.
Statement under 37 CFR 3.73(c) (Form PTO/AIA/96) submitted herewith or filed on _____.

Signature of Inventor or Patent Owner	Liam O'Hora	Date	April 22, 2024
Name	CIARAN O'GARA	Telephone	
Title and Company	Director , Foras Technologies Ltd.		

NOTE: Signatures of all the inventors or patent owners of the entire interest or their representative(s) are required. If more than one signature is required, submit multiple forms, check the box below, and identify the total number of forms submitted in the blank below.

■ A total of 1 forms are submitted. *If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.*

UNITED STATES
PATENT AND TRADEMARK OFFICE

P.O. Box 1450
Alexandria, VA 22313 - 1450
www.uspto.gov

ELECTRONIC PAYMENT RECEIPT

APPLICATION #	RECEIPT DATE / TIME	ATTORNEY DOCKET #
90/019,243	04/23/2024 12:21:23 PM Z ET	185945.012600

Title of Invention

SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

Application Information

APPLICATION TYPE	Utility / ex parte reexam	PATENT #	7624302
CONFIRMATION #	1079	FILED BY	Laura Alto
PATENT CENTER #	65206026	AUTHORIZED BY	Antonio Papageorgiou
CUSTOMER #	56436	FILING DATE	09/01/2023
CORRESPONDENCE ADDRESS	-	FIRST NAMED INVENTOR	- 7624302

Payment Information

PAYMENT METHOD	PAYMENT TRANSACTION ID	PAYMENT AUTHORIZED BY
CARD / 1047	E20244MC21466341	Laura Alto

FEE CODE	DESCRIPTION	ITEM PRICE($)	QUANTITY	ITEM TOTAL($)
1463	PETITIONS REQUIRING THE PETITION FEE SET FORTH IN 37 CFR 1.17(G) (GROUP II)	220.00	1	220.00
			TOTAL AMOUNT:	$220.00

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re patent of	:	Michaelis, Scott L, et al.	
Application No.	:	90/019,243	Confirmation No. : 1079
Patent No.	:	7,624,302	
Filing Date	:	10/25/2004	Issue Date: 11/24/2009
Title	:	SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR	

DECLARATION OF BRETT COOPER IN SUPPORT OF PETITION TO FOR EXTENSION OF TIME TO RESPOND TO A REEXAMINATION OFFICE ACTION PURSUANT TO 37 CFR 1.550(c)

I, the undersigned, hereby declare and state that:

1) U.S. Patent No. 7,624,302 (the "'302 Patent") is owned by FORAS TECHNOLOGIES LTD. ("FORAS") (*see* Frame/Reel No. 058992/0571) (Ex. A, Assignment 4), which is located at the following address:

THE HYDE BUILDING, SUITE 23
THE PARK, CARRICKMINES
DUBLIN 18, IRELAND

2) I am an attorney for FORAS in a litigation involving the '302 Patent, namely *Foras Technologies Ltd. v. Nissan Motor Company, Ltd, et al.*, 1:23-cv-640 (EDTX) (the "Action"). *See* Ex. B.

3) I am authorized to act on behalf of the assignee, FORAS, including to submit this statement on its behalf in support of the extension of time herein to respond to the Office Action dated February 23, 2024 (the "Office Action").

4) The request for reexamination of the '302 Patent was filed on September 1, 2023. The certificate of service by the third party requester (Unified Patents, LLC)

indicates that the request and the accompanying appendices and exhibits were served by mail as follows:

FORAS TECHNOLOGIES LTD.
c/o 56436 - Hewlett Packard Enterprise
3404 E. Harmony Road
Mail Stop 79
Fort Collins, CO 80528

5) This customer number is associated with the previous assignor, Hewlett Packard Enterprise ("HP"). (*See* Ex. A, Assignment 3). On information and belief, the third party requester did not attempt to serve FORAS directly, including at the assignee address noted above.

6) HP is under no obligation to forward documents concerning the assigned patents, and on information and belief, HP did not alert FORAS and FORAS did not therefore receive any documents concerning the ' 302 Patent that were mailed to the HP address, including the request for reexamination and the Office Action in this reexamination.

7) FORAS first learned about this reexamination and the Office Action on April 22, 2024, while getting prepared for the initial pretrial conference with Judge Howell in the Western District of Texas. *See* Ex. B (D.I. 40). Specifically, a motion to stay proceedings in the Action (filed on April 10, 2024) listed the existence of this reexamination – the opposition to this motion to stay is not due until Friday, April 26, 2024. Once the existence of this reexamination was known, I checked the status on the USPTO Patent Center. This is when FORAS first learned that an Office Action and that a response thereto is due on April 23, 2024.

8) Because of the late notice of the Office Action, FORAS does not have the ability to meaningfully reply to the Office Action.

9) Accordingly, FORAS respectfully requests a one-month extension of time, which would extend the deadline for a response to May 23, 2024, for it to reply to the Office Action.

10) I hereby declare that all statements made herein of my own knowledge are true, all statements made herein on information and belief are believed to be true, and all

statements made herein are made with the knowledge that whoever knowingly and willfully makes any false, fictitious, or fraudulent statements or representations shall be subject to the penalties set forth under 18 U.S.C. 1001.

Date: April 22, 2023

Respectfully submitted,

/Brett Cooper/
Brett Cooper, Esq.

For Foras Technologies Ltd.

EXHIBIT A

UNITED STATES
PATENT AND TRADEMARK OFFICE

Assignment abstract of title for Application 10972588

Invention title/Inventor	Patent	Publication	Application	PCT	International registration
SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR Scott L. Michaelis, Anurupa Rajkumari, William B. McHardy	7624302 Nov 24, 2009	20060090064 Apr 27, 2006	10972588 Oct 25, 2004		

Assignments (4 total)

Assignment 4

Reel/frame	Execution date	Date recorded	Properties	Pages
058992/0571	Jan 19, 2022	Feb 11, 2022	5	5

Conveyance
ASSIGNMENT OF ASSIGNORS INTEREST (SEE DOCUMENT FOR DETAILS).

Assignors
SONRAÍ MEMORY, LTD.

Correspondent
VORYS, SATER, SEYMOUR AND PEASE LLP
1909 K STREET NW
SUITE 900
WASHINGTON, DC 20006-1152

Assignee
FORAS TECHNOLOGIES LTD.
THE HYDE BUILDING, SUITE 23,
THE PARK, CARRICKMINES,
DUBLIN 18
IRELAND

Assignment 3

Reel/frame	Execution date	Date recorded	Properties	Pages
052567/0734	Apr 23, 2020	May 5, 2020	17	6

Conveyance
ASSIGNMENT OF ASSIGNORS INTEREST (SEE DOCUMENT FOR DETAILS).

Assignors
HEWLETT PACKARD ENTERPRISE DEVELOPMENT LP
HEWLETT PACKARD ENTERPRISE COMPANY

Correspondent
ADAM MULLOOLY
SUITE 23, THE HYDE BUILDING, THE PARK, CARRICKMINE
DUBLIN, DUBLIN 18 IRELAND

Assignee
SONRAÍ MEMORY, LTD.
SUITE 23, THE HYDE BUILDING, THE PARK, CARRICKMINE
DUBLIN DUBLIN 18
IRELAND

Assignment 2

Reel/frame	Execution date	Date recorded	Properties	Pages
037079/0001	Oct 27, 2015	Nov 9, 2015	11870	1112

Conveyance
ASSIGNMENT OF ASSIGNORS INTEREST (SEE DOCUMENT FOR DETAILS).

Assignors
HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P.

Correspondent
HEWLETT PACKARD ENTERPRISE
3404 E. HARMONY ROAD MS 79
FORT COLLINS, CO 80528

Assignee
HEWLETT PACKARD ENTERPRISE DEVELOPMENT LP
11445 COMPAQ CENTER DRIVE WEST
HOUSTON, TEXAS 77070

Assignment 1

Reel/frame	Execution date	Date recorded	Properties	Pages
015932/0117	Oct 19, 2004	Oct 25, 2004	1	3

Conveyance
ASSIGNMENT OF ASSIGNORS INTEREST (SEE DOCUMENT FOR DETAILS).

Assignors
MICHAELIS, SCOTT L.
RAJKUMARI, ANURUPA
MCHARDY, WILLIAM B.

Correspondent
HEWLETT-PACKARD COMPANY
INTELLECTUAL PROPERTY ADMINISTRATION
RECORDS MANAGER
P.O. BOX 272400
FORT COLLINS, COLORADO 80527-2400

Assignee
HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P.
20555 SH 249
HOUSTON, TEXAS 77070

EXHIBIT B

PATENT

U.S. District Court [LIVE]
Western District of Texas (Austin)
CIVIL DOCKET FOR CASE #: 1:23-cv-00640-RP

Foras Technologies Ltd. v. Nissan Motor Company, Ltd. et al
Assigned to: Judge Robert Pitman
Related Case: 6:23-cv-00386-RP
Cause: 35:271 Patent Infringement

Date Filed: 06/06/2023
Jury Demand: Plaintiff
Nature of Suit: 830 Patent
Jurisdiction: Federal Question

Plaintiff

Foras Technologies Ltd.

represented by **Jonathan Randy Yim**
BC Law Group, P.C.
200 Madison Avenue
24th Floor
New York, NY 10016
516-359-9668
Email: jyim@bclgpc.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Ashley Marie Ratycz
BC Law Group, P.C.
200 Madison Avenue, 24th Floor
New York, NY 10016
212-951-0100
Email: aratycz@bclgpc.com
ATTORNEY TO BE NOTICED

Drew B Hollander
BC Law Group, P.C.
200 Madison Avenue
24th Floor
New York, NY 10016
212-951-0100
Email: dhollander@bclgpc.com
ATTORNEY TO BE NOTICED

Seth Raymond Hasenour
BC Law Group, P.C.
607 Terrell Hill Drive
Ste Unit B
Austin, TX 78704
516-359-9668
Email: shasenour@bclgpc.com
ATTORNEY TO BE NOTICED

Brett E Cooper
BC LAW GROUP, P.C.
200 Madison Avenue, 24th Floor
New York, NY 10016

516-359-9668
Email: bcooper@b-clg.com
ATTORNEY TO BE NOTICED

V.

Defendant

Nissan Motor Company, Ltd.

represented by

Binal J. Patel
Banner & Witcoff, LTD
71 S. Wacker Drive, Ste 3600
Chicago, IL 60606
312-463-5463
Fax: 312-463-5001
Email: bpatel@bannerwitcoff.com
LEAD ATTORNEY
PRO HAC VICE
ATTORNEY TO BE NOTICED

John R. Hutchins
Banner & Witcoff, Ltd.
1100 13th St. Nw, Suite 1200
Washington, DC 20005
202-824-3000
Fax: 202-824-3001
Email: jhutchins@bannerwitcoff.com
LEAD ATTORNEY
PRO HAC VICE
ATTORNEY TO BE NOTICED

Paige Arnette Amstutz
Scott, Douglass & McConnico, LLP
303 Colorado Street
Suite 2400
Austin, TX 78701
(512) 495-6300
Fax: 512-495-6399
Email: pamstutz@scottdoug.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Paul Thomas Qualey
Banner & Witcoff, LTD
1100 13th St., NW, Suite 1200
Washington D.C., DC 20005
202-824-3000
Fax: 202-824-3001
Email: pqualey@bannerwitcoff.com
LEAD ATTORNEY
PRO HAC VICE
ATTORNEY TO BE NOTICED

Wesley W. Jones
Banner & Witcoff, Ltd.

1100 13th St. Nw, Suite 1200
Washington, DC 20005
202-824-3000
Fax: 202-824-3001
Email: wjones@bannerwitcoff.com
LEAD ATTORNEY
PRO HAC VICE
ATTORNEY TO BE NOTICED

Defendant

ZF Friedrichshafen AG

represented by **Binal J. Patel**
(See above for address)
LEAD ATTORNEY
PRO HAC VICE
ATTORNEY TO BE NOTICED

John R. Hutchins
(See above for address)
LEAD ATTORNEY
PRO HAC VICE
ATTORNEY TO BE NOTICED

Paige Arnette Amstutz
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Paul Thomas Qualey
(See above for address)
LEAD ATTORNEY
PRO HAC VICE
ATTORNEY TO BE NOTICED

Wesley W. Jones
(See above for address)
LEAD ATTORNEY
PRO HAC VICE
ATTORNEY TO BE NOTICED

Date Filed	#	Docket Text
06/06/2023	1	COMPLAINT (Filing fee $ 402 receipt number ATXWDC-17522685), filed by Foras Technologies Ltd.. (Attachments: # 1 Exhibit 1, # 2 Exhibit 2, # 3 Exhibit 3, # 4 Exhibit 4, # 5 Exhibit 5, # 6 Exhibit 6, # 7 Exhibit 7, # 8 Exhibit 8, # 9 Exhibit 9, # 10 Exhibit 10, # 11 Exhibit 11, # 12 Civil Cover Sheet)(Cooper, Brett) (Entered: 06/06/2023)
06/06/2023	2	Notice of Filing of Patent/Trademark Form (AO 120). AO 120 forwarded to the Director of the U.S. Patent and Trademark Office. (Cooper, Brett) (Entered: 06/06/2023)
06/06/2023	3	RULE 7 DISCLOSURE STATEMENT filed by Foras Technologies Ltd.. (Cooper, Brett) (Entered: 06/06/2023)
06/06/2023	4	NOTICE *Notice of Related Cases* by Foras Technologies Ltd. (Cooper, Brett) (Entered: 06/06/2023)

06/06/2023	5	REQUEST FOR ISSUANCE OF SUMMONS by Foras Technologies Ltd.. *as to Nissan Motor Company, Ltd.* (Cooper, Brett) (Entered: 06/06/2023)
06/06/2023	6	REQUEST FOR ISSUANCE OF SUMMONS by Foras Technologies Ltd.. *as to ZF Friedrichshafen AG* (Cooper, Brett) (Entered: 06/06/2023)
06/06/2023		Case assigned to Judge Robert Pitman. CM WILL NOW REFLECT THE JUDGE INITIALS AS PART OF THE CASE NUMBER. PLEASE APPEND THESE JUDGE INITIALS TO THE CASE NUMBER ON EACH DOCUMENT THAT YOU FILE IN THIS CASE. (pg) (Entered: 06/09/2023)
06/06/2023		If ordered by the court, all referrals and consents in this case will be assigned to Magistrate Judge Howell (pg) (Entered: 06/09/2023)
06/09/2023	7	Summons Issued as to Nissan Motor Company, Ltd. (pg) (Entered: 06/09/2023)
06/09/2023	8	Summons Issued as to ZF Friedrichshafen AG. (pg) (Entered: 06/09/2023)
06/09/2023	9	Report on Patent/Trademark sent to U.S. Patent and Trademark Office with Complaint. (pg) (Entered: 06/09/2023)
09/12/2023	10	MOTION *for Order Appointing International Process Server* by Foras Technologies Ltd.. (Attachments: # 1 Proposed Order)(Cooper, Brett) (Entered: 09/12/2023)
09/27/2023	11	ORDER GRANTING 10 MOTION for Order Appointing International Process Server. Signed by Judge Robert Pitman. (cc3) (Entered: 09/27/2023)
12/04/2023	12	Unopposed MOTION for Extension of Time to File Response/Reply as to 1 Complaint, *ZF Friedrichshafen AG's Unopposed Motion for Extension of Time to Respond to the Complaint* by ZF Friedrichshafen AG. (Attachments: # 1 Proposed Order Proposed Order) (Amstutz, Paige) (Entered: 12/04/2023)
12/05/2023		Text Order GRANTING 12 Unopposed Motion for Extension of Time to File Response/Reply entered by Judge Robert Pitman. IT IS ORDERED that ZF FRIEDRICHSHAFEN AG shall answer or otherwise respond to the complaint on or before January 8, 2024. (This is a text-only entry generated by the court. There is no document associated with this entry.) (sh) (Entered: 12/05/2023)
12/05/2023		Reset Answer Deadline: ZF Friedrichshafen AG answer due 1/8/2024. (cc3) (Entered: 12/05/2023)
12/06/2023	13	SUMMONS Returned Executed by Foras Technologies Ltd.. ZF Friedrichshafen AG served on 11/13/2023, answer due 1/8/2024. (Cooper, Brett) (Entered: 12/06/2023)
12/28/2023	14	Unopposed MOTION for Extension of Time to File Answer re 1 Complaint, by Nissan Motor Company, Ltd.. (Attachments: # 1 Proposed Order to Unopposed Motion)(Amstutz, Paige) (Entered: 12/28/2023)
01/02/2024		Text Order GRANTING 14 Unopposed Motion for Extension of Time to Answer entered by Judge Robert Pitman. IT IS ORDERED that Nissan shall answer or otherwise respond to the complaint on or before February 9, 2024. (This is a text-only entry generated by the court. There is no document associated with this entry.) (sh) (Entered: 01/02/2024)
01/02/2024		Reset Answer Deadline: Nissan Motor Company, Ltd. answer due 2/9/2024. (cc3) (Entered: 01/02/2024)
01/08/2024	15	MOTION to Dismiss for Lack of Jurisdiction *Defendant ZF Friedrichshafen AG's Motion to Dismiss the Complaint Pursuant to Fed. R. Civ. P. 12(b)(2)* by ZF Friedrichshafen AG. (Attachments: # 1 Affidavit Declaration of David Council ISO ZF Friedrichshafen AG's Motion to Dismiss Plaintiff's Complaint)(Amstutz, Paige) (Entered: 01/08/2024)

01/10/2024	16	RULE 7 DISCLOSURE STATEMENT filed by ZF Friedrichshafen AG. (Amstutz, Paige) (Entered: 01/10/2024)
01/16/2024	17	Unopposed MOTION for Extension of Time to File Response/Reply as to 15 MOTION to Dismiss for Lack of Jurisdiction *Defendant ZF Friedrichshafen AG's Motion to Dismiss the Complaint Pursuant to Fed. R. Civ. P. 12(b)(2)* by Foras Technologies Ltd.. (Attachments: # 1 Proposed Order)(Cooper, Brett) (Entered: 01/16/2024)
01/18/2024		Text Order GRANTING 17 Unopposed Motion for Extension of Time to File Response/Reply entered by Judge Robert Pitman. IT IS ORDERED that Plaintiff shall respond on or before January 29, 2024. (This is a text-only entry generated by the court. There is no document associated with this entry.) (sh) (Entered: 01/18/2024)
01/29/2024	18	Response in Opposition to Motion, filed by Foras Technologies Ltd., re 15 MOTION to Dismiss for Lack of Jurisdiction *Defendant ZF Friedrichshafen AG's Motion to Dismiss the Complaint Pursuant to Fed. R. Civ. P. 12(b)(2)* filed by Defendant ZF Friedrichshafen AG (Cooper, Brett) (Entered: 01/29/2024)
02/05/2024	19	RESPONSE in Support, filed by ZF Friedrichshafen AG, re 15 MOTION to Dismiss for Lack of Jurisdiction *Defendant ZF Friedrichshafen AG's Motion to Dismiss the Complaint Pursuant to Fed. R. Civ. P. 12(b)(2)* filed by Defendant ZF Friedrichshafen AG *Defendant ZF Friedrichshafen AG's Reply Brief In Support of its Motion to Dismiss the Complaint Pursuant to Fed. R. Civ. P. 12(b)(2)* (Amstutz, Paige) (Entered: 02/05/2024)
02/06/2024	20	SUMMONS Returned Executed by Foras Technologies Ltd.. Nissan Motor Company, Ltd. served on 12/20/2023, answer due 2/9/2024. (Cooper, Brett) (Entered: 02/06/2024)
02/09/2024	21	NOTICE *of Translation to English* by Foras Technologies Ltd. re 20 Summons Returned Executed (Cooper, Brett) (Entered: 02/09/2024)
02/09/2024	22	NOTICE *of Translation to English* by Foras Technologies Ltd. re 13 Summons Returned Executed (Cooper, Brett) (Entered: 02/09/2024)
02/09/2024	23	Motion to Dismiss for Lack of Jurisdiction *Defendant Nissan Motor Company, Ltd.'s Motion to Dismiss the Complaint Pursuant to Fed. R. Civ. P. 12(b)(2)* by Nissan Motor Company, Ltd.. (Attachments: # 1 Affidavit Declaration of Jacquelyn Trussell ISO Nissan Motor Company, Ltd.'s Motion to Dismiss Plaintiff's Complaint)(Amstutz, Paige) Modified on 2/12/2024 To edit text (bc). (Entered: 02/09/2024)
02/23/2024	24	Response in Opposition to Motion, filed by Foras Technologies Ltd., re 23 MOTION to Dismiss for Lack of Jurisdiction filed by Defendant Nissan Motor Company, Ltd. (Cooper, Brett) (Entered: 02/23/2024)
03/01/2024	25	RESPONSE in Support, filed by Nissan Motor Company, Ltd., re 23 MOTION to Dismiss for Lack of Jurisdiction filed by Defendant Nissan Motor Company, Ltd. *Defendant Nissan Motor Company Ltd.'s Reply in Support of Its Motion to Dismiss the Complaint Pursuant to Fed. R. Civ. P. 12(b)(2)* (Amstutz, Paige) (Entered: 03/01/2024)
03/13/2024	26	Order for parties to submit a Joint Proposed Scheduling Order; due by 3/27/2024. Signed by Judge Dustin M. Howell. (cc3) (Entered: 03/13/2024)
03/21/2024	27	RULE 7 DISCLOSURE STATEMENT filed by Nissan Motor Company, Ltd. identifying Corporate Parent None for Nissan Motor Company, Ltd.. (Amstutz, Paige) (Entered: 03/21/2024)
03/22/2024	28	CERTIFICATE OF SERVICE by Nissan Motor Company, Ltd. 27 Disclosure Statement (Rule 7) (Amstutz, Paige) (Entered: 03/22/2024)

03/22/2024	29	MOTION to Appear Pro Hac Vice by Paige Arnette Amstutz *for Wesley W. Jones* (Filing fee $ 100 receipt number ATXWDC-18562555) by on behalf of Nissan Motor Company, Ltd., ZF Friedrichshafen AG. (Attachments: # 1 Proposed Order)(Amstutz, Paige) (Entered: 03/22/2024)
03/22/2024	30	MOTION to Appear Pro Hac Vice by Paige Arnette Amstutz *for Binal J. Patel* (Filing fee $ 100 receipt number ATXWDC-18562636) by on behalf of Nissan Motor Company, Ltd., ZF Friedrichshafen AG. (Attachments: # 1 Proposed Order)(Amstutz, Paige) (Entered: 03/22/2024)
03/22/2024	31	MOTION to Appear Pro Hac Vice by Paige Arnette Amstutz *for John R. Hutchins* (Filing fee $ 100 receipt number ATXWDC-18562645) by on behalf of Nissan Motor Company, Ltd., ZF Friedrichshafen AG. (Attachments: # 1 Proposed Order)(Amstutz, Paige) (Entered: 03/22/2024)
03/27/2024	32	ORDER GRANTING 29 Motion to Appear Pro Hac Vice as to WESLEY W. JONES. Pursuant to our Administrative Policies and Procedures for Electronic Filing, the attorney hereby granted to practice pro hac vice in this case must register for electronic filing with our court within 10 days of this order. Registration is managed by the PACER Service Center Signed by Judge Robert Pitman. (cc3) (Entered: 03/27/2024)
03/27/2024	33	ORDER GRANTING 30 Motion to Appear Pro Hac Vice as to BINAL J. PATEL. Pursuant to our Administrative Policies and Procedures for Electronic Filing, the attorney hereby granted to practice pro hac vice in this case must register for electronic filing with our court within 10 days of this order. Registration is managed by the PACER Service Center Signed by Judge Robert Pitman. (cc3) (Entered: 03/27/2024)
03/27/2024	34	ORDER GRANTING 31 Motion to Appear Pro Hac Vice as to JOHN R. HUTCHINS. Pursuant to our Administrative Policies and Procedures for Electronic Filing, the attorney hereby granted to practice pro hac vice in this case must register for electronic filing with our court within 10 days of this order. Registration is managed by the PACER Service Center Signed by Judge Robert Pitman. (cc3) (Entered: 03/27/2024)
03/27/2024	35	NOTICE of Attorney Appearance by Ashley Marie Ratycz on behalf of Foras Technologies Ltd.. Attorney Ashley Marie Ratycz added to party Foras Technologies Ltd.(pty:pla) (Ratycz, Ashley) (Entered: 03/27/2024)
03/27/2024	36	Rule 26(f) Discovery Report/Case Management Plan - *Joint Rule 26 Report* by Foras Technologies Ltd.. (Cooper, Brett) (Entered: 03/27/2024)
03/27/2024	37	Joint MOTION for Entry of Scheduling Order Deadlines by Foras Technologies Ltd.. (Attachments: # 1 Exhibit A - Agreed Proposed Scheduling Order)(Cooper, Brett) (Entered: 03/27/2024)
04/02/2024	38	MOTION to Appear Pro Hac Vice by Paige Arnette Amstutz *Motion for Admission Pro Hac Vice Paul T. Qualey for Nissan Motor Company, Ltd. and ZF Friedrichshafen AG* (Filing fee $ 100 receipt number ATXWDC-18597104) by on behalf of Nissan Motor Company, Ltd., ZF Friedrichshafen AG. (Attachments: # 1 Proposed Order Proposed Order)(Amstutz, Paige) (Entered: 04/02/2024)
04/03/2024	39	ORDER GRANTING 38 Motion to Appear Pro Hac Vice as to PAUL T. QUALEY. Pursuant to our Administrative Policies and Procedures for Electronic Filing, the attorney hereby granted to practice pro hac vice in this case must register for electronic filing with our court within 10 days of this order. Registration is managed by the PACER Service Center Signed by Judge Robert Pitman. (cc3) (Entered: 04/03/2024)
04/09/2024	40	ORDER Setting Initial Pretrial Conference for 4/22/2024 01:30 PM before Judge Dustin M. Howell. Signed by Judge Dustin M. Howell. (bot2) (Entered: 04/09/2024)

04/10/2024	41	Opposed MOTION to Stay Case by Nissan Motor Company, Ltd., ZF Friedrichshafen AG. (Attachments: # 1 Affidavit of John R. Hutchins, # 2 Exhibit 1, # 3 Exhibit 2, # 4 Exhibit 3, # 5 Exhibit 4, # 6 Exhibit 5, # 7 Exhibit 6, # 8 Exhibit 7, # 9 Exhibit 8, # 10 Exhibit 9, # 11 Exhibit 10, # 12 Exhibit 11, # 13 Exhibit 12, # 14 Proposed Order Granting Motion to Stay)(Hutchins, John) (Entered: 04/10/2024)
04/15/2024	42	Unopposed MOTION for Extension of Time to File Response/Reply as to 41 Opposed MOTION to Stay Case by Foras Technologies Ltd.. (Attachments: # 1 Proposed Order) (Cooper, Brett) (Entered: 04/15/2024)
04/17/2024		Text Order GRANTING 42 Unopposed Motion for Extension of Time to File Response/Reply entered by Judge Robert Pitman. IT IS ORDERED that Plaintiff shall respond to the motion to stay on or before April 26, 2024. (This is a text-only entry generated by the court. There is no document associated with this entry.) (sh) (Entered: 04/17/2024)

PACER Service Center			
Transaction Receipt			
04/22/2024 13:10:37			
PACER Login:	apapageorgiou	**Client Code:**	foras
Description:	Docket Report	**Search Criteria:**	1:23-cv-00640-RP
Billable Pages:	6	**Cost:**	0.60

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
90/019,243	09/01/2023	7624302	185945.012600

CONFIRMATION NO. 1079

89220
Antonio Papageorgiou
Lombard Geliebter LLP
230 PARK AVENUE 4TH FL WEST
New York, NY 10169

POA ACCEPTANCE LETTER

OC000000071519384

Date Mailed: 04/24/2024

NOTICE OF ACCEPTANCE OF POWER OF ATTORNEY

This is in response to the Power of Attorney filed 04/23/2024.

The Power of Attorney in this application is accepted. Correspondence in this application will be mailed to the above address as provided by 37 CFR 1.33.

Questions about the contents of this notice and the requirements it sets forth should be directed to the Office of Data Management, Application Assistance Unit, at **(571) 272-4000** or **(571) 272-4200** or **1-888-786-0101**.

/pflewis/

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
90/019,243	09/01/2023	7624302	185945.012600

CONFIRMATION NO. 1079

56436
Hewlett Packard Enterprise
3404 E. Harmony Road
Mail Stop 79
Fort Collins, CO 80528

POWER OF ATTORNEY NOTICE

DC20000000715193565

Date Mailed: 04/24/2024

NOTICE REGARDING CHANGE OF POWER OF ATTORNEY

This is in response to the Power of Attorney filed 04/23/2024.

- The Power of Attorney to you in this application has been revoked by the assignee who has intervened as provided by 37 CFR 3.71. Future correspondence will be mailed to the new address of record(37 CFR 1.33).

Questions about the contents of this notice and the requirements it sets forth should be directed to the Office of Data Management, Application Assistance Unit, at **(571) 272-4000** or **(571) 272-4200** or **1-888-786-0101**.

/pflewis/

UNITED STATES
PATENT AND TRADEMARK OFFICE

P.O. Box 1450
Alexandria, VA 22313 - 1450
www.uspto.gov

ELECTRONIC ACKNOWLEDGEMENT RECEIPT

APPLICATION #	RECEIPT DATE / TIME	ATTORNEY DOCKET #
90/019,243	**04/24/2024 10:43:56 AM Z ET**	**185945.012600**

Title of Invention

SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

Application Information

APPLICATION TYPE	Utility / ex parte reexam	PATENT #	7624302
CONFIRMATION #	1079	FILED BY	Laura Alto
PATENT CENTER #	65221791	FILING DATE	09/01/2023
CUSTOMER #	89220	FIRST NAMED INVENTOR	- 7624302
CORRESPONDENCE ADDRESS	-	AUTHORIZED BY	Antonio Papageorgiou

Documents

TOTAL DOCUMENTS: 1

DOCUMENT	PAGES	DESCRIPTION	SIZE (KB)
FTL-REX243_Cert_of_Service_04-24-2024.pdf	1	Reexam Certificate of Service	68 KB

Digest

DOCUMENT	MESSAGE DIGEST(SHA-512)
FTL-REX243_Cert_of_Service_04-24-2024.pdf	DBFA4EF42C9800D2424ADA41253445DD027D7C83848E47675 86D9FBCB8EB62E282F9B936971BE3D9E85A0ADE23305B1D89 18910C11BADD283CC61FB57C8CC6C5

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent of	:	Michaelis, Scott L, et al.		
Application No.	:	90/019,243	Confirmation No. :	1079
Patent No.	:	7,624,302		
Filing Date	:	October 25, 2004	Issue Date:	November 24, 2009
Title	:	SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR		

CERTIFICATE OF SERVICE

I hereby certify that a true and correct copy of the foregoing Request for an Extension of Time Pursuant to 37 C.F.R. 1.550(c); Declaration of Brett Cooper in Support of Petition to for Extension of Time to Respond to a Reexamination Office Action Pursuant to 37 CFR 1.550(c), Power of Attorney and Assignee showing of ownership per 37 CFR 3.73 was electronically filed on April 23, 2024 and served upon counsel of record via First Class Mail on April 24, 2024 to:

GREENBERG TRAURIG (NJ)
500 CAMPUS DRIVE, SUITE 400
P.O. BOX677
FLORHAM PARK NJ 07932

Date: April 24, 2024

Customer No. 89220

/Antonio Papageorgiou/
Antonio Papageorgiou
USPTO Reg. No. 53431

Attorney for Patent Owner, Foras Technologies Ltd

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
90/019,243	09/01/2023	7624302	185945.012600	1079

89220 7590 04/25/2024

Antonio Papageorgiou
Lombard Geliebter LLP
230 PARK AVENUE 4TH FL WEST
New York, NY 10169

EXAMINER	
TON, MY TRANG	

ART UNIT	PAPER NUMBER
3992	

MAIL DATE	DELIVERY MODE
04/25/2024	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Commissioner for Patents
United States Patents and Trademark Office
P.O.Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

4/25/2024

THIRD PARTY REQUESTER'S CORRESPONDENCE ADDRESS

GREENBERG TRAURIG (NJ)
500 CAMPUS DRIVE, SUITE 400
P.O. BOX 677
FLORHAM PARK, NEW JERSEY 07932

EX PARTE REEXAMINATION COMMUNICATION TRANSMITTAL FORM

REEXAMINATION CONTROL NO. : 90019243
PATENT NO. : 7624302
ART UNIT : 3992

Enclosed is a copy of the latest communication from the United States Patent and Trademark Office in the above-identified ex parte reexamination proceeding (37 CFR 1.550(f)).

Where this copy is supplied after the reply by requester, 37 CFR 1.535, or the time for filing a reply has passed, no submission on behalf of the ex parte reexamination requester will be acknowledged or considered (37 CFR 1.550(g)).

Decision on Petition for Extension of Time in Reexamination	**Application No.:** 90/019,243	**Applicant(s):** 7,624,302	
	Examiner TON, MY TRANG	**Art Unit** 3992	

1. THIS IS A DECISION ON THE PETITION FILED 04/23/2024.

2. THIS DECISION IS ISSUED PURSUANT TO:
   - A. ☒ 37 CFR 1.550(c) – The time for taking any action by a patent owner in a third party requested *ex parte* reexamination proceeding will be extended only for sufficient cause and for a reasonable time specified.
   - B. ☐ 37 CFR 1.550(c) – The time for taking action by a patent owner in a patent owner requested *ex parte* reexamination proceeding will only be extended for more than two months for sufficient cause and for a reasonable time specified.
   - C. ☐ 37 CFR 1.956 – The time for taking any action by a patent owner in an *inter partes* reexamination proceeding will be extended only for sufficient cause and for a reasonable time specified.

   The petition is before the Central Reexamination Unit for consideration.

3. FORMAL MATTERS: Patent Owner requests that the time period for filing a response to the non-final Office action mailed on 02/23/2024, which set a two (2) month shortened statutory period for response, be extended by one (1) month.
   - A. Petition fee per 37 CFR § 1.17(g)):
     - i. ☐ Petition includes authorization to debit a deposit account.
     - ii. ☒ Petition includes authorization to charge a credit card account.
     - iii. ☐ Other: _____.
   - B. ☒ Proper certificate of service was provided. (Not required in reexamination where patent owner is requester).
   - C. ☒ Petition was timely filed.
   - D. ☒ Petition properly signed.

4. DECISION (See MPEP 2265 and 2665)
   - A. ☒ Granted or ☐ Granted-in-part for _____. (See 37 CFR 1.550(c) and 37 CFR 1.956).
     - i. ☐ Other/comment: _____.
   - B. ☐ Dismissed because:
     - i. ☐ Formal matters (See unchecked box(es) (A, B, C and/or D) in section 3 above).
     - ii. ☐ Petitioner failed to provide a factual accounting of reasonably diligent behavior by all those responsible for preparing a response to the outstanding Office action within the statutory time period.
     - iii. ☐ Petitioner failed to explain why, in spite of the action taken thus far, the requested additional time is needed.
     - iv. ☐ The statements provided fail to establish sufficient cause to warrant extension of the time for taking action (See attached).
     - v. ☐ The petition is moot.
     - vi. ☐ Other/comment: _____.

5. CONCLUSION:

Direct telephone inquiries to either QAS Stephen J. Stein at 571-272-1544 or SPRS Andrew J. Fischer at 571-272-6779.	/Andrew J. Fischer/, Supervisory Patent Reexamination Specialist, Central Reexamination Unit

UNITED STATES
PATENT AND TRADEMARK OFFICE

P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

CORRESPONDENCE ADDRESS CHANGE

APPLICATION #	FILING OR 371(C) DATE	REQUEST ID
90/019,243	09/01/2023	206518

The following fields have been changed to Customer Number 199393 on 04/26/2024 via Patent Center in view of the certification copied below that authorized the change.

- Correspondence Address

The address for Customer Number 199393 is:

199393 - Atlantic IP
c/o Lombard Geliebter LLP
1325 Sixth Aveneue
28th Floor
New York , NY 10019

I certify, in accordance with 37 CFR 1.4(d)(4), that I am :

An attorney or Agent of Record registered to practice before the Patent and Trademark Office who has been given power of attorney in this application

Signature	Name	Registration #
/Antonio PAPAGEORGIOU/	Antonio Papageorgiou	53431

UNITED STATES
PATENT AND TRADEMARK OFFICE

P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

FEE ADDRESS CHANGE

APPLICATION #	FILING OR 371(C) DATE	REQUEST ID
90/019,243	09/01/2023	206518

The following fields have been changed to Customer Number 199393 on 04/26/2024 via Patent Center in view of the certification copied below that authorized the change.

- Maintenance Fee Address

The address for Customer Number 199393 is:

199393 - Atlantic IP
c/o Lombard Geliebter LLP
1325 Sixth Aveneue
28th Floor
New York, NY 10019

I certify, in accordance with 37 CFR 1.4(d)(4), that I am :

An attorney or Agent of Record registered to practice before the Patent and Trademark Office who has been given power of attorney in this application

Signature	**Name**	**Registration #**
/Antonio PAPAGEORGIOU/	Antonio Papageorgiou	53431

Organization CRU Bldg./Room

UNITED STATES PATENT AND TRADEMARK OFFICE
P.O. Box 1450
Alexandria, VA. 22313-1450
If Undeliverable Return In Ten Days

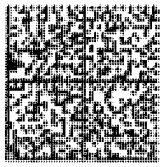
Official Business
Penalty For Private Use, $300

AN EQUAL OPPORTUNITY EMPLOYER

RECEIVED

MAY 06 2024

USPTO MAIL CENTER



US POSTAGE PITNEY BOWES
ZIP 22314 $ 000.64⁰
02 4W
0000355389 APR. 29. 2024

IXIE 100 FE 1 0005/02/24

RETURN TO SENDER
NOT DELIVERABLE AS ADDRESSED
UNABLE TO FORWARD

BC: 22313145050 *1992-05502-29-42

OPAP LHDTSV-7
MAY 0 6 2024
PATENT & TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
90/019,243	09/01/2023	7624302	185945.012600	1079

89220 7590 04/25/2024

Antonio Papageorgiou
Lombard Geliebter LLP
230 PARK AVENUE 4TH FL WEST
New York, NY 10169

EXAMINER	
TON, MY TRANG	

ART UNIT	PAPER NUMBER
3992	

MAIL DATE	DELIVERY MODE
04/25/2024	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patents and Trademark Office
P.O.Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

4/25/2024

THIRD PARTY REQUESTER'S CORRESPONDENCE ADDRESS

GREENBERG TRAURIG (NJ)
500 CAMPUS DRIVE, SUITE 400
P.O. BOX 677
FLORHAM PARK, NEW JERSEY 07932

EX PARTE REEXAMINATION COMMUNICATION TRANSMITTAL FORM

REEXAMINATION CONTROL NO. : 90019243
PATENT NO. : 7624302
ART UNIT : 3992

Enclosed is a copy of the latest communication from the United States Patent and Trademark Office in the above-identified ex parte reexamination proceeding (37 CFR 1.550(f)).

Where this copy is supplied after the reply by requester, 37 CFR 1.535, or the time for filing a reply has passed, no submission on behalf of the ex parte reexamination requester will be acknowledged or considered (37 CFR 1.550(g)).

Decision on Petition for Extension of Time in Reexamination	Application No.: 90/019,243	Applicant(s): 7,624,302	
	Examiner TON, MY TRANG	Art Unit 3992	

1. THIS IS A DECISION ON THE PETITION FILED 04/23/2024.

2. THIS DECISION IS ISSUED PURSUANT TO:
   - A. ☒ 37 CFR 1.550(c) – The time for taking any action by a patent owner in a third party requested *ex parte* reexamination proceeding will be extended only for sufficient cause and for a reasonable time specified.
   - B. ☐ 37 CFR 1.550(c) – The time for taking action by a patent owner in a patent owner requested *ex parte* reexamination proceeding will only be extended for more than two months for sufficient cause and for a reasonable time specified.
   - C. ☐ 37 CFR 1.956 – The time for taking any action by a patent owner in an *inter partes* reexamination proceeding will be extended only for sufficient cause and for a reasonable time specified.

   The petition is before the Central Reexamination Unit for consideration.

3. FORMAL MATTERS: Patent Owner requests that the time period for filing a response to the non-final Office action mailed on 02/23/2024, which set a two (2) month shortened statutory period for response, be extended by one (1) month.
   - A. Petition fee per 37 CFR § 1.17(g)):
     - i. ☐ Petition includes authorization to debit a deposit account.
     - ii. ☒ Petition includes authorization to charge a credit card account.
     - iii. ☐ Other: _____.
   - B. ☒ Proper certificate of service was provided. (Not required in reexamination where patent owner is requester).
   - C. ☒ Petition was timely filed.
   - D. ☒ Petition properly signed.

4. DECISION (See MPEP 2265 and 2665)
   - A. ☒ Granted or ☐ Granted-in-part for _____. (See 37 CFR 1.550(c) and 37 CFR 1.956).
     - i. ☐ Other/comment: _____.
   - B. ☐ Dismissed because:
     - i. ☐ Formal matters (See unchecked box(es) (A, B, C and/or D) in section 3 above).
     - ii. ☐ Petitioner failed to provide a factual accounting of reasonably diligent behavior by all those responsible for preparing a response to the outstanding Office action within the statutory time period.
     - iii. ☐ Petitioner failed to explain why, in spite of the action taken thus far, the requested additional time is needed.
     - iv. ☐ The statements provided fail to establish sufficient cause to warrant extension of the time for taking action (See attached).
     - v. ☐ The petition is moot.
     - vi. ☐ Other/comment: _____.

5. CONCLUSION:

Direct telephone inquiries to either QAS Stephen J. Stein at 571-272-1544 or SPRS Andrew J. Fischer at 571-272-6779.	/Andrew J. Fischer/, Supervisory Patent Reexamination Specialist, Central Reexamination Unit

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re.	:	Michaelis, Scott L, et al.		
Reexam Cont. No.	:	90/019,243	Confirmation No. :	1079
Patent No.	:	7,624,302	App. No.	: 10/972,588
Issue Date	:	11/24/2009	Filing Date	: 10/25/2004
Title	:	SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR		

RESPONSE TO NON-FINAL OFFICE ACTION

This response is submitted in reply to the Non-Final Office Action in the above referenced Reexamination dated February 23, 2024. A one-month extension of time was granted, which extends the time for this response by May 23, 2024.

Remarks begin on Page 2.

I. Remarks

A. Status of Claims

Claims 1, 5-8, 13, 17, 21-22, and 25-26 are subject to reexamination. The patentability of claims 21-22 and 25-26 are confirmed. Claims 1, 5-8, 13, and 17 were rejected.

B. Identification of the Real Party-In-Interest

Foras Technologies Limited is the Patent Owner and the real party-in-interest.

C. Related Proceedings

U.S. Patent 7,624,302 (the "'302 Patent") is the subject of the following U.S. District Court proceedings:

1. *Foras Technologies, Ltd vs Nissan Motor Co., Ltd*, Case No. 1 :23-cv-00640 (W.D. Tex., June 6, 2023).

2. *Foras Technologies Ltd v. Toyota Motor North America, Inc., et al.*, Case No. 2:23-cv-00150 (E.D. Tex., April 5, 2023).

II. Response to the Non-Final Office Action (NFOA)

A. Overview of the '302 Patent

Silent Data Corruption ("SDC") is a difficult problem in computing. '302 Patent, 1:55-56. The '302 Patent indicates that "[n]umerous techniques have been developed for detecting SDC to prevent the SDC from remaining "silent" or "undetected" within a system, as well as preventing such SDC from propagating through the system." *Id. at* 2:1-4. For example, "Parity/ECC mechanisms have been employed extensively for caches, memories, and similar data storage arrays. In the remaining circuitry on a processor, such as data paths, control logic, execution logic, and registers (the "execution core"), it is more difficult to apply parity/ECC mechanisms for SDC

detection. Thus, there is typically some unprotected area on a processor in which data corruption may occur and the parity/ECC mechanisms do not prevent the corrupted data from actually making it out onto the system bus." *Id. at* 2:19-26.

The '302 Patent further provides that a "pair of lockstep processors may, from time to time, lose their lockstep. "Loss of lockstep" (or "LOL") is used broadly to refer to any error in the pair of lockstep processors." *Id. at* 2:40-42. "If the processors are instructed to act on the data of their first cache locations, then a lockstep mismatch will occur as the output of each of the processors will not match. For instance, if the processors each add the data stored to the first location of their respective cache with the data stored to the second location of their respective cache, the first processor (having the corrupt data) will output a result of "1" (0+1=1) while the second processor outputs a result of "2" (1+1=2), and thus their respective outputs will not match." *Id. at* 2:64-3:6.

"Prior solutions attempting to resolve at least some detected SDCs without requiring the system to be crashed have been Operating System ("OS") centric." *Id. at* 3:31-33. The OS centric approach places a heavy burden on most commonly used operating systems, such that the current state of the art did not employ lockstep recovery. *Id. at* 3:40-42. Moreover, based on the evidence of record, it does not appear that boot processors were configured to operate in lockstep at the time of the '302 Patent and, correspondingly, there is no evidence that computer systems were configured to determine loss of lockstep during the boot process. The '302 Patent alleviates these problems in the art by assigning a role of boot processor and a role of spare processor to distinct processors and having logic, responsive to detecting loss of lockstep in the boot processor, for switching the role of the boot processor that has lost lockstep to the spare processor. *Id. at* Abstract, 3:64-4:11, *see also, id.* at Fig. 5.

B. Legal Standard

In rejecting patent claims, the burden lies solely on the Office to show that the claims of a patent are unpatentable. *See* MPEP 2131 (anticipation); MPEP 2141 (obviousness).

To establish prima facie obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974); *In re Wilson*, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970); *see also* MPEP 2143.03. "Rejections on obviousness cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness" MPEP 2141 (quoting *KSR International Co. v. Teleflex Inc.*, 550 U.S. 398, 418 (2007)), and there must be a showing of reasonable expectation of success (MPEP 2143.03). Moreover, it is improper to parse the invention into individual components. Rather, the claimed invention must be considered as a whole. MPEP 2141.02(I).

C. The claims of the '302 Patent References

The '302 Patent includes 27 claims: Claims 1, 13, 21 and 26 are independent:

Claim 1 recites: A method comprising:

> *detecting loss of lockstep (LOL) for a processor in a multi-processor system*;
> ***determining that the processor for which said LOL is detected is assigned a role of boot processor;***
> ***changing a system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected, to a different value, which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor; and***
> *changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to a spare processor without shutting down the system's operating system.* (emphasis added).

Claim 13 recites: A method comprising:

system firmware determining that loss of lockstep (LOL) is detected for a lockstep pair of processors that are assigned the role of boot processor in a system;

determining one of said lockstep pair of processors that is not the cause of said LOL;

copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to a spare processor;

changing a system identifier, used by the system's operating system to recognize the boot processor, of said lockstep pair of processors that are assigned the role of boot processor, to a different value, which causes the s system's operating system to not recognize the lock step pair of processors for which said LOL is detected as the processor; and

changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to said spare processor. (emphasis added)

The Patent Owner submits that the references of record fail to disclose or otherwise suggest at least the limitations emphasized above and claims 1, 5-8, 13, and 17 are therefore patentable over these references for at least these reasons.

D. The Rejections

In the NFOA, the Office presented one ground of rejection as:

Rejection 1 - Claims 1, 5-8, 13, 17, were rejected under 35 U.S.C. 103(a) as being obvious over U.S. Patent No. 7,328,371 to Kalyanasundaram ("Kalyanasundaram) in view of U.S. Patent No. 5,450,576 to Kennedy ("Kennedy"). NFOA at p. 6.

These rejections are respectfully traversed.

E. Rejection 1, The proposed combination of Kalyanasundharam with Kennedy fails to teach or suggest each and every element of Independent Claims 1 and 13

1. Kalyanasundaram

Kalyanasundharam relates generally to chip multiprocessors (CMPs). Kalyanasundharam at 1:6-7. Kalyanasundharam notes that early detection of faults in CMPs improve reliability, but

detecting discrepancies between CMPs can be costly and complex. *Id.* at 1:20-35. Kalyanasundharam's attempt to address such issues is with a node controller that controls processors cores to operate in a redundant execution mode. *Id.* at 1:39-51. Specifically, in this mode, the Kalyanasundharam system and more specifically, "the node controller 20A may route the communication concurrently to each processor core 18A-18E3 (block 74). For example, in the embodiment of FIG. 2, the communication may be scheduled when it can be routed to each processor core 18A-18E3, and may be routed through the crossbar 34 to both processor cores 18A-18E3 concurrently." *Id.* at 11:20-25. If an error is detected, a processor core may be disabled or replaced with a redundant core. *Id.* at 11:34-59. Kalyanasundharam does not disclose that the redundant execution mode operates during the boot process or with the use of firmware. Rather, Kalyanasundharam performs redundancy checks via a controller 20A, 20B on the multi-processor chips or nodes 12A, 12B, respectively. *See id.* at Fig. 1-2, 5, 7 (all showing the redundancy checking being performed on CMP chips.); *see also, id.* at 14:4-9. Moreover, when an error is detected, Kalyanasundharam will disable the primary core and cycle through to the next available core, i.e., core 0 to N-1. *Id.* at 13:16-34. Specifically, the "control circuit 84 may also update the PC field of the configuration register 42 to reflect the newly selected primary processor core." *Id.* at 13:27-29, *see also, id.* at Fig. 2 and 5 (showing register 42 on CMP redundancy check circuit 26A. In other words, Kalyanasundharam performs these functions on the processor hardware, not via firmware or in the operating system.

2. Kennedy

Kennedy provides: "In the multiprocessor system 100 which utilizes a system bus 102, ***a number of problems can arise*** if more than one processor type is installed on the 64-bit bus 104. For instance, ***provision needs to be made to determine which processor*** will boot the system, and

if this "default" boot processor fails to boot the system, an alternative processor, ***if one is installed***, should take over boot operations to enhance the availability of the system." Kennedy at 7:5-13 (emphasis added). For example, Kennedy discuses an embodiment with three CPUs: "the CPU1 120 installed in slot 1, the CPU2 132 installed in slot 2, and a CPU3160 installed in slot 3 with a corresponding ISTC memory 161 and a slot select circuit 148C." *Id.* at 6:25-29.

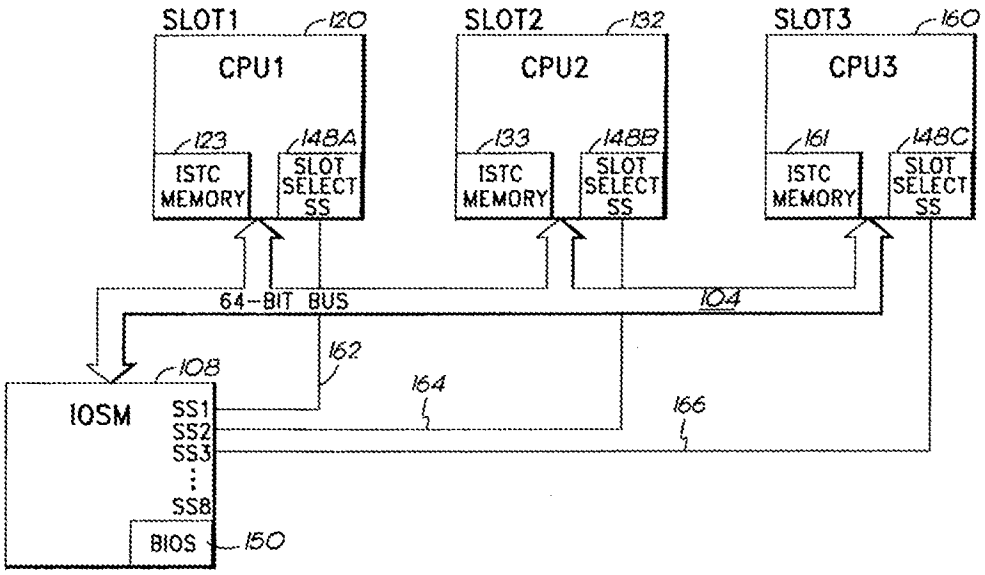


FIG. 3

Fig. 3 (shown above) further illustrates the connections from the slot select circuits 148A, 148B, and 148C to the IOSM 108. As seen in FIG. 3, the slot select circuit 148A for the CPU 120 *physically* connects to the IOSM 108 with a slot select signal line (SS1) 162. Kennedy, 6:29-33. As noted in Kennedy, "the slot select circuits 148B and 148C connect to the IOSM108 with slot select signal lines (SS2 and SS3) 164 and 166 respectively. Any other slot select circuits on CPUs installed in the system have a corresponding slot select signal line from the IOSM 108." Kennedy, 6:33-38. Kennedy cycles through the processors in succession, much like Kalyanasundharam. *Id.*

at 13:38-44. None of these Kennedy's processors work in lockstep, nor is Kennedy's physical architecture capable of having processors CPU1 – CPUN work in lockstep.

In other words, Kennedy recognizes that in a multiprocessor system, a default processor will be used to boot the system, and if that processor fails, an alternative processor will be used. Kennedy only describes the operation of the single boot/alternative processors being booted in succession at initialization. *Id.* at 7:3 *et sec.* More importantly, Kennedy teaches a specific solution that involves architecture in which the default boot processor is located in the ***physical*** first slot and any alternative processors in ***physical*** secondary slots. *See id.* at 2:13-15, 2:22-27, 2:30-38. Therefore, the default and alternative distinctions appear to be based on the physical location of the processors, not based on a software based designation.

3. Claims 1 and 13: *"determining that the processor for which said LOL is detected is assigned a role of boot processor"*

The Examiner contends that Kalyanasundharam describes the use of multiple pairs of processors (18A-18B) that operate in lockstep. NFOA, p.43. The Examiner relies on a portion of Kalyanasundharam where a mismatch between a primary processor and auxiliary processor core is detected. *Id.* (citing Kalyanasundharam at 13:16-22; 13:60-64). But, as the Examiner correctly acknowledges, there is no mention of a boot processor anywhere in Kalyanasundharam. *Id.* Additionally, Kalyanasundharam is devoid of any discussion to teach or suggest the act of *determining* that the processor *for which the loss of lockstep is detected* is *assigned* a particular role, namely the boot processor role, as claimed. As shown in Fig. 5 of the '302 Patent, block 505, this determination is a discrete step. *See also*, '302 Patent at 17:32-36. In contrast, in the Kalyanasundharam system, the controller cycles through the available processor cores upon error detection in the primary core. Kalyanasundharam at 13:16-42. The designation of "primary"

generally refers to the active processor that provides the communications that the node controller will propagate through the system. *See id.* at 4:36-40. Because in the Kalyanasundharam, error detection is performed only for the active/primary processor core, that designation is a given and there is therefore no deed to determine whether the error is in the primary processor. Kalyanasundharam therefore does not disclose or suggest discreetly *determining the role of the processor found to have lost lockstep*, as recited in claims 1 and 13.

To overcome the deficiency acknowledged by the Examiner, the Examiner asserts that Kennedy discusses "a multiprocessor system that identifies default boot processors and alternative boot processors should the default boot processor fail to boot." NFOA p. 43 (citing Kennedy at 7:5-14 and 13:38-44). Kennedy, however, does not cure Kalyanasundharam's shortcomings. As discussed above, Kennedy recognizes that in a multiprocessor system, a default processor will be used to boot the system, and if that processor fails, an alternative processor will be used. Kennedy at 7:5-14. Kennedy, not unlike Kalyanasundharam, cycles through the available processors CPU1-3 in slots 1-3 (*id.* at 13:38-44) and there is therefore no discrete determination of the role of the defective "default" processor.

Moreover, the Examiner has not explained why one skilled in the art would combine Kennedy, which only describes the operation of single boot/alternative processors being booted in succession at initialization (*id.* at 7:3 *et sec.*, *infra*), with Kalyanasundharam, which relates to parallel processing ***after*** the boot process (*see* Kalyanasundharam at 13:60-64). Indeed, the record does not contain any evidence to show that lockstep processing during the boot process is even possible, let alone desirable. Rather, the art of record shows that parallel redundant processing is a solution to avoid crashing the system during application processing, i.e., after the boot process. *Id.* (explaining the redundant process in relation to the execution of software, i.e., after the

operating system has booted). In contrast, Kennedy teaches default and alternative processors operating alternatively and in succession at the time that the system is ***initialized***, *i.e.*, before the operating system boot process. Kennedy at 7:4-13; 8:5-8 ("When a circuit board is initializing, the board specific ISTC is transferred from the associated ISTC memory to the memory 130 by the boot processor and assembled into executable code."), *id.* at 9:53-55 ("Once each CPU in the system has executed its ISTC and has checked in, the CPU1 120 controls the system again and begins the operating system boot process."). Because Kennedy relates to the startup process only, loss in system availability is not a concern because the time to boot a processor is relatively short, as compared to booting the operating system and the associated applications for continuity. Because Kalyanasundharam and Kennedy provide solutions unrelated contextually to each other, a person of ordinary skilled in the art would not look to combine Kennedy with any of the other reference of record.

Furthermore, Kennedy teaches an architecture in which the default boot processor in the physical first slot and any alternative processors are in physical secondary slots. *See* Kennedy at 2:13-15, 2:22-27, 2:30-38. One skilled in the art would not modify Kennedy to yield the claimed invention involving switchable system identifiers for boot and spare processors, because doing so would frustrate the purpose of Kennedy's physical architecture as a solution with regard to overcoming problems associated with which processors boot the system. *See id.* at 1:60-2:12; 2:29-38.

The Wolfe declaration cited by the Examiner for the rationale and expectation of success contains mere conclusory statements that ignore these distinctions between Kalyanasundharam and Kennedy. Conclusory statements by the third party requestor's expert cannot meet the Office's burden with respect to either the rationale or the expectation of success of the proposed

Kalyanasundharam and Kennedy combination. *See In re NuVasive, Inc.*, 842 F.3d 1376, 120 U.S.P.Q.2d 1821 (Fed. Cir. 2016).

Accordingly, the Office has failed to show that claims 1 and 13, and the claims dependent thereon, are unpatentable for at least these reasons.

4. **Claim 1 and 13:** ***"changing a system identifier, used by the system's operating system to recognize the boot processor … to a different value, which causes the [] operating system*** **to not recognize** ***the processor for which said LOL is detected as the boot processor"***

With respect to ***"changing a system identifier, used by the system's operating system to recognize the boot processor…to a different value"***, as recited in claims 1 and 13, the Examiner contends that "Kalyanasundharam discloses in Fig. 7 a control circuit 84 that can chang[e] a system identifier." NFOA at p. 46 (citing Kalyanasundharam at 13:19-29). The Examiner also cites to Kennedy for this element at 13:38-44 and 15:9-14 and for the "…***which causes the [] operating system to not recognize the processor for which said LOL is detected as the boot processor***". *Id.* at 47-45. The Examiner is mistaken.

As discussed herein, Kalyanasundharam performs redundancy checks via a controller 20A, 20B on the multi-processor chips or nodes 12A, 12B, respectively. *See* Kalyanasundharam at Fig. 1-2, 5, 7 (all showing the redundancy checking being performed on CMP chips.); *see also, id.* at 14:4-9. Moreover, when an error is detected, Kalyanasundharam will disable the primary core and cycle through to the next available core, i.e., core 0 to N-1. *Id.* at 13:16-34. Specifically, the "control circuit 84 may also update the PC field of the configuration register 42 to reflect the newly selected primary processor core." *Id.* at 13:27-29, *see also, id.* at Fig. 2 and 5 (showing register 42 on CMP redundancy check circuit 26A). In other words, Kalyanasundharam performs these functions on the processor hardware, not via firmware or in the operating system. Kennedy

similarly cycles through processors during the boot process using hardware, *i.e.*, physical slots, only, not via the operating system. Indeed, the references relied on by the Examiner do not mention the role of the operating system even once.

The Examiner's reliance on the discussion of Kalyanasundharam's control circuit for this element is therefore misplaced. Kalyanasundharam at 13:19-29 merely discusses that "the control circuit 84 may disable the primary processor core and select one of the auxiliary processor cores to be the primary processor core (block102)", which may be accomplished by "clearing the corresponding enable bit in the ***configuration register 42***, or [by updating] the PC field of the ***configuration register 42*** to reflect the newly selected primary processor core." The registers and the control circuit are both on the CMP redundancy check circuit. *Id.* at 13:19-29 (emphasis added); Fig. 5 (showing register 42 and circuity 84 on CMP redundancy check circuit 26A).

Moreover, the information from register 42 is used in Kalyanasundharam by the comparator 40, ***which is also on the CMP redundancy check circuit***. *See id.* at 7:14-16, Fig. 2 (ref. no. 40); *see also*, *id.* at 8:63-67 ("In the illustrated embodiment, the comparator circuit 40 may receive the communications stored in the flops 38A 38B for comparison. If valid communications are received and a mismatch is detected by the comparator circuit 40, the redundancy check circuit 26A may signal an execution error."); *id. at* 10:5-10 ("In one embodiment, the machine check exception may be delivered to the primary processor core. If a valid communication is received from each processor core (decision block 54, 'yes' leg), the redundancy check circuit 26A may determine if the communications match (decision block 58)…"), *id. at* 12:57-61 ("If redundant execution mode is active (decision block 90, 'yes' leg), the control circuit 84 may mask the comparison results (from the comparator circuits 82A-82N) according to which processor cores are currently enabled, as indicated in the configuration register

42 (block 94)."). This is consistent with the '302 Patent, which recognizes OS centric attempts to resolve processor errors were disfavored because of higher processing demands and burdensome implementation. '302 Patent at 3:31-42.

Kalyanasundharam does not therefore disclose "***a system identifier, used by the system's operating system to recognize the boot processor***" as claimed. As such, Kalyanasundharam fails to disclose changing the system identifier such that "***the [] operating system [will] not recognize the processor for which said LOL is detected as the boot processor***", as recited in claims 1 and 13.

Kennedy at 13:38-44 and 15:9-14 does not close this gap. Kennedy recites "If the CPU1 120 in slot 1 does not boot (proper boot is indicated by a change in the level of the SLOT_ SELECT signal line 193 from high to low) before the timer 184 in slot 2 times-out, then the CPU2 132 exits the reset state and attempts to boot the system as an alternate boot CPU." Kennedy at 15:9-14. Like Kalyanasundharam, the failure of Kennedy's CPU in slot 1 to boot is not "***a system identifier to be used by the system's operating system to recognize the boot processor***." Indeed, Kennedy teaches default and alternative processors operating alternatively and in succession at the time that the system is ***initialized***, *i.e.*, ***before the operating system boot process***. Kennedy at 7:4-13; 8:5-8 ("When a circuit board is initializing, the board specific ISTC is transferred from the associated ISTC memory to the memory 130 by the boot processor and assembled into executable code."), *id. at* 9:53-55 ("Once each CPU in the system has executed its ISTC and has checked in, the CPU1 120 controls the system again and begins the operating system boot process.").

Kennedy does not therefore disclose "***a system identifier, used by the system's operating system to recognize the boot processor***" as claimed. As such, Kennedy fails to disclose changing

the system identifier such that "***the [] operating system [will] not recognize the processor for which said LOL is detected as the boot processor***", as recited in claims 1 and 13.

For the reasons discussed above, there is also no motivation to combine and modify Kalyanasundharam and Kennedy to yield the claimed invention. Accordingly, the Office has failed to show that claims 1 and 13, and the claims dependent thereon, are unpatentable for at least these reasons.

The Examiner has therefore failed to make a *prima facie* case of obviousness with regard to claims 1-27 for at least the reasons discussed above.

III. Conclusion

The other dependent claims are patentable over the references of record for the same reasons noted above, as well as additional reasons. The Patent Owner reserves the right to argue additional distinctions in its discretion.

In view of the foregoing, the Patent Owner respectfully requests that the Examiner certify the patentability of claims 1-27 of the '302 Patent.

Date: May 23, 2024

Respectfully submitted,

/Antonio Papageorgiou/
Antonio Papageorgiou
Reg. No. 53,431

UNITED STATES
PATENT AND TRADEMARK OFFICE

P.O. Box 1450
Alexandria, VA 22313 - 1450
www.uspto.gov

ELECTRONIC ACKNOWLEDGEMENT RECEIPT

APPLICATION #	RECEIPT DATE / TIME	ATTORNEY DOCKET #
90/019,243	**05/23/2024 08:57:18 PM Z ET**	**185945.012600**

Title of Invention

SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

Application Information

APPLICATION TYPE	Utility / ex parte reexam	PATENT #	7624302
CONFIRMATION #	1079	FILED BY	Antonio Papageorgiou
PATENT CENTER #	65654995	FILING DATE	09/01/2023
CUSTOMER #	199393	FIRST NAMED INVENTOR	- 7624302
CORRESPONDENCE ADDRESS	-	AUTHORIZED BY	-

Documents

TOTAL DOCUMENTS: 2

DOCUMENT	PAGES	DESCRIPTION	SIZE (KB)
FTL-REX243_Cert_of_Service.pdf	1	Reexam Certificate of Service	22 KB
20240523_243_302_Response_to_Non-Final_Office_Action.pdf	14	Amendment/Request for Reconsideration-After Non-Final Rejection	230 KB

Digest

DOCUMENT	MESSAGE DIGEST(SHA-512)
FTL-	85A90D30AA7999010BD61BDBE4C85041C12D9D78F67234B05

REX243_Cert_of_Service.pdf	B338DF3461C3539515C6E014BA1813CF48DA07768ADC23E44CB76EEB07F4B83CB80AA25E0FCC27D
20240523_243_302_Response_to_Non-Final_Office_Action.pdf	06D0D6A0FF2FCDB6776DC842C45225C9B9F7B7054FA5FF90230FEEAF9FE9A103FB7DC8ED14F9EC28DE4F280928CACD4F59F15313568C086BE563D623EC5017B4

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re.	:	Michaelis, Scott L, et al.		
Reexam Cont. No.	:	90/019,243	Confirmation No. :	1079
Patent No.	:	7,624,302	App. No.	: 10/972,588
Issue Date	:	11/24/2009	Filing Date	: 10/25/2004
Title	:	SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR		

CERTIFICATE OF SERVICE

I hereby certify that a true and correct copy of the **Response to Office Action** was electronically filed with the USPTO and served upon counsel of record for the third-party requestor via email on May 23, 2024 with a courtesy copy via First Class Mail to:

Jared Lee
GREENBERG TRAURIG
222 South Main Street, Suite 1730
Salt Lake City, UT 84101
Jared.Lee@gtlaw.com

GREENBERG TRAURIG (NJ)
500 Campus Drive, Suite 400
P.O. Box 677
Florham Park, NJ 07932

Date: May 23, 2024

Customer No. 89220

/Antonio Papageorgiou/
Antonio Papageorgiou
USPTO Reg. No. 53431

Attorney for Patent Owner,
Foras Technologies Ltd

Litigation Search Report CRU 3999

Reexamination Control No. 90/019,243

To: My Trang Ton	From: Renee Preston
Location: CRU	Location: CRU 3999
Art Unit: 3992	Phone: (571) 272-1607
Date: 06/06/2024	E-mail: renee.preston@uspto.gov

Search Notes

U.S. Patent No. 7,624,302

1. Performed a KeyCite Search in Westlaw, which retrieves all history on the patent including any litigation.
2. Performed a search on the patent in Lexis CourtLink for any open dockets or closed cases.
3. Performed a search in Lexis in the Federal Courts and Administrative Materials databases for any cases found.
4. Performed a search in Lexis in the IP Journal and Periodicals database for any articles on the patent.
5. Performed a search in Lexis in the news databases for any articles about the patent or any articles about litigation on this patent.

Litigation: Cases Found

Status	Docket	Description
Open	1:23cv640	Foras Technologies Ltd. V. Nissan Motor Company, Ltd. Et Al
Closed	2:23cv150	Foras Technologies Ltd. V. Toyota Motor North America, Inc. Et Al

Citing References (13)

Treatment	Title	Date	Type	Depth	Headnote(s)
—	**1.** BOOT PROCESSOR ROLE SWITCHING METHOD FOR MULTIPROCESSOR SYSTEM, INVOLVES DETERMINING WHETHER PROCESSOR FOR WHICH LOSS OF LOCKSTEP DETECTED IS ASSIGNED ROLE OF BOOT PROCESSOR, AND SWITCHING ROLE OF BOOT TO SPARE PROCESSOR Out Of Plan DWPI 2006-328202	Oct. 25, 2004	DWPI	—	—
—	**2.** RF 058992/0571 Out Of Plan	Feb. 11, 2022	Assignments	—	—
—	**3.** RF 052567/0734 Out Of Plan	May 05, 2020	Assignments	—	—
—	**4.** RF 037079/0001 Out Of Plan	Nov. 09, 2015	Assignments	—	—
—	**6.** RF 015932/0117 Out Of Plan	Oct. 25, 2004	Assignments	—	—
—	**7.** PatStat 7624302 Out Of Plan	Mar. 09, 2010	Patent Status Files	—	—
—	**8.** INTEGRATED CIRCUIT CHIP WITH CORES ASYMMETRICALLY ORIENTED WITH RESPECT TO EACH OTHER Out Of Plan US PAT 11698841 , U.S. PTO Utility	July 11, 2023	Patents	—	—
—	**9.** INTEGRATED CIRCUIT CHIP WITH CORES ASYMMETRICALLY ORIENTED WITH RESPECT TO EACH OTHER Out Of Plan US PAT 11269742 , U.S. PTO Utility	Mar. 08, 2022	Patents	—	—
—	**10.** INTEGRATED CIRCUIT CHIP WITH CORES ASYMMETRICALLY ORIENTED WITH RESPECT TO EACH OTHER Out Of Plan US PAT 10649865 , U.S. PTO Utility	May 12, 2020	Patents	—	—
—	**11.** INTEGRATED CIRCUIT CHIP WITH CORES ASYMMETRICALLY ORIENTED WITH RESPECT TO EACH OTHER Out Of Plan US PAT 10002056 , U.S. PTO Utility	June 19, 2018	Patents	—	—
—	**12.** STORAGE SYSTEM AND CONTROL METHOD OF STORAGE SYSTEM Out Of Plan US PAT 8312314 , U.S. PTO Utility	Nov. 13, 2012	Patents	—	—
—	**13.** STORAGE SYSTEM AND CONTROL METHOD OF STORAGE SYSTEM Out Of Plan US PAT 7958391 , U.S. PTO Utility	June 07, 2011	Patents	—	—
—	**14.** STORAGE SYSTEM AND CONTROL METHOD OF STORAGE SYSTEM Out Of Plan US PAT 7809981 , U.S. PTO Utility	Oct. 05, 2010	Patents	—	—

WESTLAW © 2025 Thomson Reuters. No claim to original U.S. Government Works.

☐ US7624302B2 - System and method for switching the role of boot processor to a spare processor responsive to detection of loss of lockstep in a boot processor

Original

Bibliographic information & Images

Publication number and date
US7624302B2
2009-11-24

Granted

Application number and date
US10972588 (/api/control?client=&query=AN:("US10972588")&autosearch=true)
2004-10-25

Priority number and date
US10972588 (/api/control?client=&query=PRN:("US10972588")&autosearch=true)
2004-10-25

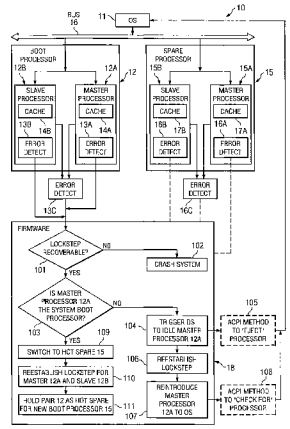
Ultimate owner
FORAS TECHNOLOGIES (/api/control?client=&query=uo:("FORAS%20TECHNOLOGIES")&autosearch=true)

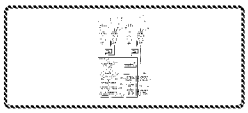
Original assignee
Scott L. Michaelis (/api/control?client=&query=pa:("Scott%20L.%20Michaelis")&autosearch=true)

Current assignee
SONRAÍ MEMORY, LTD. (/api/control?client=&query=pacu:("SONRA%C3%8D%20MEMORY%2C%20LTD.")&autosearch=true)

Inventor
Scott L. Michaelis (/api/control?client=&query=in:("Scott%20L.%20Michaelis")&autosearch=true)

1/5 Clipped image









Abstract

According to one embodiment, a method comprises detecting loss of lockstep (LOL) for a processor in a multi-processor system. The method further comprises determining that the processor for which the LOL is detected is assigned the role of boot processor, and switching the role of boot processor to a spare processor without shutting down the system's operating system. In another embodiment, a method comprises system firmware determining that an LOL is detected for a lockstep pair of processors that are assigned the role of boot processor in a system. The method further comprises determining one of the lockstep pair of processors that is not the cause of the LOL, and copying the state of the determined one of the lockstep pair of processors that is not the cause of the LOL to a spare processor. The method further comprises switching the role of boot processor to the spare processor.

Original

Claims

1. A method comprising: detecting loss of lockstep (LOL) for a processor in a multi-processor system; determining that the processor for which said LOL is detected is assigned a role of boot processor; changing a system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected, to a different value, which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor; and changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to a spare processor without shutting down the system's operating system.

2. The method of claim 1 wherein the spare processor is a hot spare that is held in idle until switched the role of boot processor.

3. The method of claim 1 wherein the spare processor is an active spare.

4. The method of claim 3 wherein said switching further comprises: causing the system's operating system to idle the active spare.

5. The method of claim 1 wherein the spare processor is a processor module that includes a master and slave processors that operate in lockstep.

6. The method of claim 1 wherein said determining that the processor for which said LOL is detected is assigned the role of boot processor comprises: accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor.

7. The method of claim 1 further comprising: determining the processor that is assigned the role of spare processor.

8. The method of claim 7 wherein said determining the processor that is assigned the role of spare processor comprises: accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of spare processor.

9. The method of claim 1 wherein said determining step is performed by system firmware.

10. The method of claim 1 wherein said switching step is performed by system firmware.

11. The method of claim 1 wherein the processor for which said LOL is detected includes a lockstep pair of processors, and wherein said switching comprises: determining one of the lockstep pair of processors that is not the cause of said LOL; and copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to said spare processor.

12. The method of claim 11 wherein said spare processor includes a lockstep pair of processors, and wherein said copying comprises: copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to each of said lockstep pair of processors of said spare processor.

13. A method comprising: system firmware determining that loss of lockstep (LOL) is detected for a lockstep pair of processors that are assigned the role of boot processor in a system; determining one of said lockstep pair of processors that is not the cause of said LOL; copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to a spare processor; changing a system identifier, used by the system's operating system to recognize the boot processor, of said lockstep pair of processors that are assigned the role of boot processor, to a different value, which causes the s system's operating system to not recognize the lock step pair of processors for which said LOL is detected as the processor; and changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to said spare processor.

14. The method of claim 13 wherein the spare processor is a hot spare that is held in idle until switched the role of boot processor.

15. The method of claim 13 wherein the spare processor is an active spare.

16. The method of claim 15 further comprising: causing the system's operating system to idle the active spare.

17. The method of claim 13 wherein said switching causes the system's operating system to recognize said spare processor as said boot processor without shutting down said operating system.

18. The method of claim 13 wherein said determining step is performed by system firmware.

19. The method of claim 13 wherein said copying step is performed by system firmware.

20. The method of claim 13 wherein said switching step is performed by system firmware.

21. A system comprising: a plurality of processor modules that each include a master processor and a slave processor that operate in lockstep; an operating system; error detection logic operable to detect loss of lockstep (LOL) for at least one of said processor modules; and system firmware operable, responsive to the detection of LOL for a first of said processor modules, to determine whether said first processor module is assigned a role of system boot processor; wherein if determined that said first processor module is assigned the role of system boot processor, said system firmware is further operable to cause said operating system to recognize another processor module as system boot processor without shutting down said operating system.

22. The system of claim 21 wherein the another processor module is a processor module assigned a role of spare processor.

23. The system of claim 21 wherein the another processor module is held in idle until said system firmware causes said operating system to recognize said another processor module as said system boot processor.

24. The system of claim 21 wherein the system firmware causes the operating system to idle the another processor module before causing said operating system to recognize said another processor module as said system boot processor.

25. The system of claim 21 further comprising: non-volatile data storage storing information that assigns the role of system boot processor to one of said plurality of processor modules.

26. A system comprising: system firmware means for detecting loss of lockstep (LOL) for a processor in a multi-processor system; means for determining whether the processor for which said LOL is detected is assigned the role of boot processor; and system firmware means for switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which said LOL is detected is assigned the role of boot processor.

27. The system of claim 26 wherein said processor in said multi-processor system is an Itanium Processor Family (IPF) processor.

Original

Families

Display in a list (/api/control?client=&familiesSearch=_) INPADOC

This section contains all family relations of this particular document. For more information about the different families, refer to the User Manual (Glossary/Families).

Extended (2) Priority date/claims	Complete (2) Priority date/claims	Main (2) Priority date/claims	Domestic (2) Priority date/claims	Publication number	Application number	Title	Status
2004-10-25 US10972588	2004-10-25 US10972588	2004-10-25 US10972588	2004-10-25	US7624302B2	US10972588	System and method for switching the role of boot processor to a spare processor responsive to detection of loss of lockstep in a boot processor	Granted
				US20060090064A1	US10972588	System and method for switching the role of boot processor to a spare processor responsive to detection of loss of lockstep in a boot processor	Granted

Description

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] The present application is related to the following concurrently filed and commonly assigned U.S. Patent Applications: U.S. patent application Ser. No. 10/973,077 (U.S. Published Patent Application 20060107112) titled "SYSTEM AND METHOD FOR ESTABLISHING A SPARE PROCESSOR FOR RECOVERING FROM LOSS OF LOCKSTEP IN A BOOT PROCESSOR"; U.S. patent application Ser. No. 10/973,004 (U.S. Published Patent Application 20060107117, now U.S. Pat. No. 7,308,566) titled "SYSTEM AND METHOD FOR CONFIGURING LOCKSTEP MODE OF A PROCESSOR MODULE"; U.S. patent application Ser. No, 10/972,835 (U.S. Published Patent Application 20060107114) titled "SYSTEM AND METHOD FOR USING INFORMATION RELATING TO A DETECTED LOSS OF LOCKSTEP FOR DETERMINING A RESPONSIVE ACTION"; U.S. patent application Ser. No. 10/972,588 (U.S. Published Patent Application 20060090064) titled "SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR"; U.S. patent application Ser. No. 10/973,003 (U.S. Published Patent Application 20060107116) titled "SYSTEM AND METHOD FOR REESTABLISHING LOCKSTEP FOR A PROCESSOR MODULE FOR WHICH LOSS OF LOCKSThP IS DETECTED"; U.S. patent application Ser. No. 10/972,888 (U.S. Published Patent Application 20060107115) titled "SYSTEM AND METHOD FOR SYSTEM FIRMWARE CAUSING AN OPERATING SYSTEM TO IDLE A PROCESSOR"; U.S. patent application Ser. No. 10/973,075 (U.S. Published Patent Application 20060107111) titled "SYSTEM AND METHOD FOR REINTRODUCING A PROCESSOR MODULE TO AN OPERATING SYSTEM AFTER LOCKSTEP RECOVERY"; U.S. patent application Ser. No. 10/972,796 (U.S. Published Patent Application 20060107106) titled "SYSTEM AND METHOD FOR MAINTAINING IN A MULTI-PROCESSOR SYSTEM A SPARE PROCESSOR THAT IS IN LOCKSTEP FOR USE IN RECOVERING FROM LOSS OF LOCKSTEP FOR ANOTHER PROCESSOR", the disclosures of which arc hereby incorporated herein by reference.

DESCRIPTION OF RELATED ART

[0002] Silent Data Corruption ("SDC") is a difficult problem in the computing industry. In general, SDC refers to data that is corrupt, but which the system does not detect as being corrupt. SDCs primarily occur due to one of two factors: a) a broken hardware unit or b) a "cosmic" event that causes values to change somewhere in the system. Broken hardware means that a "trusted" piece of hardware is silently giving wrong answers. For example, the arithmetic unit in a processor is instructed to add 1+1 and it returns the incorrect answer 3 instead of the correct answer 2. An example of a cosmic event is when a charged particle (e.g., alpha particle or cosmic ray) strikes a region of a computing system and causes some bits to change value (e.g., from a 0 to a 1 or from a 1 to a 0).

[0003] Numerous techniques have been developed for detecting SDC to prevent the SDC from remaining "silent" or "undetected" within a system, as well as preventing such SDC from propagating through the system. Examples of these techniques include parity-based mechanisms and error correcting codes (ECCs) on buses and memory locations, as well as checksums and/or cyclic redundancy checks (CRC) over regions of memory. Parity-based mechanisms are often employed in processors, wherein a parity bit is associated with each block of data when it is stored. The parity bit is set to one or zero according to whether there is an odd or even number of ones in the data block. When the data block is read out of its storage location, the number of ones in the block is compared with the parity bit. A discrepancy between the values indicates that the data block has been corrupted. ECCs are parity-based mechanisms that track additional information for each data block. The additional information allows the corrupted bit(s) to be identified and corrected.

[0004] Parity/ECC mechanisms have been employed extensively for caches, memories, and similar data storage arrays. In the remaining circuitry on a processor, such as data paths, control logic, execution logic, and registers (the "execution core"), it is more difficult to apply parity/ECC mechanisms for SDC detection. Thus, there is typically some unprotected area on a processor in which data corruption may occur and the parity/ECC mechanisms do not prevent the corrupted data from actually making it out onto the system bus. One approach to SDC detection in an execution core (or other unprotected area of the processor chip) is to employ "lockstep processing." Generally, in lockstep processing two processors are paired together, and the two processors perform exactly the same operations and the results are compared (e.g., with an XOR gate). If there is ever a discrepancy between the results of the lockstep processors, an error is signaled. The odds of two processors experiencing the exact same error at the exact same moment (e.g., due to a cosmic event occurring in both processors at exactly the same time or due to a mechanical failure occurring in each processor at exactly the same time) is nearly zero.

[0005] A pair of lockstep processors may, from time to time, lose their lockstep. "Loss of lockstep" (or "LOL") is used broadly herein to refer to any error in the pair of lockstep processors. One example of LOL is detection of data corruption (e.g., data cache error) in one of the processors by a parity-based mechanism and/or ECC mechanism. Another example of LOL is detection of the output of the paired processors not matching, which is referred to herein as a "lockstep mismatch." It should be recognized that in some cases the data in the cache of a processor may become corrupt (e.g., due to a cosmic event), which once detected (e.g., by a parity-based mechanism or ECC mechanism of the processor) results in LOL. Of course, unless such corrupt data is acted upon by the processor, the output of that processor will not fail to match the output of its paired processor and thus a "lockstep mismatch" will not occur. For example, suppose that a value of "1" is stored to first location of cache in each of a pair of lockstep processors and a value of "1" is also stored to a second location of cache in each of the pair of lockstep processors. Further suppose that a cosmic event occurs for a first one of the processors, resulting in the first location of its cache being changed from "1" to "0", and thus corrupted. This data corruption in the first processor is a LOL for the pair. An error detection mechanism of this first processor may detect the data corruption, thus detecting the LOL. If the processors are instructed to act on the data of their first cache locations, then a lockstep mismatch will occur as the output of each of the processors will not match. For instance, if the processors each add the data stored to the first location of their respective cache with the data stored to the second location of their respective cache, the first processor (having the corrupt data) will output a result of "1" (0+1=1) while the second processor outputs a result of "2" (1+1=2), and thus their respective outputs will not match.

[0006] By employing such techniques as parity-based error detection mechanisms and output comparisons for lockstep paired processors, SDC detection can be enhanced such that practically no SDC occurring in a processor goes undetected (and thus such SDC does not remain "silent") but instead results in detection of LOL. However, the issue then becomes how best for the system to respond to detected LOL. The traditional response to detected LOL has been to crash the system to ensure that the detected error is not propagated through the system. That is, LOL in one pair of lockstep processors in a system halts processing of the system even if other processors that have not encountered an error are present in the system. However, with the increased desire for many systems to maintain high availability, crashing the system each time LOL is detected is not an attractive proposition. This is particularly unattractive for large systems having many processors because cosmic events typically occur more frequently as the processor count goes up, which would result in much more frequent system crashes in those large systems. High availability is a major desire for many customers having large, multi-processor systems, and thus having their system crash every few weeks is not an attractive option. Of course, permitting corrupt data to propagate through the system is also not a viable option.

[0007] Prior solutions attempting to resolve at least some detected SDCs without requiring the system to be crashed have been Operating System ("OS") centric. That is, in certain solutions the OS has been implemented in a manner to recover from a detected LOL without necessarily crashing the system. This OS-centric type of solution requires a lot of processor and platform specific knowledge to be embedded in the OS, and thus requires that the OS provider maintain the OS up-to-date as changes occur in later versions of the processors and platforms in which the OS is to be used. This is such a large burden that most commonly used OSs do not support lockstep recovery.

[0008] Certain solutions have attempted to recover from a LOL without involving the OS in such recovery procedure. For instance, in one technique upon LOL being detected, firmware is used to save the state of one of the processors in a lockstep pair (the processor that is considered "good") to memory, and then both processors of the pair are reset and reinitialized. Thereafter, the state is copied from the memory to each of the processors in the lockstep pair. This technique makes the processors unavailable for an amount of time without the OS having any knowledge regarding this unavailability, and if the amount of time required for recovery is too long, the system may crash. That is, typically, if a processor is unresponsive for X amount of time, the OS will assume that the processor is hung and will crashdump the system so that the problem can be diagnosed. Further, in the event that a processor in the pair cannot be reset and reinitialized (e.g., the processor has a physical problem and fails to pass its self-test), this technique results in crashing the system.

BRIEF SUMMARY OF THE INVENTION

[0009] According to one embodiment, a method comprises detecting loss of lockstep (LOL) for a processor in a multi-processor system. The method further comprises determining that the processor for which the LOL is detected is assigned the role of boot processor, and switching the role of boot processor to a spare processor without shutting down the system's operating system.

[0010] According to one embodiment, a method comprises system firmware determining that loss of lockstep (LOL) is detected for a lockstep pair of processors that are assigned the role of boot processor in a system. The method further comprises determining one of the lockstep pair of processors that is not the cause of the LOL, and copying the state of the determined one of the lockstep pair of processors that is not the cause of the LOL to a spare processor. The method further comprises switching the role of boot processor to the spare processor.

[0011] According to one embodiment, a system comprises a plurality of processor modules that each include a master processor and a slave processor that operate in lockstep. The system further comprises an operating system, and error detection logic operable to detect loss of lockstep (LOL) for at least one of the processor modules. The system further comprises system firmware operable, responsive to detection of LOL for a first of the processor modules, to determine whether the first processor module is assigned a role of system boot processor. If determined that the first processor module is assigned the role of system boot processor, the system firmware is further operable to cause the operating system to recognize another processor module as system boot processor without shutting down the operating system.

[0012] According to one embodiment, a system comprises means for detecting loss of lockstep (LOL) for a processor in a multi-processor system. The system further comprises means for determining whether the processor for which the LOL is detected is assigned the role of boot processor, and means for switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which the LOL is detected is assigned the role of boot processor.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] FIG. 1 shows an example multi-processor system in which firmware utilizes a spare processor for recovering from LOL for the system's boot processor;

[0014] FIG. 2 shows a block diagram of one embodiment implemented for the IA-64 processor architecture;

[0015] FIG. 3 shows an example system having multi-processor cells in which an embodiment for switching the role of boot processor to a spare processor responsive to detected LOL in the boot processor may be employed;

[0016] FIG. 4 shows an exemplary operational flow diagram for switching the role of boot processor to a hot spare processor according to one embodiment;

[0017] FIG. 5 shows an exemplary operational flow diagram for switching the role of boot processor to an active spare processor according to one embodiment; and

[0018] FIG. 6 shows a more general operational flow diagram for certain embodiments.

DETAILED DESCRIPTION OF THE INVENTION

[0019] As described further herein and in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/973,076 (U.S. Published Patent Appplication 20060107107) titled "SYSTEM AND METHOD FOR PROVIDING FIRMWARE RECOVERABLE LOCKSTEP PROTECTION," the disclosure of which is incorporated herein by reference, certain techniques are provided for recovering from LOL detected for a boot processor in a multi-processor system, in which a spare processor is utilized for such recovery. For instance, upon LOL being detected for the system's boot processor, the spare processor assumes the role of boot processor, and then recovery of lockstep for the original boot processor is attempted. Accordingly, for such lockstep recovery technique, a spare processor is used for attempting to recover from LOL detected for the boot processor. Exemplary techniques are described in concurrently flied and commonly assigned U.S. patent application Ser. No. 10/973,077 (U.S. Published Patent Application 20060107112) titled "SYSTEM AND METHOD FOR ESTABLISHING A SPARE PROCESSOR FOR RECOVERING FROM LOSS OF LOCKSTEP IN A BOOT PROCESSOR," the disclosure of which is incorporated herein by reference, for establishing a spare processor for the boot processor. Such exemplary techniques include techniques for establishing a hot spare processor that is held in reserve until such time as it is used to assume the role of boot processor responsive to a detected LOL in the boot processor. The exemplary techniques further include techniques for establishing an active spare processor that is not held in reserve, but is instead made available to the OS for executing instructions during normal system operation and is dynamically transformed into a hot spare (i.e., it is idled) such that it can assume the role of the boot processor responsive to detection of LOL in the boot processor.

[0020] Embodiments are provided herein for switching the role of boot processor to an established spare processor responsive to detection of LOL in the boot processor. For instance, exemplary actions by system firmware for switching the role of boot processor to an established spare processor and thus causing the OS to recognize the established spare processor as the system's boot processor are provided in accordance with certain embodiments hereof.

[0021] As described further in concurrently filed and commonly assigned U.S. patent application Ser. No. U.S. patent application Ser. No. 10/973,077 (U.S. Published patent application 20060107112) titled "SYSTEM AND METHOD FOR ESTABLISHING A SPARE PROCESSOR FOR RECOVERING FROM LOSS OF LOCKSTEP IN A BOOT PROCESSOR," in certain embodiments, information (referred to herein as a "device tree") is stored to a multi-processor system, wherein such information identifies the processor in the system that is the boot processor and identifies the processor that is the spare for the boot processor. The information is stored such that it is accessible by the system's firmware, and the firmware, upon detecting LOL for the boot processor, determines the spare processor from the device tree and assigns the spare processor the role of boot processor. The firmware then attempts to recover lockstep for the processor for which LOL was detected. If the recovery of lockstep is successful, the processor for which lockstep was recovered (i.e., the former boot processor) may be assigned the role of spare for the current boot processor. Thus, the multi-processor system may recover from LOL detected for the boot processor without requiring a crash of the system (at least for certain types of "recoverable" LOLs).

[0022] In some implementations described herein, the spare processor for the boot processor is a "hot" spare, in which it is idled and not made available to the OS for processing instructions during normal system operation. That is, the hot spare processor is held in reserve, idling until such time as it assumes the role of boot processor responsive to detected LOL for the boot processor. In other implementations, the spare processor is not held in reserve, but is instead an "active" processor that is made available to the OS for executing instructions during normal system operation. Upon LOL being detected for the boot processor, the designated spare processor is dynamically transformed into a hot spare (i.e., it is idled) such that it can assume the role of the boot processor. For instance, upon the system's firmware detecting a LOL for the boot processor, the firmware accesses the device tree and determines the designated spare processor, and then the firmware instructs the OS to idle the designated spare processor (i.e., stop scheduling instructions for the processor) to make the "active" spare a "hot" spare, which then assumes the role of boot processor.

[0023] Concurrently filed and commonly assigned U.S. patent application Ser. No. 10/973,076 (U.S. Published Patent Application 20060107107) titled "SYSTEM AND METHOD FOR PROVIDING FIRMWARE RECOVERABLE LOCKSTEP PROTECTION," discloses a technique in which lockstep can be recovered for processors (particularly, non-boot processors) without requiring use of a spare processor. Rather, upon LOL being detected for a lockstep pair of processors, the firmware instructs the OS to idle (or "eject") the lockstep pair of processors for which LOL was detected and then the firmware attempts to reestablish lockstep (e.g., by resetting the processors). If successful in reestablishing lockstep, the firmware triggers the OS to recognize the lockstep pair of processors as again being available for servicing instructions.

[0024] However, for various reasons, in certain system architectures problems arise in attempting to idle (or eject) the boot processor from the system. Thus, in certain embodiments, a spare processor is used for recovering from LOL for the system's boot processor. Again, embodiments are provided herein for switching the role of boot processor to an established spare processor responsive to detection of LOL for the boot processor. In certain embodiments, the spare processor is used for recovering from LOL only in the case in which LOL is detected for the boot processor. For instances in which LOL is detected for non-boot processors, the lockstep may be recovered without the use of a spare by using the recovery technique described in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/973,076 (U.S. Published Patent Application 20060107107 titled "SYSTEM AND METHOD FOR PROVIDING FIRMWARE RECOVERABLE LOCKSTEP PROTECTION," for example.

[0025] Turning to FIG. 1, an example embodiment of a system **10** in which a spare processor is used in recovering from detected loss of lockstep (LOL) for a system boot processor is shown. System **10** includes OS **11**, as well as master processor **12**A and slave processor **12**B (collectively referred to as a lockstep processor pair **12**). In certain implementations the lockstep processor pair **12** may be implemented on a single silicon chip, which is referred to as a "dual core processor" in which master processor **12**A is a first core and slave processor **12**B is a second core. Further, lockstep processor pair **12** may be referred to as a processor or CPU "module" because it includes a plurality of processors (**12**A and **12**B) in such module. As described further herein, to certain portions of system **10**, such as its OS **11**, the

processor module **12** may appear as a single processor because slave processor **12**B is merely used for redundant processing for checking the output of master processor **12**A. In this example, lockstep processor pair **12** is designated as the boot processor for system **10**. As described further herein, in certain embodiments lockstep processor pair **12** is designated as the system boot processor by storing information to the system that specifies such designation.

[0026] Master processor **12**A includes cache **14**A, and slave processor **12**B includes cache **14**B. OS **11** and lockstep processor pair **12** are communicatively coupled to bus **16**. Typically, master processor **12**A and slave processor **12**B are coupled to bus **16** via an interface that allows each of such processors to receive the same instructions to process, but such interface only communicates the output of master processor **12**A back onto bus **16**. The output of slave processor **12**B is used solely for checking the output of mater processor **12**A.

[0027] In this example, master processor **12**A includes error detect logic **13**A, and slave processor **12**B includes error detect logic **13**B. While shown as included in each of the processors **12**A and **12**B in this example, in certain embodiments the error detect logic **13**A and **13**B may be implemented external to processors **12**A and **12**B. Error detect logic **13**A and **13**B include logic for detecting errors, such as data cache errors, present in their respective processors **12**A and **12**B. Examples of error detect logic **13**A and **13**B include known parity-based mechanisms and ECC mechanisms. Error detect logic **13**C is also included, which may include an XOR (exclusive OR) gate, for detecting a lockstep mismatch between master processor **12**A and slave processor **12**B. As mentioned above, a lockstep mismatch refers to the output of master processor **12**A and slave processor **12**B failing to match. While shown as external to the lockstep processor pair **12** in this example, in certain embodiments error detect logic **13**C may be implemented on a common silicon chip with processors **12**A and **12**B.

[0028] Lockstep mismatch is one way of detecting a LOL between the master processor **12**A and slave processor **12**B. A detection of an error by either of error detect logic **13**A and **13**B also provides detection of LOL in the processors **12**A and **12**B. Because the detection of LOL by error detect logic **13**A and **13**B may occur before an actual lockstep mismatch occurs, the detection of LOL by error detect logic **13**A and **13**B may be referred to as a detection of a "precursor to lockstep mismatch". In other words, once an error (e.g., corrupt data) is detected by error detect logic **13**A or **13**B, such error may eventually propagate to a lockstep mismatch error that is detectable by error detect logic **13**C.

[0029] System **10** further includes a second lockstep processor pair **15** that is communicatively coupled to bus **16**. Lockstep processor pair **15** includes master processor **15**A and slave processor **15**B. Master processor **15**A includes cache **17**A, and slave processor **15**B includes cache **17**B. As with error detect logic **13**A-**13**C implemented for lockstep processor pair **12**, lockstep processor pair **15** has error detect logic **16**A-**16**C. While only two lockstep processor pairs **12** and **15** are shown for simplicity in the example of FIG. 1, system **10** may include any number of such lockstep processor pairs. As one specific example, in one implementation system **10** includes **64** lockstep processor pairs, wherein the master processors of the pairs perform parallel processing for the system. In the specific example of FIG. 1, lockstep processor pair **15** is designated as a spare for boot processor **12**, as described further herein.

[0030] In certain embodiments described herein, techniques are provided for switching the role of boot processor to the designated spare processor responsive to detection of LOL for the boot processor. Thus, the spare processor is used for recovery when LOL is detected for the system's boot processor. That is, the designated spare processor assumes the role of boot processor in the event that LOL is detected for the boot processor. In accordance with certain embodiments, upon LOL being detected for a boot processor, a hot spare processor (i.e., an idling processor that is available in the system) is transferred the role of boot processor, and then the old boot processor having LOL is reset to reestablish its lockstep. In other embodiments, a processor is designated as a spare processor for the boot processor, but such designated spare processor is not held in reserve (idled) during normal system operation. Rather, the designated spare is utilized as an active processing resource made available to OS **11** for processing instructions that OS **11** schedules for such "active" spare processor. Upon LOL being detected for the boot processor, the designated spare processor is idled (thereby, transforming such active processor into a hot spare), and is transferred the role of boot processor. Thereafter, the former boot processor for which the LOL was detected may be reset to reestablish its lockstep. Thus, in certain embodiments, a designated spare processor is not required to be held in reserve, but is instead an available resource for use within the system and is only transformed into an actual spare processor when needed for recovering from LOL for the boot processor.

[0031] Firmware **18** is also included in system **10**, which in this embodiment is invoked upon an error being detected by any of the error detect logics **13**A-**13**C and **16**A-**16**C. In certain embodiments, processors **12**A, **12**B, **15**A, and **15**B are processors from the Itanium Processor Family (IPF). IPF is a 64-bit processor architecture co-developed by Hewlett-Packard Company and Intel Corporation, which is based on Explicitly Parallel Instruction Computing (EPIC). IPF is a well-known family of processors. IPF includes processors such as those having the code names of MERCED, MCKINLEY, and MADISON. In addition to supporting a 64-bit processor bus and a set of 128 registers, the 64-bit design of IPF allows access to a very large memory (VLM) and exploits features in EPIC. While a specific example implementation of one embodiment is described below for the IPF architecture, embodiments described herein for establishing a spare processor for recovering from LOL detected for a boot processor are not limited in application to an IPF architecture, but may be applied as well to other architectures (e.g., 32-bit processor architectures, etc.).

[0032] Processor architecture generally comprises corresponding supporting firmware, such as firmware **18** of system **10**. For example, as described further below in conjunction with the specific example of FIG. 2, the IPF processor architecture comprises such supporting firmware as Processor Abstraction Layer (PAL), System Abstraction Layer (SAL), and Extended Firmware Interface (EFI). Such supporting firmware may enable, for example, the OS to access a particular function implemented for the processor. For instance, the OS may query the PAL as to the size of the cache implemented for the processor, etc. Other well-known functions provided by the supporting firmware (SAL, EFI) include, for example: (a) performing I/O configuration accesses to discover and program the I/O Hardware (SAL_PCI_CONFIG_READ and SAL_PCI_CONFIG_WRITE); (b) retrieving error log data from the platform following a Machine Check Abort (MCA) event (SAL_GET_STATE_INFO); (c) accessing persistent store configuration data stored in non-volatile memory (EFI variable services: GetNextVariableName, GetVariable and SetVariable); and accessing the battery-backed real-time clock/calendar (EFI GetTime and SetTime). Accordingly, the supporting firmware, such as the PAL, is implemented to provide an interface to the processor(s) for accessing the functionality provided by such processor(s). Each of those interfaces provide standard, published procedure calls that are supported.

[0033] In the example system **10** of FIG. 1, firmware **18** utilizes the above-mentioned hot spare technique for recovering from LOL for a boot processor. That is, processor **15** is designated as a hot spare for boot processor **12**, and such processor **15** is held as an idle, spare processor and used in the manner described further herein for assuming the role of boot processor in the event that a LOL is detected for boot processor **12**. Of course, as described further herein, the above-mentioned technique in which a designated spare is not held in reserve but is instead dynamically transformed into a spare processor may be implemented by firmware **18** in alternative embodiments. As described further herein, in certain embodiments, a device tree is stored to a data storage device (e.g., SRAM) such that it is accessible to the firmware **18**, wherein the device tree includes information identifying processor **12** as the system's boot processor and identifying processor **15** as the spare. The information specifying the boot processor and/or spare may be received as input from a user in certain embodiments. For instance, in certain embodiments a user interface is provided that enables a system administrator to designate the boot processor and/or spare processor in the multi-processor system. In other embodiments, the firmware **18** selects the boot processor and/or spare.

[0034] Suppose that during the system boot-up procedure, master processor **12**A assumes the role of boot processor, as shown in the example of FIG. 1. In this example embodiment, lockstep processor pair **15** is held as a hot spare for recovering from a LOL that may be detected for the boot processor **12**A. Of course, additional lockstep processor pairs may be included in system **10** (not specifically shown in the example of FIG. 1), and those additional lockstep processor pairs, which are not the system boot processor, may recover from LOL in a different manner that does not utilize a hot spare processor. Thus, in accordance with certain embodiments hereof, hot spare processor pair **15** is not needed for recovering from LOL detected for any non-boot processor, but may instead be used for recovery of LOL only if such LOL is detected for the boot processor (processor **12**A in this example).

[0035] Firmware **18** is included in this example, and upon detection of LOL by any of error detect logics **13**A-**13**C, firmware **18** determines, in operational block **101**, whether the detected LOL is a recoverable LOL. That is, firmware **18** determines in block **101** whether the detected LOL is of a type from which the firmware can recover lockstep for the lockstep processor pair **12** without crashing the system. As described further herein, lockstep is recoverable for certain detected LOLs (which may be referred to as "recoverable LOLs"), while lockstep is not recoverable for other detected LOLs (which may be referred to as "non-recoverable LOLs"). If the lockstep is not recoverable from the detected LOL, then in the example of FIG. 1 firmware **18** crashes the system in block **102**.

[0036] If the lockstep is recoverable, then operation advances to block **103** whereat the firmware determines whether the processor for which the LOL was detected is the system's boot processor. Because the LOL is detected for the lockstep processor pair **12** in this example, firmware **18** determines whether master processor **12**A is the system's boot processor. This can be determined, for example, by accessing the device tree **203** of FIG. 2 (described further below). The device tree may, in certain implementations, include a field for each processor indicating whether such processor is the system's boot processor. This field in the device tree may be set

by the firmware during the boot-up process to identify the corresponding processor that is used as the system's boot processor or it may be pre-set by a system administrator (e.g., via a user interface), as examples. If determined in block **103** that the master processor **12**A is not the system's boot processor, then operation advances in this example to block **104**, and a lockstep recovery process is performed that does not utilize hot spare **15**.

[0037] In the specific example shown in FIG. 1, if the lockstep is determined to be recoverable in block **101** and the processor for which the LOL is detected is determined in block **103** to not be the system boot processor, firmware **18** cooperates with OS **11** via standard OS methods to recover the lockstep. For instance, in the example embodiment of FIG. 1, Advanced Configuration and Power Interface (ACPI) methods are used by firmware **18** to cooperate with OS **11**. Accordingly, no processor or platform specific knowledge is required to be embedded in OS **11**, but instead any ACPI-compatible OS may be used, including without limitation HP-UX and Open VMS operating systems. Example techniques that may be used for recovering from LOL for non-boot processors of a system are described further in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/973,076 titled "SYSTEM AND METHOD FOR PROVIDING FIRMWARE RECOVERABLE LOCKSTEP PROTECTION," the disclosure of which is incorporated herein by reference.

[0038] In the example embodiment of FIG. 1, if determined in block **103** that the processor for which the LOL is detected is not the system boot processor, firmware **18** triggers OS **11** to idle the master processor **12**A in operational block **104**. In this embodiment, firmware **18** utilizes an ACPI method **105** to "eject" master processor **12**A, thereby triggering OS **11** to idle the master processor **12**A (i.e., stop scheduling tasks for the processor). Of course, by idling master processor **12**A, slave processor **12**B will in turn be idled. Thus, idling master processor **12**A results in idling the lockstep processor pair **12**. In this example embodiment, OS **11** is not aware of the presence of slave processor **12**B, but is instead aware of master processor **12**A. The interface of lockstep processor pair **12** to bus **16** manages copying to slave processor **12**B the instructions that are directed by OS **11** to master processor **12**A. Thus, firmware **18** need not direct OS **11** to eject slave processor **12**B, as OS **11** is not aware of such slave processor **12**B in this example implementation. Again, by idling master processor **12**A, slave processor **12**B is also idled as it merely receives copies of the instructions directed to master processor **12**A. Of course, if in a given implementation OS **11** is aware of slave **12**B, firmware **18** may be implemented to also direct OS **11** to idle such slave processor **12**B in a manner similar to that described for idling master processor **12**A.

[0039] Firmware **18** then attempts to recover lockstep for the lockstep processor pair **12** in block **106**. For instance, firmware **18** resets the processor pair **12**. During such reset of processor pair **12**, system **10** can continue to operate on its remaining available processors. Once the processor pair **12** is reset and lockstep is recovered, firmware **18** reintroduces master processor **12**A to OS **11** in operational block **107**. In this embodiment, firmware **18** updates the ACPI device table information for master processor **12**A to indicate that such master processor **12**A is "present, functioning and enabled." As discussed in the ACPI 2.0 specification for the _STA status method of a device, the _STA (status) object returns the status of a device, which can be one of the following: enabled, disabled, or removed. In this respect, in the result code (bitmap) bit **0** is set if the device is present; bit **1** is set if the device is enabled and decoding its resources; bit **2** is set if the device should be shown in the UI; bit **3** is set if the device is functioning properly (cleared if the device failed its diagnostics); bit **4** is set if the battery is present; and bits **5-31** are reserved. A device can only decode its hardware resources if both bits **0** and **1** are set. If the device is not present (bit **0** cleared) or not enabled (bit **1** cleared), then the device must not decode its resources. Bits **0**, **1** and **3** are the "present, enabled and functioning" bits mentioned above. Firmware **18** utilizes an ACPI method **108** to trigger OS **11** to "check for" master processor **12**A, thereby reintroducing the master processor **12**A to OS **11**. As a result of checking for master processor **12**A, OS **11** will recognize that such master processor **12**A is again available and will thus begin scheduling tasks for master processor **12**A once again.

[0040] In view of the above, a recovery technique is provided that does not require utilizing a spare processor for recovering from LOL detected for non-boot processors. For various reasons, in certain system architectures problems arise in attempting to idle (or eject) the boot processor from the system. Thus, in certain embodiments, a hot spare processor is used for recovering from LOL for the system's boot processor. For instance, in the example of FIG. 1, if determined in block **103** that the master processor **12**A is the system's boot processor, operation advances to block **109** whereat the state of the "good" processor in the lockstep processor pair **12** is copied over to each of processors **15**A and **15**B in the spare lockstep processor pair **15**. Exemplary techniques that may be utilized by firmware **18** for switching the role of boot processor to the hot spare are described further herein below.

[0041] Thus, operational block **109** essentially makes the spare processor pair **15** the system's boot processor, and then in certain embodiments firmware **18** resets the lockstep processor pair **12** and reestablishes its lockstep in block **110**. Once lockstep is reestablished for lockstep processor pair **12**, that pair is held in operational block **111** as a hot spare for the new boot processor pair **15**. That is, firmware **18** updates the device tree **203** of FIG. 2 to reflect that lockstep processor pair **12** is a hot spare for the boot processor pair **15**. Thus, should a recoverable LOL be detected for the new boot processor **15**, the above process may be used by firmware **18** to make the spare lockstep processor pair **12** the boot processor and then recover lockstep for pair **15**. An example technique for resetting the lockstep processor pair **12** and reestablishing its lockstep in which the lockstep processor pair **12** is then held as a hot spare is described further in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/972,796 (U.S. Published Patent Application 20060107106) titled "SYSTEM AND METHOD FOR MAINTAINING IN A MULTI-PROCESSOR SYSTEM A SPARE PROCESSOR THAT IS IN LOCKSTEP FOR USE IN RECOVERING FROM LOSS OF LOCKSTEP FOR ANOTHER PROCESSOR", the disclosure of which is hereby incorporated herein by reference.

[0042] FIG. 2 shows a block diagram of one embodiment of the above system **10**, which is implemented for the IPF processor architecture and is labeled as system $\mathbf{10}_A$. The quintessential model of the traditional IPF architecture is given in the *Intel IA-64 Architecture Software Developer's Manual, Volume 2: IA-64 System Architecture*, in section 11.1 *Firmware Model*, the disclosure of which is hereby incorporated herein by reference. Accordingly, in this example embodiment of system $\mathbf{10}_A$, firmware **18**, labeled as firmware $\mathbf{18}_A$, includes processor abstraction layer (PAL) **201** and platform/system abstraction layer (SAL) **202**. In general, PAL **201** is firmware provided by Intel for its processors, and SAL **202** is developed by an original equipment manufacturer (OEM) for the specific system/platform in which the processors are to be employed. PAL **201**, SAL **202**, as well as an extended firmware interface (EFI) layer (not shown), together provide, among other things, the processor and system initialization for an OS boot in an IPF system.

[0043] The boot-up process of a traditional IPF system, for example, proceeds as follows: When the system is first powered on, there are some sanity checks (e.g., power on self-test) that are performed by microprocessors included in the system platform, which are not the main system processors that run applications. After those checks have passed, power and clocks are given to a boot processor (which may, for example, be master processor **12**A). The boot processor begins executing code out of the system's Read-Only Memory (ROM) (not specifically shown in FIG. 2). The code that executes is the PAL **201**, which gets control of system **10**. PAL **201** executes to acquire all of the processors in system $\mathbf{10}_A$ (recall that there may be many lockstep processor pairs, such as pairs **12** and **15**) such that the processors begin executing concurrently through the same firmware.

[0044] After it has performed its duty of initializing the processor(s), PAL **201** passes control of system $\mathbf{10}_A$ to SAL **202**. It is the responsibility of SAL **202** to discover what hardware is present on the system platform, and initialize it to make it available for the OS **11**. When main memory is initialized and functional, the firmware $\mathbf{18}_A$ is copied into the main memory. Then, control is passed to EFI (not shown), which is responsible for activating boot devices, which typically includes the disk. The EFI reads the disk to load a program into memory, typically referred to as an operating system loader. The EFI loads the OS loader into memory, and then passes it control of system $\mathbf{10}_A$ by branching the boot processor into the entry point of such OS loader program.

[0045] The OS loader program then uses the standard firmware interfaces to discover and initialize system $\mathbf{10}_A$ further for control. One of the things that the OS loader typically has to do in a multi-processor system is to retrieve control of the other processors (those processors other than the boot processor). For instance, at this point in a multi-processor system, the other processors may be executing in do-nothing loops. In an ACPI-compatible system, OS **11** makes ACPI calls to parse the ACPI tables to discover the other processors of a multi-processor system in a manner as is well-known in the art. Then OS **11** uses the firmware interfaces to cause those discovered processors to branch into the operating system code. At that point, OS **11** controls all of the processors and the firmware $\mathbf{18}_A$ is no longer in control of system $\mathbf{10}_A$.

[0046] As OS **11** is initializing, it has to discover from the firmware $\mathbf{18}_A$ what hardware is present at boot time. And in the ACPI standards, it also discovers what hardware is present or added or removed at run-time. Further, the supporting firmware (PAL, SAL, and EFI) are also used during system runtime to support the processor. For example, OS **11** may access a particular function of master processor **12**A via the supporting firmware $\mathbf{18}_A$, such as querying PAL **201** for the number, size, etc., of the processor's cache **14**A. Some other well-known firmware functions that OS **11** may employ during runtime include: (a) PAL **201** may be invoked to configure or change processor features such as disabling transaction queuing (PAL_BUS_SET_FEATURES); (b) PAL **201** may be invoked to flush processor caches (PAL_CACHE_FLUSH); (c) SAL **202** may be invoked to retrieve error logs following a system error (SAL_GET_STATE_INFO, SAL_CLEAR_STATE_INFO); (d) SAL **202** may be invoked as part of hot-plug sequences in which new I/O cards are installed into the hardware (SAL_PCI_CONFIG_READ, SAL_PCI_CONFIG_WRIT); (e) EFI may be invoked to change the boot device path for the next time the system reboots (SetVariable); (f) EFI may be invoked to change the clock/calendar hardware settings; and (g) EFI may be invoked to shutdown the system (ResetSystem).

[0047] A "device tree" is provided, which is shown as device tree **203** in this example. Device tree **203** is stored in SRAM (Scratch RAM) on the cell, which is RAM that is reinitialized. Firmware **18**A builds the device tree **203** as it discovers what hardware is installed in the system. Firmware then converts this information to the ACPI tables format and presents it to OS **11** so that OS **11** can know what is installed in the system. The ACPI device tables (not shown) are only consumed by OS **11** at boot time, so they are never updated as things change. For OS **11** to find the current status, it calls an ACPI "method" to discover the "current status". The _STA method described above is an example of such an ACPI method. When _STA is called, the AML can look for properties on the device specified in the firmware device tree and convert that into the Result Code bitmap described above. So, if lockstep has been lost on a processor, firmware **15**A will set the device tree property that indicates loss of lockstep, then when OS **11** calls _STA for that device, the "lockstep lost" property directs the AML code to return to "0" in the "functioning properly" bit so that OS **11** can know there is a problem with that processor.

[0048] If a lost lockstep is recovered in accordance with the recovery technique described herein, firmware **18**A can indicate that lockstep has been recovered in the device tree **203**. Then when _STA is called on that device responsive to the OS receiving the "check for device" ACPI method, the present, enabled and functioning bits will all be set and OS **11** will know the CPU is safe to use.

[0049] Further, device tree **203** includes information designating which processor is the boot processor, as well as a spare processor for the boot processor. A simple example of device tree **203** is shown below in Table 1:

[0050]

TABLE 1

Device	Status	Lockstep Enabled	Boot Processor?	Spare for Boot Processor?
Processor A	Present, Enabled, and Functioning	Yes	Yes	No
Processor B	N/A	Yes	No	Yes
Processor C	Present, Enabled, and Functioning	Yes	No	No
. . .	. . .	. . .	. . .	. . .

[0051] In the above example of Table 1, 3 processors "Processor A," "Processor B," and "Processor C" are identified in the device tree as being included in the multi-processor system. The status of each of processors Processor A and Processor B is identified as "Present, Enabled, and Functioning." In this example Processor B is a hot spare, so it is held in reserve and does not show up in the static ACPI tables that are built when the system boots. Accordingly the OS does not know about Processor B and does not have a device handle to call the _STA method on. As described above, the _STA method is a known ACPI method for checking the status of the corresponding device. Therefore, the status of Processor B is designated N/A (not applicable) in this example because the _STA method will not be called for this processor as long as it is the hot spare. Of course, upon such spare assuming the role of boot processor, its status is updated to "Present, Enabled, and Functioning." In other embodiments in which Processor B is an active spare, the OS will be aware of such processor and accordingly its status may be also indicated as "Present, Enabled, and Functioning."

[0052] Further, lockstep is identified in this example as enabled for these processors. In certain embodiments, the system administrator can designate, via user interface **20**, whether lockstep mode is to be enabled for the system's processors. Further, Processor A is designated as the system's boot processor in the example device tree shown in Table 1, and Processor B is designated as the spare processor. Thus, upon detection of LOL for one of the processors in the system, the firmware can access this device tree to determine if the processor for which LOL is detected is the system's boot processor; and if determined that the processor for which LOL is detected is the system's boot processor, the firmware can further determine from this device tree the designated spare processor to use in the lockstep recovery process.

[0053] In the example of FIG. 2, a user interface **20** is included, which allows a user (e.g., system administrator) to designate the system's boot processor and spare processor. In certain embodiments, the user interface **20** further allows the user to designate other information, such as whether lockstep mode is enabled for the processors.

[0054] One example system in which firmware $\mathbf{18}_A$ of FIG. 2 may be employed is shown in FIG. 3. FIG. 3 shows system **30** that includes multi-processor cells A-D, labeled $\mathbf{31}_A$-$\mathbf{31}_D$, respectively. Cell A $\mathbf{31}_A$ is shown in more detail, and should be understood that cells B-D $\mathbf{31}_B$-$\mathbf{31}_D$ have substantially the same architecture as that of cell A $\mathbf{31}_A$ in this example. The architecture of cell A $\mathbf{31}_A$ described hereafter corresponds to that of Hewlett-Packard's Superdome™ systems. As shown, multi-processor cell A $\mathbf{31}_A$ includes coherency controller **304** that is communicatively coupled to two buses, labeled **305** and **306**, respectively. A plurality of processor modules are included within cell A $\mathbf{31}_A$, shown as CPU **0**, CPU **1**, CPU **2** and CPU **3** (and labeled **300**-**303**, respectively). More particularly, a plurality of processors modules are communicatively coupled to each of the buses **305** and **306**. In this example, CPU **0** (**300**) and CPU **1** (**301**) are each communicatively coupled to the first bus **305**, and CPU **2** (**302**) and CPU **3** (**303**) are each communicatively coupled to the second bus **306**. It should be understood that each of CPU **0** (**300**), CPU **1** (**301**), CPU **2** (**302**), and CPU **3** (**303**) is in actuality a lockstep processor pair, such as the lockstep processor pair **12** of FIGS. 1-3. Thus, for instance, CPU **0** (**300**) includes both a master processor and a slave processor, as with processors **12**A and **12**B included in the lockstep processor pair **12** of FIGS. 1-2.

[0055] Coherency controller **304** decodes the address an interrupt is targeted towards and determines which bus (**305** or **306**) the interrupt packet should be delivered to. Each CPU on the destination bus sees the interrupt and compares the interrupt target address with its internal "LID" register to determine if it is the targeted CPU. If the match is correct, the CPU responds to the interrupt. If the match is not correct, the CPU ignores the interrupt packet.

[0056] As shown, the example firmware $\mathbf{18}_A$ described above with FIGS. 1-2 may be employed to manage the recovery from detected LOLs for the processors of cells A-D ($\mathbf{31}_A$-$\mathbf{31}_D$). In this example architecture, when designating a spare for the boot processor, such spare is selected to be another processor communicatively coupled to the same bus as the boot processor. For instance, suppose that during the boot-up of system **30**, CPU **2** (**302**) is designated as the boot processor. Accordingly, firmware $\mathbf{18}^A$ will identify another processor that is communicatively coupled to bus **306** to designate as a spare for such boot processor, such as CPU **3** (**303**). During runtime if a LOL is detected for CPU **2** (**302**), then firmware $\mathbf{18}_A$ may operate according to the process described above in FIG. 1 to switch the designation of boot processor to a designated spare, CPU **3** (**303**). In performing this switch the LID register of the CPU that lost lockstep is copied into the LID register of the spare CPU and the spare CPU pair will then assume the ID of the failed CPU and start responding to interrupts directed to that ID. The LID of the failed CPU is quickly re-written to a different value so that it will no longer respond to interrupts directed to that ID. CPU **2** (**302**) can then be reset to recover its lockstep, and such CPU **2** (**302**) may then be designated as a spare for the new boot processor CPU **3** (**303**).

[0057] It should be recognized that in the example system of FIG. 3, if a hot spare were used for recovering from LOL for all processors, rather than just for the boot processor, much of the potential processing resources would be required to be held in reserve as spare resources. For instance, a hot spare would be required on each of the buses of each cell. Because there are two processors coupled to each bus in this example (i.e., CPU **0** and CPU **1** coupled to bus **305**, and CPU **2** and CPU **3** coupled to bus **306**), reserving a hot spare processor for each bus would result in holding half of the total processing resources in reserve. This is an undesirably expensive and wasteful solution. Thus, according to the illustrated embodiment, a hot spare processor is reserved only for the boot processor. Thus, continuing with the above example in which CPU **2** is the boot processor and CPU **3** is held as a "hot" spare, both of CPU **0** and CPU **1** are active (i.e., neither are held in spare). Further, none of the processors of cells B-D need to be held in reserve as hot spares. In the event of a recoverable LOL occurring for any of non-boot processors of the cells, the lockstep recovery process of FIG. 1 (i.e., operational blocks **104**-**108**) can be utilized without requiring a spare for lockstep recovery for those non-boot processors.

[0058] While in the above example embodiment, a hot spare CPU is maintained for recovery from LOL for the system's boot processor, in an alternative embodiment a hot spare need not be held as such, but rather a processor can be dynamically made a "spare" when needed for recovery from LOL encountered for the system boot processor. That is, assuming as in the above example of FIG. 3 that CPU **2** is the system's boot processor, CPU **3** does not have to be held in reserve (idled) as a hot spare for CPU **2**, but instead CPU **3** can be active and dynamically idled and made a "spare" upon detection of LOL in CPU **2**. Accordingly, CPU **3** may be

designated as a spare for the boot processor, but such CPU **3** need not be held in reserve during normal system operation. Instead, the CPU **3** may be dynamically (during system run-time) transformed into a spare processor (e.g., by causing the OS to idle such processor) in the event that a LOL is detected for the boot processor.

[0059] Thus, the resource of this spare CPU need not be wasted during normal runtime, but can be dynamically turned into a spare processor that is available for recovering from LOL encountered on CPU **2** in the manner described above. As an example of this embodiment, the firmware may send an "eject request" on a healthy processor pair (e.g., CPU **3**), and lie to the OS by indicating (e.g., in response to an ACPI _STA method for the healthy processor pair) that such processor pair is not functioning. This would cause the OS to eject that processor pair as though it had encountered a LOL. This processor would become the "hot spare" and be used to replace the boot processor. The system boot processor role would be assumed by the idled healthy processor pair, just as described above in the example in which the healthy CPU **3** is held idle for the system boot processor, CPU **2**. The only difference being that now the firmware had to take some initial action to turn the healthy processor pair into a "hot spare" by idling it and making it appear to the OS as though it was not functioning. Thus, the LIDs would be swapped. That is, the LID register of the boot processor that lost lockstep is copied into the LID register of the now spare processor pair (CPU **3** in this example), and the LID register of this spare processor pair would be copied into the LID register of the original boot processor. Lockstep is then reestablished on the original boot processor, and it can be reintroduced to the OS with the ID of the original non-boot processor that replaced it as the system boot processor. The processor that was formerly the boot processor may be designated, in device tree **203**, as the spare for the now boot processor.

[0060] Turning to FIG. 4, an exemplary operational flow diagram for switching the role of boot processor to a hot spare processor according to one embodiment is shown. In this case, an LOL has been detected for the boot processor. In operational block **401**, the system firmware allocates space to save the status of the good processor of the lockstep pair of processors for which LOL was detected. Exemplary techniques for determining which of the lockstep pair of processors had the error are disclosed in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/972,835 (U.S. Published Patent Application 20060107114) titled "SYSTEM AND METHOD FOR USING INFORMATION RELATING TO A DETECTED LOSS OF LOCKSTEP FOR DETERMINING A RESPONSIVE ACTION," the disclosure of which is hereby incorporated herein by reference. Accordingly, an appropriate amount of data storage space for storing the cache of the "good" processor in the lockstep pair of processors is allocated in block **401**. For instance, an amount of SRAM may be allocated for storing the registers from the good processor.

[0061] In operational block **402**, the state of the good processor is stored to the allocated storage space. For instance, in this illustrative example, PAL is called to dump the processor state of the good processor of the current boot processor (CBP) module (for which LOL was detected) to the allocated storage space.

[0062] In operational block **403**, the LID of the current boot processor (for which LOL was detected) is reprogrammed to a different value. For instance, the current boot processor may be assigned a LID that is reserved just for this situation (i.e., for switching the role of boot processor). Accordingly, the boot processor for which LOL was detected will not respond to the interrupts addressed for its original LID because the boot processor's LID value is reprogrammed to a different value.

[0063] In operational block **404**, the state of the good processor that was stored in block **402** to the allocated storage space is stored to the hot spare processor. In this example, the PAL is called to "Restore Processor State" on the hot spare processor, wherein the PAL is provided a pointer to the address to which the good processor's state was stored in block **402**. Accordingly, all the values from that storage space will be read into the registers of the spare processor. As a result, the LID value of the spare processor is programmed with the value of the boot processor that was saved to the allocated storage space. Accordingly, the spare processor automatically, as a result of receiving this stored state, receives the Local ID corresponding to the boot processor for interrupt handling.

[0064] In operational block **405**, the system firmware updates its device tree information to reflect that the hot spare is assuming the role of boot processor. The system firmware further updates its device tree, in this example, to indicate that the former boot processor (for which LOL was detected) is assuming the role of hot spare for the new boot processor.

[0065] In operational block **406**, PAL_MC_RESUME (which is a standard PAL call for resuming normal execution after a machine check (MC)) is called for the hot spare (which is assuming the role of boot processor) to cause the OS to recognize the hot spare processor as available for processing instructions, wherein the OS recognizes the hot spare as the boot processor. Thus, the hot spare has now assumed the role of the boot processor, taking the place of the former boot processor for which LOL was detected. Accordingly, in operational block **407** the former hot spare (which has assumed the role of boot processor) resumes normal operation as the system's boot processor with its lockstep protection enabled. Thus, in this exemplary embodiment, upon detection of LOL for a boot processor, the role of boot processor is switched to a hot spare that has lockstep protection, wherein the system effectively recovers lockstep for its boot processor without requiring a system shut down (i.e., without shutting down the OS). In certain implementations, the firmware may go on to take action to attempt to reestablish lockstep for the former boot processor (e.g., by resetting the former boot processor), and if successful in reestablishing its lockstep, such former boot processor may be held in reserve as a hot spare processor for the new boot processor in the event that LOL is detected for this new boot processor.

[0066] Turning to FIG. 5, an exemplary operational flow diagram for switching the role of boot processor to an active spare processor according to one embodiment is shown. Again, an LOL has been detected for the boot processor. In operational block **501**, the system firmware, responsive to such detection of LOL for the boot processor, examines the device tree to determine which processor module is assigned the role of active spare. In operational block **502** the system firmware (e.g., SAL) asserts a General Purpose Event Interrupt (GPE) by writing to the appropriate GPE register in the ACPI register space. This causes the OS to execute its interrupt handler, which will run. an AML Method associated with this particular interrupt. In this example, this AML method that is executed responsive to the interrupt generated by the GPE includes an AML instruction called "Notify". This "Notify" operation indicates the determined active spare processor object and that an Eject is being requested as arguments. Accordingly, an AML method is used to request the OS to eject the active spare that was determined in block **501**. In response, the OS idles the active spare processor and returns control over the active spare processor to the system firmware, in block **503**.

[0067] In operational block **504**, the system firmware allocates space to save the status of the good processor of the lockstep pair of processors for which LOL was detected (i.e., the lockstep pair of processors of the CBP). Exemplary techniques for determining which of the lockstep pair of processors had the error are disclosed in concurrently filed and commonly assigned U.S. patent application Ser. No. 10/972,835 (U.S. Published Patent Application No. 20060107114) titled "SYSTEM AND METHOD FOR USING INFORMATION RELATING TO A DETECTED LOSS OF LOCKSTEP FOR DETERMINING A RESPONSIVE ACTION," the disclosure of which is hereby incorporated herein by reference. Accordingly, an appropriate amount of data storage space for storing the cache of the "good" processor in the lockstep pair of processors is allocated in block **504**. For instance, an amount of SRAM may be allocated for storing the registers from the good processor.

[0068] In operational block **505**, the state of the good processor is stored to the allocated storage space. For instance, in this illustrative example, PAL is called to dump the processor state of the good processor of the current boot processor (CBP) module (for which LOL was detected) to the allocated storage space.

[0069] In operational block **506**, the LID of the current boot processor (for which LOL was detected) is reprogrammed to a different value. For instance, the current boot processor may be assigned a LID that is reserved just for this situation (i.e., for switching the role of boot processor). Accordingly, the boot processor for which LOL was detected will not respond to the interrupts addressed for its original LID because the boot processor's LID value is reprogrammed to a different value.

[0070] In operational block **507**, the state of the good processor that was stored in block **505** to the allocated storage space is stored to the idled spare processor. In this example, the PAL is called to "Restore Processor State" on the active spare processor, wherein the PAL is provided a pointer to the address to which the good processor's state was stored in block **505**. Accordingly, all the values from that storage space will be read into the registers of the spare processor. As a result, the LID value of the spare processor is programmed with the value of the boot processor that was saved to the allocated storage space. Accordingly, the spare processor automatically, as a result of receiving this stored state, receives the Local ID (LID) corresponding to the boot processor for interrupt handling.

[0071] In operational block **508**, the system firmware updates its device tree information to reflect that the active spare is assuming the role of boot processor. The system firmware further updates its device tree, in this example, to indicate that the former boot processor (for which LOL was detected) is assuming the role of active spare for the new boot processor.

[0072] In operational block **509**, PAL_MC_RESUME (which is a standard PAL call for resuming normal execution after a machine check (MC)) is called for the active spare (which is assuming the role of boot processor) to cause the OS to recognize the active spare processor as available for processing instructions, wherein the OS recognizes the active spare as the boot processor. Thus, the active spare has now assumed the role of the boot processor, taking the place of the former boot processor for which LOL was detected. Accordingly, in operational block **510** the former active spare (which has assumed the role of boot processor) resumes normal operation as the system's boot processor with its lockstep protection enabled. Thus, in this exemplary embodiment, upon detection of LOL for a boot processor, an active spare processor is idled, and the role of boot processor is switched to such idled spare that has lockstep protection, wherein the system effectively recovers

lockstep for its boot processor without requiring a system shut down (i.e., without shutting down the OS). In certain implementations, the firmware may go on to take action to attempt to reestablish lockstep for the former boot processor (e.g., by resetting the former boot processor), and if successful in reestablishing its lockstep, such former boot processor may be reintroduced to the OS as an active spare processor for the new boot processor in the event that LOL is detected for this new boot processor.

[0073] The exemplary procedures of FIGS. 4 and 5 switch the role of boot processor to a spare processor in accordance with certain embodiments. While exemplary operational flows are provided in FIGS. 4 and 5, embodiments hereof are not limited to those illustrative examples. For instance, FIG. 6 shows a more general operational flow diagram for certain embodiments. In operational block **601**, LOL is detected for a processor in a multi-processor system. In operational block **602**, the system (e.g., system firmware) determines whether the processor for which LOL is detected is the system's boot processor. As described above, this is determined in certain embodiments by accessing the node of the device tree corresponding to the processor for which LOL is detected and determining if the information in that node of the device tree indicates that the processor is the system's boot processor.

[0074] In operational block **603**, responsive to determining that the processor for which LOL is detected is the system's boot processor, the system's firmware switches the role of boot processor to a spare processor, which may be either a hot spare or an active spare, as described above. Accordingly, the system's OS recognizes the spare processor as the system boot processor. In view of the above, embodiments hereof enable the role of boot processor to be dynamically switched, during system runtime, to a spare processor responsive to a detected LOL for the boot processor, thus enabling recovery from the LOL without requiring that the system be crashed or re-booted.

Original

Classifications

IPC	CPC	US Class	ECLA	ICO
G06F11/00	G06F11/1641	714/12	G06F11/20P10	S06F11:20P4
	G06F11/1658	714/11	G06F11/16C6	
	G06F11/2028		G06F11/16C8	
	G06F11/2035		G06F11/16D	
	G06F11/2038		G06F11/20P2E	
	G06F11/2043		G06F11/20P6	

Persons

Ultimate Owner

Original

FORAS TECHNOLOGIES (/api/control?client=&query=uo:("FORAS%20TECHNOLOGIES")&autosearch=true)

Original Assignees

Original	Standardized	Normalized
Scott L. Michaelis (/api/control?client=&query=pa:("Scott%20L.%20Michaelis")&autosearch=true) Plano, Texas, United States of America, US		
Anurupa Rajkumari (/api/control?client=&query=pa:("Anurupa%20Rajkumari")&autosearch=true) Round Rock, Texas, United States of America, US		
William B. McHardy (/api/control?client=&query=pa:("William%20B.%20McHardy")&autosearch=true) Fort Collins, Colorado, United States of America, US		

Current Assignees

Original	Standardized	Normalized
SONRAÍ MEMORY, LTD. (/api/control?client=&query=pacu:("SONRA%C3%8D%20MEMORY%2C%20LTD.")&autosearch=true) SUITE 23, THE HYDE BUILDING, THE PARK, CARRICKMINE, DUBLIN 18, DUBLIN, IRELAND	SONRAI MEMORY (/api/control?client=&query=pacus:("SONRAI%20MEMORY")&autosearch=true)	

Inventor(s)

Original

Scott L. Michaelis (/api/control?client=&query=in:("Scott%20L.%20Michaelis")&autosearch=true)
Plano, Texas, United States of America, US

Anurupa Rajkumari (/api/control?client=&query=in:("Anurupa%20Rajkumari")&autosearch=true)
Round Rock, Texas, United States of America, US

William B. McHardy (/api/control?client=&query=in:("William%20B.%20McHardy")&autosearch=true)
Fort Collins, Colorado, United States of America, US

Legal information

Standardized Post issuance INPADOC

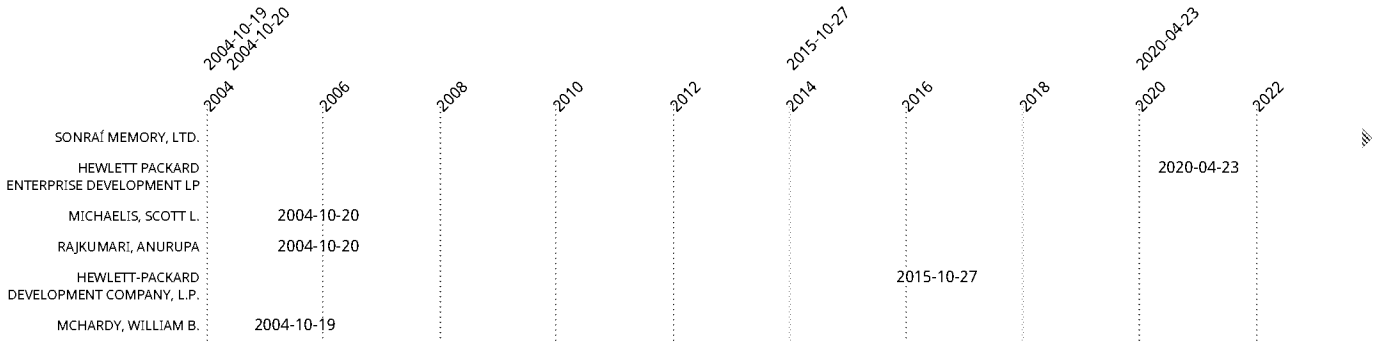
Legal status

Event	Date	Description
Granted	2009-11-24	GRANTED
Filed	2004-10-25	FILED

Designated states

Date	Event code	Description

Ownership



Payment status

Date	Status	Event code	Description
2021-05-12	Fee paid	M1553	PAYMENT OF MAINTENANCE FEE, 12TH YEAR, LARGE ENTITY.
2017-06-28	Fee paid	M1552	PAYMENT OF MAINTENANCE FEE, 8TH YEAR, LARGE ENTITY.
2013-03-08	Fee paid	M1551	PAYMENT OF MAINTENANCE FEE, 4TH YEAR, LARGE ENTITY.

Litigation/Opposition

Docket number	Docket description	Court	Date Filed

No CourtLink®

Pharmaceuticals

Product	Type	Active Ingredient	Applicant	Application No.	Application Type	Approval Date	Patent Expiry Date	Patent usage code	Exclusivity Code	Exclusivity Date	Dosage	Strength	Product Number	Route	RLD	Patent Delist Requested	Excl Text

No pharmaceuticals data

Citations

Display in a list (/api/control?client=&citationsSearch=_)

☐							
☐	backward	US4358823A	By other than the examiner	1982-11-09	Double redundant processor	TRW	
☐	backward	US5226152A	By other than the examiner	1993-07-06	Functional lockstep arrangement for redundant processors	MOTOROLA	
☐	backward	US5249188A	By other than the examiner	1993-09-28	Synchronizing two processors as an integral part of fault detection	COMMUNICATION SYSTEMS	
☐	backward	US5537655A	By other than the examiner	1996-07-16	Synchronized fault tolerant reset	BOEING	
☐	backward	US5675794A	By other than the examiner	1997-10-07	Method and apparatus for configuring multiple agents in a computer system	INTEL	
☐	backward	US5751932A	By other than the examiner	1998-05-12	Fail-fast, fail-functional, fault-tolerant multiprocessor system	TANDEM COMPUTERS	
☐	backward	US5758058A	By other than the examiner	1998-05-26	Apparatus and method for initializing a master/checker fault detecting microprocessor	INTEL	
☐	backward	US5764660A	By other than the examiner	1998-06-09	Processor independent error checking arrangement	ELSAG INT	
☐	backward	US5896523A	By other than the examiner	1999-04-20	Loosely-coupled, synchronized execution	MARATHON TECHN	
☐	backward	US5915082A	Cited by the examiner	1999-06-22	Error detection and fault isolation for lockstep processor systems	LOCKHEED	
☐	backward	US5991900A	By other than the examiner	1999-11-23	Bus controller	SUN MICROSYSTEMS	
☐	backward	US6065135A	By other than the examiner	2000-05-16	Error detection and fault isolation for lockstep processor systems	LOCKHEED	
☐	backward	US6141770A	By other than the examiner	2000-10-31	Fault tolerant computer system	GEN DYNAMICS INF SYSTEMS	
☐	backward	US6148348A	By other than the examiner	2000-11-14	Bridge interfacing two processing sets operating in a lockstep mode and having a posted write buffer storing write operations upon detection of a lockstep error	SUN MICROSYSTEMS	
☐	backward	US6220581B1	By other than the examiner	2001-04-24	Bicycle seat gas spring adjustment system	MUELLER THOMAS L	
☐	backward	US6263452B1	By other than the examiner	2001-07-17	Fault-tolerant computer system with online recovery and reintegration of redundant components	COMPAQ COMPUTER	
☐	backward	US6438687B2	By other than the examiner	2002-08-20	Method and apparatus for improved storage of computer system configuration information	MICRON TECHNOLOGY	
☐	backward	US6473869B2	By other than the examiner	2002-10-29	Fault resilient/fault tolerant computing	MARATHON TECHNOLOGIES	
☐	backward	US6560682B1	By other than the examiner	2003-05-06	System and method for terminating lock-step sequences in a multiprocessor system	INTEL	
☐	backward	US6604177B1	By other	2003-08-05	Communication of dissimilar data between lock-	HEWLETT PACKARD	

than the examiner	stepped processors	DEVELOPMENT

Non-patent literature

U.S. Appl. No. 10/972,796, filed concurrently herewith, entitled "System and Method for Maintaining in a Multi-Processor System a Spare Processor That is in Lockstep for Use in Recovering From Loss of Lockstep for Another Processor".
U.S. Appl. No. 10/973,077, filed concurrently herewith, entitled "System and Method for Establishing a Spare Processor for Recovering From Loss of Lockstep in a Boot Processor".
U.S. Appl. No. 10/972,835, filed concurrently herewith, entitled "System and Method for Using Information Relating to a Detected Loss of Lockstep for Determining a Responsive Action".
U.S. Appl. No. 10/973,076, filed concurrently herewith, entitled "System and Method for Providing Firmware Recoverable Lockstep Protection".
U.S. Appl. No. 10/973,004, filed concurrently herewith, entitled "System and Method for Configuring Lockstep Mode of a Prcessor Module".
U.S. Appl. No. 10/973,003, filed concurrently herewith, entitled "System and Method for Reestablishing Lockstep for a Processor Module for Which Loss of Lockstep is Detected".
U.S. Appl. No. 10/972,888, filed concurrently herewith, entitled "System and Method for System Firmware Causing an Operating System to Idle a Processor".
U.S. Appl. No. 10/973,075, filed concurrently herewith, entitled "System and Method for Reintroducing a Processor Module to an Operating System After Lockstep Recovery".
Notice of Allowance for U.S. Appl. No. 10/972,835 dated Nov. 20, 2008 in the United States.
Notice of Allowance for U.S. Appl. No. 10/972,835 dated Aug. 1, 2008 in the United States.
Office Action for U.S. Appl. No. 10/972,835 dated Mar. 3, 2008 in the United States.
Office Action for U.S. Appl. No. 10/972,835 dated Jul. 12, 2007 in the United States.
Office Action for U.S. Appl. No. 10/973,003 dated Mar. 5, 2008 in the United States.
Office Action for U.S. Appl. No. 10/973,003 dated Sep. 12, 2007 in the United States.
Notice of Allowance for U.S. Appl. No. 10/973,076 dated Nov. 28, 2008 in the United States.
Notice of Allowance for U.S. Appl. No. 10/973,076 dated Jul. 16, 2008 in the United States.
Notice of Allowance for U.S. Appl. No. 10/973,076 dated Jun. 3, 2008 in the United States.
Final Office Action for U.S. Appl. No. 10/973,076 dated Feb. 27, 2008 in the United States.
Office Action for U.S. Appl. No. 10/973,076 dated Sep. 12, 2007 in the United States.
Notice of Allowance for U.S. Appl. No. 10/972,888 dated Nov. 14, 2007 in the United States.
Office Action for U.S. Appl. No. 10/972,888 dated Jul. 18, 2007 in the United States.
Notice of Allowance for U.S. Appl. No. 10/973,004 dated Aug. 1, 2007 in the United States.
Office Action for U.S. Appl. No. 10/973,004 dated Apr. 10, 2007 in the United States.
Final Office Action for U.S. Appl. No. 10/973,075 dated May 12, 2009 in the United States.
Office Action for U.S. Appl. No. 10/973,075 dated Oct. 29, 2008 in the United State.
Office Action for U.S. Appl. No. 10/973,075 dated Mar. 12, 2008 in the United States.
Office Action for U.S. Appl. No. 10/973,075 dated Jul. 2, 2007 in the United States.
Notice of Allowance for U.S. Appl. No. 10/972,796 dated Dec. 19, 2007 in the United States.
Office Action for U.S. Appl. No. 10/972,796 dated Jul. 12, 2007 in the United States.
Shoichi Araki, Real-Time Tracking of Multiple Moving Object Contours in a Moving Camera Image Sequence, vol. E83-D IEICE Trans. Inf. & Syst, Jul. 7, 2000, 9 pages. (/in tapi/scopus/0034229432) **42** Citations in Scopus
Jiang-Bin Zheng, Proceedings of the First International Conference on Machine Learning and Cybernetics, Beijing, Nov. 4-5, 2002, 4 pages.
Jong Ryul Kim, Automatic Localization and Trackign of Moving Objects Using Adaptive Snake Algorithm, ICICS-PCM 2003, Dec. 15-18, 2003, 5 pages, Singapore.
Jianhoa Meng, CVEPS-A Compressed Video Editing and Parsing System, Nov. 18, 1996, pp. 1-11.
Notice of Allowance dated Feb. 11, 2009 for U.S. Appl. No. 10/973,077 in the United States.
Notice of Allowance dated Sep. 23, 2008 for U.S. Appl. No. 10/973,077 in the United States.
Office Action dated Mar. 19, 2008 for U.S. Appl. No. 10/973,077 in the United States.
Office Action dated Sep. 20, 2007 for U.S. Appl. No. 10/973,077 in the United States.
Final Office Action for U.S. Appl. No. 10/972,835 dated Nov. 21, 2007 in the United States.
Office Action for U.S. Appl. No. 10/973,075 dated Dec. 13, 2007 in the United States.
Notice of Allowance, dated Jul. 23, 2009, U.S. Appl. No. 10/973,077, 20 pages.

Similar documents

Display in a list (/api/control?client=&similarSearch=_)

The similarity check is done by matching this document to all documents in the database. Only the title and abstract text are matched, resulting in a score. The 10 highest-scoring documents are displayed here.

☐ All

☐ System and method for maintaining in a multi-processor system a spare processor that is in lockstep for use in recovering from loss of lockstep for another processor

Publication number	US20060107106A1
Publication date	2006-05-18
Application date	2004-10-25

According to one embodiment, a method comprises assigning a first processor of a multi-processor system a role of spare processor for at least a second processor, and responsive to detecting loss of lockstep (LOL) for any of the at least a second processor, the first processor replaces the processor for which the LOL is detected. The method further comprises reestablishing lockstep for the processor for which the LOL is detected, and assigning the processor having its lockstep reestablished the role of spare processor for at least the first processor.

☐ System and method for maintaining in a multi-processor system a spare processor that is in lockstep for use in recovering from loss of lockstep for another processor

Publication number	US7366948B2
Publication date	2008-04-29
Application date	2004-10-25

According to one embodiment, a method comprises assigning a first processor of a multi-processor system a role of spare processor for at least a second processor, and responsive to detecting loss of lockstep (LOL) for any of the at least a second processor, the first processor replaces the processor for which the LOL is detected. The method further comprises reestablishing lockstep for the processor for which the LOL is detected, and assigning the

processor having its lockstep reestablished the role of spare processor for at least the first processor.

☐ System and method for reestablishing lockstep for a processor module for which loss of lockstep is detected

Publication number US20060107116A1
Publication date 2006-05-18
Application date 2004-10-25

According to one embodiment, a method comprises, responsive to detection of loss of lockstep (LOL) for a processor module in a system, firmware requesting an operating system to idle the processor module. The method further comprises the operating system idling the processor module and returning control of the processor module to the firmware. According to one embodiment, a method comprises an operating system idling a processor module for which A loss of lockstep (LOL) is detected, and system firmware receiving control of the processor module. The method further comprises the system firmware determining whether a LOL was detected for the processor module, and if determined that LOL was detected for the processor module, the system firmware resetting the processor module to reestablish lockstep for the processor module.

☐ System and method for establishing a spare processor for recovering from loss of lockstep in a boot processor

Publication number US7627781B2
Publication date 2009-12-01
Application date 2004-10-25

A system comprises a plurality of processors, and data storage storing information that assigns a role of boot processor to one of the plurality of processors and assigns a role of spare processor to another of the plurality of processors. The system further comprises logic operable, responsive to detecting loss of lockstep for the boot processor, for transferring, during system runtime, the role of boot processor to the spare processor.

☐ System and method for establishing a spare processor for recovering from loss of lockstep in a boot processor

Publication number US20060107112A1
Publication date 2006-05-18
Application date 2004-10-25

A system comprises a plurality of processors, and data storage storing information that assigns a role of boot processor to one of the plurality of processors and assigns a role of spare processor to another of the plurality of processors. The system further comprises logic operable, responsive to detecting loss of lockstep for the boot processor, for transferring, during system runtime, the role of boot processor to the spare processor.

☐ System and method for reintroducing a processor module to an operating system after lockstep recovery

Publication number US20060107111A1
Publication date 2006-05-18
Application date 2004-10-25

According to one embodiment, a method comprises, responsive to detection of loss of lockstep (LOL) for a processor module in a system, setting status information stored to the system for the processor module to indicate that the processor module has an error. The method further comprises reestablishing lockstep for the processor module without shutting down the system's operating system, and updating the status information for the processor module to indicate that the processor module no longer has the error. The method further comprises causing the system's operating system to recognize that the processor module having its lockstep reestablished is available for processing.

☐ System and method for reintroducing a processor module to an operating system after lockstep recovery

Publication number US7818614B2
Publication date 2010-10-19
Application date 2004-10-25

According to one embodiment, a method comprises, responsive to detection of loss of lockstep (LOL) for a processor module in a system, setting status information stored to the system for the processor module to indicate that the processor module has an error. The method further comprises reestablishing lockstep for the processor module without shutting down the system's operating system, and updating the status information for the processor module to indicate that the processor module no longer has the error. The method further comprises causing the system's operating system to recognize that the processor module having its lockstep reestablished is available for processing.

☐ System and method for using information relating to a detected loss of lockstep for determining a responsive action

Publication number US20060107114A1
Publication date 2006-05-18
Application date 2004-10-25

According to one embodiment, a method comprises detecting a loss of lockstep (LOL) for a processor module. The method further comprises determining a type of LOL that is detected, and, based at least in part on the determined type of LOL, determining a responsive action to take for the LOL. According to one embodiment, a method comprises detecting a loss of lockstep (LOL) for a processor module. The method further comprises using information identifying at least one of type of the detected LOL and source of the detected LOL to determine a responsive action to take for the LOL.

☐ System and method for using information relating to a detected loss of lockstep for determining a responsive action

Publication number US7516359B2
Publication date 2009-04-07
Application date 2004-10-25

According to one embodiment, a method comprises detecting a loss of lockstep (LOL) for a processor module. The method further comprises determining a type of LOL that is detected, and, based at least in part on the determined type of LOL, determining a responsive action to take for the LOL. According to one embodiment, a method comprises detecting a loss of lockstep (LOL) for a processor module. The method further comprises using information identifying at least one of type of the detected LOL and source of the detected LOL to determine a responsive action to take for the LOL.

☐ System and method for system firmware causing an operating system to idle a processor

Publication number US20060107115A1
Publication date 2006-05-18
Application date 2004-10-25

According to one embodiment, a method comprises system firmware instructing a system's operating system to idle a processor, and responsive to the instructing, the operating system idling the processor and returning control over the processor to the system firmware. According to one embodiment, a method comprises detecting loss of lockstep (LOL) for a

processor module in a system, and responsive to the detecting LOL for the processor module, system firmware instructing an operating system to idle the processor module.

Results for: 7624302 or 7,624,302 Terms and Connectors

Cases 0

No documents found in Cases

Try the following:

- Review your results in the other categories.
- Modify your search (edit or remove some search terms or add synonyms).
- Check for spelling errors.
- Note: Some content may not be visible based on the restrictions of your subscription.

Copyright © 2024 LexisNexis.

User Name: Renee Preston
Date and Time: Thursday, June 6, 2024 11:11:00AM EDT
Job Number: 226018313

Document (1)

1. *Patent No. 7,624,302 Issued on Nov. 24, Assigned to HP Development for Boot Processor Role Method (Texas , Colorado Inventors)*

Client/Matter: -None-
Search Terms: 7624302 or 7,624,302
Search Type: Terms and Connectors
Narrowed by:

Content Type	Narrowed by
	-None-

Patent No. 7,624,302 Issued on Nov. 24, Assigned to HP Development for Boot Processor Role Method (Texas, Colorado Inventors)

US Fed News

November 25, 2009 Wednesday 10:44 AM EST

Copyright 2009 HT Media Ltd. All Rights Reserved

Length: 302 words

Dateline: ALEXANDRIA, Va.

Body

ALEXANDRIA, Va., Nov. 30 -- Anurupa Rajkumari of Round Rock, Texas, William B. McHardy of Fort Collins, Colo., and Scott L. Michaelis of Plano, Texas, have developed a method for switching the role of boot processor to a spare processor responsive to detection of loss of lockstep in a boot processor. The inventors were issued U.S. ***Patent*** No. ***7,624,302*** on Nov. 24.

The ***patent*** has been assigned to Hewlett-Packard Development Co. LP, Houston.

According to the abstract released by the U.S. ***Patent*** & Trademark Office: "According to one embodiment, a method comprises detecting loss of lockstep (LOL) for a processor in a multi-processor system. The method further comprises determining that the processor for which the LOL is detected is assigned the role of boot processor, and switching the role of boot processor to a spare processor without shutting down the system's operating system. In another embodiment, a method comprises system firmware determining that an LOL is detected for a lockstep pair of processors that are assigned the role of boot processor in a system. The method further comprises determining one of the lockstep pair of processors that is not the cause of the LOL, and copying the state of the determined one of the lockstep pair of processors that is not the cause of the LOL to a spare processor. The method further comprises switching the role of boot processor to the spare processor."

The original application was filed on Oct. 25, 2004.

For further information please visit:http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&p=1&u=%2Fnetahtml%2FPTO%2Fsearch-bool.html&r=1&f=G&l=50&co1=AND&d=PTXT&s1=7624302&OS=7624302&RS=7624302 For more information about US Fed News contract awards please contact: Sarabjit Jagirdar, US Fed News, Email:-htsyndication@hindustantimes.com

For any query with respect to this article or any other content requirement, please contact Editor at htsyndication@hindustantimes.com

Load-Date: November 30, 2009

End of Document

User Name: Renee Preston
Date and Time: Thursday, June 6, 2024 10:50:00AM EDT
Job Number: 226015352

Document (1)

1. 1:23cv640, Foras Technologies Ltd. V. Nissan Motor Company, Ltd. Et Al
   Client/Matter: -None-
   Search Terms: 7624302 or 7,624,302
   Search Type: Terms and Connectors
   Narrowed by:

Content Type	Narrowed by
Dockets	-None-

1:23cv640, Foras Technologies Ltd. V. Nissan Motor Company, Ltd. Et Al

US District Court Docket

United States District Court, Texas Western

(Austin)

This case was retrieved on 06/06/2024

Header

Case Number: 1:23cv640
Date Filed: 06/06/2023
Assigned To: Judge Robert Pitman
Nature of Suit: Patent (830)
Cause: Patent Infringement
Lead Docket: None
Other Docket: 6:23cv00386
Jurisdiction: Federal Question

Class Code: Open
Statute: 35:271
Jury Demand: Plaintiff
Demand Amount: $0
NOS Description: Patent

Participants

Litigants

Foras Technologies Ltd.
Plaintiff

Attorneys

Jonathan Randy Yim
LEAD ATTORNEY;ATTORNEY TO BE NOTICED
BC Law Group, P.C.
200 Madison Avenue 24th Floor
New York, NY 10016
USA
516-359-9668 Email:Jyim@bclgpc.Com

Ashley Marie Ratycz
ATTORNEY TO BE NOTICED
BC Law Group, P.C.
200 Madison Avenue, 24th Floor
New York, NY 10016
USA
212-951-0100 Email:Aratycz@bclgpc.Com

Drew B Hollander
ATTORNEY TO BE NOTICED
BC Law Group, P.C.
200 Madison Avenue 24th Floor
New York, NY 10016
USA
212-951-0100 Email:Dhollander@bclgpc.Com

Seth Raymond Hasenour
ATTORNEY TO BE NOTICED
BC Law Group, P.C.
607 Terrell Hill Drive Ste Unit B
Austin, TX 78704

Litigants	Attorneys
	USA 516-359-9668 Email:Shasenour@bclgpc.Com Brett E Cooper ATTORNEY TO BE NOTICED BC LAW GROUP, P.C. 200 Madison Avenue, 24th Floor New York, NY 10016 USA 516-359-9668 Email:Bcooper@b-Clg.Com
Nissan Motor Company, Ltd. **Defendant**	Binal J. Patel LEAD ATTORNEY;PRO HAC VICE;ATTORNEY TO BE NOTICED Banner & Witcoff, LTD 71 S. Wacker Drive, Ste 3600 Chicago, IL 60606 USA 312-463-5463 Fax: 312-463-5001 Email:Bpatel@bannerwitcoff.Com John R. Hutchins LEAD ATTORNEY;PRO HAC VICE;ATTORNEY TO BE NOTICED Banner & Witcoff, Ltd. 1100 13th St. Nw, Suite 1200 Washington, DC 20005 USA 202-824-3000 Fax: 202-824-3001 Email:Jhutchins@bannerwitcoff.Com Paige Arnette Amstutz LEAD ATTORNEY;ATTORNEY TO BE NOTICED Scott, Douglass & McConnico, LLP 303 Colorado Street Suite 2400 Austin, TX 78701 USA (512) 495-6300 Fax: 512-495-6399 Email:Pamstutz@scottdoug.Com Paul Thomas Qualey LEAD ATTORNEY;PRO HAC VICE;ATTORNEY TO BE NOTICED Banner & Witcoff, LTD 1100 13th St., Nw, Suite 1200 Washington D.C., DC 20005 USA 202-824-3000 Fax: 202-824-3001 Email:Pqualey@bannerwitcoff.Com Wesley W. Jones LEAD ATTORNEY;PRO HAC VICE;ATTORNEY TO BE NOTICED Banner & Witcoff, Ltd. 1100 13th St. Nw, Suite 1200 Washington, DC 20005 USA 202-824-3000 Fax: 202-824-3001 Email:Wjones@bannerwitcoff.Com

Litigants	Attorneys
	Lauren Christine Ditty ATTORNEY TO BE NOTICED Scott Douglass & McConnico LLP 303 Colorado Street, Suite 2400 Austin, TX 78701 USA 512-495-6300 Fax: 512-495-6399 Email:Lditty@scottdoug.Com
ZF Friedrichshafen AG **Defendant**	Binal J. Patel LEAD ATTORNEY;PRO HAC VICE;ATTORNEY TO BE NOTICED Banner & Witcoff, LTD 71 S. Wacker Drive, Ste 3600 Chicago, IL 60606 USA 312-463-5463 Fax: 312-463-5001 Email:Bpatel@bannerwitcoff.Com John R. Hutchins LEAD ATTORNEY;PRO HAC VICE;ATTORNEY TO BE NOTICED Banner & Witcoff, Ltd. 1100 13th St. Nw, Suite 1200 Washington, DC 20005 USA 202-824-3000 Fax: 202-824-3001 Email:Jhutchins@bannerwitcoff.Com Paige Amette Amstutz LEAD ATTORNEY;ATTORNEY TO BE NOTICED Scott, Douglass & McConnico, LLP 303 Colorado Street Suite 2400 Austin, TX 78701 USA (512) 495-6300 Fax: 512-495-6399 Email:Pamstutz@scottdoug.Com Paul Thomas Qualey LEAD ATTORNEY;PRO HAC VICE;ATTORNEY TO BE NOTICED Banner & Witcoff, LTD 1100 13th St., Nw, Suite 1200 Washington D.C., DC 20005 USA 202-824-3000 Fax: 202-824-3001 Email:Pqualey@bannerwitcoff.Com Wesley W. Jones LEAD ATTORNEY;PRO HAC VICE;ATTORNEY TO BE NOTICED Banner & Witcoff, Ltd. 1100 13th St. Nw, Suite 1200 Washington, DC 20005 USA 202-824-3000 Fax: 202-824-3001 Email:Wjones@bannerwitcoff.Com Lauren Christine Ditty ATTORNEY TO BE NOTICED

Litigants

Attorneys

Scott Douglass & McConnico LLP
303 Colorado Street, Suite 2400
Austin, TX 78701
USA
512-495-6300 Fax: 512-495-6399
Email:Lditty@scottdoug.Com

Proceedings

#	Date	Proceeding Text	Source
1	06/06/2023	COMPLAINT (Filing fee $ 402 receipt number ATXWDC-17522685), filed by Foras Technologies Ltd.. (Attachments: # 1 Exhibit 1, # 2 Exhibit 2, # 3 Exhibit 3, # 4 Exhibit 4, # 5 Exhibit 5, # 6 Exhibit 6, # 7 Exhibit 7, # 8 Exhibit 8, # 9 Exhibit 9, # 10 Exhibit 10, # 11 Exhibit 11, # 12 Civil Cover Sheet)(Cooper, Brett) (Entered: 06/06/2023)	
2	06/06/2023	Notice of Filing of Patent/Trademark Form (AO 120). AO 120 forwarded to the Director of the U.S. Patent and Trademark Office. (Cooper, Brett) (Entered: 06/06/2023)	
3	06/06/2023	RULE 7 DISCLOSURE STATEMENT filed by Foras Technologies Ltd.. (Cooper, Brett) (Entered: 06/06/2023)	
4	06/06/2023	NOTICE Notice of Related Cases by Foras Technologies Ltd. (Cooper, Brett) (Entered: 06/06/2023)	
5	06/06/2023	REQUEST FOR ISSUANCE OF SUMMONS by Foras Technologies Ltd.. as to Nissan Motor Company, Ltd. (Cooper, Brett) (Entered: 06/06/2023)	
6	06/06/2023	REQUEST FOR ISSUANCE OF SUMMONS by Foras Technologies Ltd.. as to ZF Friedrichshafen AG (Cooper, Brett) (Entered: 06/06/2023)	
	06/06/2023	Case assigned to Judge Robert Pitman. CM WILL NOW REFLECT THE JUDGE INITIALS AS PART OF THE CASE NUMBER. PLEASE APPEND THESE JUDGE INITIALS TO THE CASE NUMBER ON EACH DOCUMENT THAT YOU FILE IN THIS CASE. (pg) (Entered: 06/09/2023)	
	06/06/2023	If ordered by the court, all referrals and consents in this case will be assigned to Magistrate Judge Howell (pg) (Entered: 06/09/2023)	
7	06/09/2023	Summons Issued as to Nissan Motor Company, Ltd. (pg) (Entered: 06/09/2023)	
8	06/09/2023	Summons Issued as to ZF Friedrichshafen AG. (pg) (Entered: 06/09/2023)	
9	06/09/2023	Report on Patent/Trademark sent to U.S. Patent and Trademark Office with Complaint. (pg) (Entered: 06/09/2023)	
10	09/12/2023	MOTION for Order Appointing International Process Server by Foras Technologies Ltd.. (Attachments: # 1 Proposed Order)(Cooper, Brett) (Entered: 09/12/2023)	
11	09/27/2023	ORDER GRANTING 10 MOTION for Order Appointing International Process Server. Signed by Judge Robert Pitman. (cc3) (Entered: 09/27/2023)	
12	12/04/2023	Unopposed MOTION for Extension of Time to File Response/Reply as to 1 Complaint, ZF Friedrichshafen AG's Unopposed Motion for Extension of Time to Respond to the Complaint by ZF Friedrichshafen AG. (Attachments: # 1 Proposed Order Proposed Order)(Amstutz, Paige) (Entered: 12/04/2023)	
	12/05/2023	Text Order GRANTING 12 Unopposed Motion for Extension of Time to File Response/Reply entered by Judge Robert Pitman. IT	

#	Date	Proceeding Text	Source
		IS ORDERED that ZF FRIEDRICHSHAFEN AG shall answer or otherwise respond to the complaint on or before January 8, 2024. (This is a text-only entry generated by the court. There is no document associated with this entry.) (sh) (Entered: 12/05/2023)	
	12/05/2023	Reset Answer Deadline: ZF Friedrichshafen AG answer due 1/8/2024. (cc3) (Entered: 12/05/2023)	
13	12/06/2023	SUMMONS Returned Executed by Foras Technologies Ltd.. ZF Friedrichshafen AG served on 11/13/2023, answer due 1/8/2024. (Cooper, Brett) (Entered: 12/06/2023)	
14	12/28/2023	Unopposed MOTION for Extension of Time to File Answer re 1 Complaint, by Nissan Motor Company, Ltd.. (Attachments: # 1 Proposed Order to Unopposed Motion)(Amstutz, Paige) (Entered: 12/28/2023)	
	01/02/2024	Text Order GRANTING 14 Unopposed Motion for Extension of Time to Answer entered by Judge Robert Pitman. IT IS ORDERED that Nissan shall answer or otherwise respond to the complaint on or before February 9, 2024. (This is a text-only entry generated by the court. There is no document associated with this entry.) (sh) (Entered: 01/02/2024)	
	01/02/2024	Reset Answer Deadline: Nissan Motor Company, Ltd. answer due 2/9/2024. (cc3) (Entered: 01/02/2024)	
15	01/08/2024	MOTION to Dismiss for Lack of Jurisdiction Defendant ZF Friedrichshafen AG's Motion to Dismiss the Complaint Pursuant to Fed. R. Civ. P. 12(b)(2) by ZF Friedrichshafen AG. (Attachments: # 1 Affidavit Declaration of David Council ISO ZF Friedrichshafen AG's Motion to Dismiss Plaintiff's Complaint)(Amstutz, Paige) (Entered: 01/08/2024)	
16	01/10/2024	RULE 7 DISCLOSURE STATEMENT filed by ZF Friedrichshafen AG. (Amstutz, Paige) (Entered: 01/10/2024)	
17	01/16/2024	Unopposed MOTION for Extension of Time to File Response/Reply as to 15 MOTION to Dismiss for Lack of Jurisdiction Defendant ZF Friedrichshafen AG's Motion to Dismiss the Complaint Pursuant to Fed. R. Civ. P. 12(b)(2) by Foras Technologies Ltd.. (Attachments: # 1 Proposed Order)(Cooper, Brett) (Entered: 01/16/2024)	
	01/18/2024	Text Order GRANTING 17 Unopposed Motion for Extension of Time to File Response/Reply entered by Judge Robert Pitman. IT IS ORDERED that Plaintiff shall respond on or before January 29, 2024. (This is a text-only entry generated by the court. There is no document associated with this entry.) (sh) (Entered: 01/18/2024)	
18	01/29/2024	Response in Opposition to Motion, filed by Foras Technologies Ltd., re 15 MOTION to Dismiss for Lack of Jurisdiction Defendant ZF Friedrichshafen AG's Motion to Dismiss the Complaint Pursuant to Fed. R. Civ. P. 12(b)(2) filed by Defendant ZF Friedrichshafen AG (Cooper, Brett) (Entered: 01/29/2024)	
19	02/05/2024	RESPONSE in Support, filed by ZF Friedrichshafen AG, re 15 MOTION to Dismiss for Lack of Jurisdiction Defendant ZF Friedrichshafen AG's Motion to Dismiss the Complaint Pursuant to Fed. R. Civ. P. 12(b)(2) filed by Defendant ZF Friedrichshafen AG Defendant ZF Friedrichshafen AG's Reply Brief In Support of its Motion to Dismiss the Complaint Pursuant to Fed. R. Civ. P. 12(b)(2) (Amstutz, Paige) (Entered: 02/05/2024)	
20	02/06/2024	SUMMONS Returned Executed by Foras Technologies Ltd.. Nissan Motor Company, Ltd. served on 12/20/2023, answer due 2/9/2024. (Cooper, Brett) (Entered: 02/06/2024)	
21	02/09/2024	NOTICE of Translation to English by Foras Technologies Ltd. re 20 Summons Returned Executed (Cooper, Brett) (Entered: 02/09/2024)	

#	Date	Proceeding Text	Source
22	02/09/2024	NOTICE of Translation to English by Foras Technologies Ltd. re 13 Summons Returned Executed (Cooper, Brett) (Entered: 02/09/2024)	
23	02/09/2024	Motion to Dismiss for Lack of Jurisdiction Defendant Nissan Motor Company, Ltd.'s Motion to Dismiss the Complaint Pursuant to Fed. R. Civ. P. 12(b)(2) by Nissan Motor Company, Ltd.. (Attachments: # 1 Affidavit Declaration of Jacquelyn Trussell ISO Nissan Motor Company, Ltd.'s Motion to Dismiss Plaintiff's Complaint)(Amstutz, Paige) Modified on 2/12/2024 To edit text (bc). (Entered: 02/09/2024)	
24	02/23/2024	Response in Opposition to Motion, filed by Foras Technologies Ltd., re 23 MOTION to Dismiss for Lack of Jurisdiction filed by Defendant Nissan Motor Company, Ltd. (Cooper, Brett) (Entered: 02/23/2024)	
25	03/01/2024	RESPONSE in Support, filed by Nissan Motor Company, Ltd., re 23 MOTION to Dismiss for Lack of Jurisdiction filed by Defendant Nissan Motor Company, Ltd. Defendant Nissan Motor Company Ltd.'s Reply in Support of Its Motion to Dismiss the Complaint Pursuant to Fed. R. Civ. P. 12(b)(2) (Amstutz, Paige) (Entered: 03/01/2024)	
26	03/13/2024	Order for parties to submit a Joint Proposed Scheduling Order; due by 3/27/2024. Signed by Judge Dustin M. Howell. (cc3) (Entered: 03/13/2024)	
27	03/21/2024	RULE 7 DISCLOSURE STATEMENT filed by Nissan Motor Company, Ltd. Identifying Corporate Parent None for Nissan Motor Company, Ltd.. (Amstutz, Paige) (Entered: 03/21/2024)	
28	03/22/2024	CERTIFICATE OF SERVICE by Nissan Motor Company, Ltd. 27 Disclosure Statement (Rule 7) (Amstutz, Paige) (Entered: 03/22/2024)	
29	03/22/2024	MOTION to Appear Pro Hac Vice by Paige Arnette Amstutz for Wesley W. Jones (Filing fee $ 100 receipt number ATXWDC-18562555) by on behalf of Nissan Motor Company, Ltd., ZF Friedrichshafen AG. (Attachments: # 1 Proposed Order)(Amstutz, Paige) (Entered: 03/22/2024)	
30	03/22/2024	MOTION to Appear Pro Hac Vice by Paige Arnette Amstutz for Binal J. Patel (Filing fee $ 100 receipt number ATXWDC-18562636) by on behalf of Nissan Motor Company, Ltd., ZF Friedrichshafen AG. (Attachments: # 1 Proposed Order)(Amstutz, Paige) (Entered: 03/22/2024)	
31	03/22/2024	MOTION to Appear Pro Hac Vice by Paige Arnette Amstutz for John R. Hutchins (Filing fee $ 100 receipt number ATXWDC-18562645) by on behalf of Nissan Motor Company, Ltd., ZF Friedrichshafen AG. (Attachments: # 1 Proposed Order)(Amstutz, Paige) (Entered: 03/22/2024)	
32	03/27/2024	ORDER GRANTING 29 Motion to Appear Pro Hac Vice as to WESLEY W. JONES. Pursuant to our Administrative Policies and Procedures for Electronic Filing, the attorney hereby granted to practice pro hac vice in this case must register for electronic filing with our court within 10 days of this order. Registration is managed by the PACER Service Center Signed by Judge Robert Pitman. (cc3) (Entered: 03/27/2024)	
33	03/27/2024	ORDER GRANTING 30 Motion to Appear Pro Hac Vice as to BINAL J. PATEL. Pursuant to our Administrative Policies and Procedures for Electronic Filing, the attorney hereby granted to practice pro hac vice in this case must register for electronic filing with our court within 10 days of this order. Registration is managed by the PACER Service Center Signed by Judge Robert Pitman. (cc3) (Entered: 03/27/2024)	
34	03/27/2024	ORDER GRANTING 31 Motion to Appear Pro Hac Vice as to	

#	Date	Proceeding Text	Source
		JOHN R. HUTCHINS. Pursuant to our Administrative Policies and Procedures for Electronic Filing, the attorney hereby granted to practice pro hac vice in this case must register for electronic filing with our court within 10 days of this order. Registration is managed by the PACER Service Center Signed by Judge Robert Pitman. (cc3) (Entered: 03/27/2024)	
35	03/27/2024	NOTICE of Attorney Appearance by Ashley Marie Ratycz on behalf of Foras Technologies Ltd.. Attorney Ashley Marie Ratycz added to party Foras Technologies Ltd.(pty:pla) (Ratycz, Ashley) (Entered: 03/27/2024)	
36	03/27/2024	Rule 26(f) Discovery Report/Case Management Plan - Joint Rule 26 Report by Foras Technologies Ltd.. (Cooper, Brett) (Entered: 03/27/2024)	
37	03/27/2024	Joint MOTION for Entry of Scheduling Order Deadlines by Foras Technologies Ltd.. (Attachments: # 1 Exhibit A - Agreed Proposed Scheduling Order)(Cooper, Brett) (Entered: 03/27/2024)	
38	04/02/2024	MOTION to Appear Pro Hac Vice by Paige Arnette Amstutz Motion for Admission Pro Hac Vice Paul T. Qualey for Nissan Motor Company, Ltd. and ZF Friedrichshafen AG (Filing fee $ 100 receipt number ATXWDC-18597104) by on behalf of Nissan Motor Company, Ltd., ZF Friedrichshafen AG. (Attachments: # 1 Proposed Order Proposed Order)(Amstutz, Paige) (Entered: 04/02/2024)	
39	04/03/2024	ORDER GRANTING 38 Motion to Appear Pro Hac Vice as to PAUL T. QUALEY. Pursuant to our Administrative Policies and Procedures for Electronic Filing, the attorney hereby granted to practice pro hac vice in this case must register for electronic filing with our court within 10 days of this order. Registration is managed by the PACER Service Center Signed by Judge Robert Pitman. (cc3) (Entered: 04/03/2024)	
40	04/09/2024	ORDER Setting Initial Pretrial Conference for 4/22/2024 01:30 PM before Judge Dustin M. Howell. Signed by Judge Dustin M. Howell. (bot2) (Entered: 04/09/2024)	
41	04/10/2024	Opposed MOTION to Stay Case by Nissan Motor Company, Ltd., ZF Friedrichshafen AG. (Attachments: # 1 Affidavit of John R. Hutchins, # 2 Exhibit 1, # 3 Exhibit 2, # 4 Exhibit 3, # 5 Exhibit 4, # 6 Exhibit 5, # 7 Exhibit 6, # 8 Exhibit 7, # 9 Exhibit 8, # 10 Exhibit 9, # 11 Exhibit 10, # 12 Exhibit 11, # 13 Exhibit 12, # 14 Proposed Order Granting Motion to Stay)(Hutchins, John) (Entered: 04/10/2024)	
42	04/15/2024	Unopposed MOTION for Extension of Time to File Response/Reply as to 41 Opposed MOTION to Stay Case by Foras Technologies Ltd.. (Attachments: # 1 Proposed Order)(Cooper, Brett) (Entered: 04/15/2024)	
	04/17/2024	Text Order GRANTING 42 Unopposed Motion for Extension of Time to File Response/Reply entered by Judge Robert Pitman. IT IS ORDERED that Plaintiff shall respond to the motion to stay on or before April 26, 2024. (This is a text-only entry generated by the court. There is no document associated with this entry.) (sh) (Entered: 04/17/2024)	
43	04/22/2024	Minute Entry for proceedings held before Judge Dustin M. Howell: Initial Pretrial Conference held by telephone on 4/22/2024. Written order forthcoming (Minute entry documents are not available electronically) (Court Reporter AT&T conf. line)(cc3) (Entered: 04/22/2024)	
44	04/22/2024	AGREED SCHEDULING ORDER: Consent to Trial by Magistrate due by 6/26/2025, Jury Trial set for 9/8/2025 at 9:30 AM before Judge Robert Pitman, ADR Report Deadline due by 6/20/2024, Amended Pleadings due by 1/9/2025, Discovery due by	

#	Date	Proceeding Text	Source
		4/17/2025, Joinder of Parties due by 1/9/2025, and Dispositive Motions due by 6/26/2025. Signed by Judge Robert Pitman. (cc3) (Entered: 04/22/2024)	
45	04/23/2024	NOTICE of Inter Partes Review by Nissan Motor Company, Ltd., ZF Friedrichshafen AG (Hutchins, John) (Entered: 04/23/2024)	
46	04/26/2024	Response in Opposition to Motion, filed by Foras Technologies Ltd., re 41 Opposed MOTION to Stay Case filed by Defendant ZF Friedrichshafen AG, Defendant Nissan Motor Company, Ltd. (Cooper, Brett) (Entered: 04/26/2024)	
	05/03/2024	Text Order REFERRING 41 Motion to Stay Case entered by Judge Robert Pitman. IT IS ORDERED that Defendants' Motion to Stay is REFERRED to United States Magistrate Judge Dustin Howell for disposition pursuant to 28 U.S.C. § 636(b)(1)(A), Federal Rule of Civil Procedure 72, and Rule 1(c) of Appendix C of the Local Rules of the United States District Court for the Western District of Texas. (This is a text-only entry generated by the court. There is no document associated with this entry.) (sh) (Entered: 05/03/2024)	
	05/03/2024	MOTION REFERRED: referred 41 Opposed MOTION to Stay Case . Referral Judge: Dustin M. Howell. (cc3) (Entered: 05/03/2024)	
47	05/03/2024	REPLY to Response to Motion, filed by Nissan Motor Company, Ltd., ZF Friedrichshafen AG, re 41 Opposed MOTION to Stay Case filed by Defendant ZF Friedrichshafen AG, Defendant Nissan Motor Company, Ltd. (Attachments: # 1 Affidavit of John Hutchins, # 2 Exhibit 13, # 3 Exhibit 14, # 4 Exhibit 15, # 5 Exhibit 16, # 6 Exhibit 17)(Hutchins, John) (Entered: 05/03/2024)	
48	05/08/2024	NOTICE of Attorney Appearance by Lauren Christine Ditty on behalf of Nissan Motor Company, Ltd., ZF Friedrichshafen AG. Attorney Lauren Christine Ditty added to party Nissan Motor Company, Ltd.(pty:dft), Attorney Lauren Christine Ditty added to party ZF Friedrichshafen AG(pty:dft) (Ditty, Lauren) (Entered: 05/08/2024)	
49	05/10/2024	ORDER SETTING HEARING re 41 Opposed MOTION to Stay Case : for 5/21/2024 at 3:00 PM before Judge Dustin M. Howell. Signed by Judge Dustin M. Howell. (cc3) (Entered: 05/10/2024)	
50	05/16/2024	ADVISORY TO THE COURT by Nissan Motor Company, Ltd., ZF Friedrichshafen AG Joint Advisory Regarding Defendants' Motion to Stay. (Hutchins, John) (Entered: 05/16/2024)	
51	05/21/2024	ORDER GRANTING 41 Motion to Stay Case. Signed by Judge Dustin M. Howell. (dm) (Entered: 05/22/2024)	
	05/21/2024	Reset Deadlines: Status Report due by 7/22/2024. (dm) (Entered: 05/22/2024)	
52	05/21/2024	Minute Entry for proceedings held before Judge Dustin M. Howell: Motion Hearing held on 5/21/2024 re 41 Opposed MOTION to Stay Case filed by ZF Friedrichshafen AG, Nissan Motor Company, Ltd. (Minute entry documents are not available electronically.). (Court Reporter FTR Gold -ERO.)(pg) (Entered: 05/22/2024)	

Patents

Number	Title	Issued	Class	Subclass
7,502,958	System and method for providing firmware recoverable lockstep protection	03/10/2009	714	12
7,624,302	System and method for switching the role of boot processor to a spare processor responsive to detection of loss of lockstep in a boot processor	11/24/2009	714	12

Number	Title	Issued	Class	Subclass
7,627,781	System and method for establishing a spare processor for recovering from loss of lockstep in a boot processor	12/01/2009	714	13

Copyright © LexisNexis CourtLink, Inc. All Rights Reserved.
*** THIS DATA IS FOR INFORMATIONAL PURPOSES ONLY ***

End of Document

User Name: Renee Preston
Date and Time: Thursday, June 6, 2024 11:05:00AM EDT
Job Number: 226017454

Document (1)

1. 2:23cv150, Foras Technologies Ltd. V. Toyota Motor North America, Inc. Et Al
   Client/Matter: -None-
   Search Terms: 7624302 or 7,624,302
   Search Type: Terms and Connectors
   Narrowed by:

Content Type	Narrowed by
Dockets	-None-

2:23cv150, Foras Technologies Ltd. V. Toyota Motor North America, Inc. Et Al

US District Court Docket

United States District Court, Texas Eastern

(Marshall)

This case was retrieved on 06/06/2024

Header

Case Number: 2:23cv150
Date Filed: 04/05/2023
Assigned To: District Judge Rodney Gilstrap
Nature of Suit: Patent (830)
Cause: Patent Infringement
Lead Docket: None
Other Docket: None
Jurisdiction: Federal Question

Class Code: Closed
Closed: 05/03/2024
Statute: 35:145
Jury Demand: Both
Demand Amount: $0
NOS Description: Patent

Participants

Litigants

Foras Technologies Ltd.
Plaintiff

Attorneys

Drew Bennett Hollander
ATTORNEY TO BE NOTICED
BC Law Group, P.C.
200 Madison Avenue 24th Floor
New York, NY 10016
USA
212-951-0100 Email:Dhollander@bclgpc.Com

Jonathan Randy Yim
ATTORNEY TO BE NOTICED
BC Law Group, P.C.
200 Madison Avenue 24th Floor
New York, NY 10016
USA
516-359-9668 Email:Jyim@bclgpc.Com

Justin Kurt Truelove
ATTORNEY TO BE NOTICED
Truelove Law Firm
207 N Wellington P O Box 1409
Marshall, TX 75671
USA
903-938-8321 Fax: 903-215-8510
Email:Kurt@trueloveiawfirm.Com

Seth Raymond Hasenour
ATTORNEY TO BE NOTICED
BC Law Group, P.C.

Litigants	Attorneys
	607 Terrell Hill Drive Ste Unit B Austin, TX 78704 USA 516-359-9668 Email:Shasenour@bclgpc.Com Brett E Cooper ATTORNEY TO BE NOTICED BC Law Group, P.C. 200 Madison Avenue 24th Floor New York, NY 10016 USA 516-359-9668 Email:Bcooper@b-Clg.Com
Toyota Motor North America, Inc. [Terminated: 07/12/2023] **Defendant**	Steven Routh LEAD ATTORNEY [Terminated: 07/12/2023] Orrick, Herrington & Sutcliffe LLP Columbia Center 2100 Pennsylvania Ave, Nw Washington, DC 20037 USA 202-339-8509 Fax: 202-339-8500 Email:Srouth@orrick.Com
Denso Corporation **Defendant**	Steven Routh LEAD ATTORNEY;ATTORNEY TO BE NOTICED Orrick, Herrington & Sutcliffe LLP Columbia Center 2100 Pennsylvania Ave, Nw Washington, DC 20037 USA 202-339-8509 Fax: 202-339-8500 Email:Srouth@orrick.Com Andrew Thompson (Tom) Gorham ATTORNEY TO BE NOTICED Gillam & Smith LLP 102 N College, Suite 800 Tyler, TX 75702 USA 9039348450 Fax: 9039349257 Email:Tom@gillamsmithlaw.Com Jasmine Wenxi Zhu ATTORNEY TO BE NOTICED Orrick, Herrington & Sutcliffe LLP 2100 Pennsylvania Ave, Nw Washington, DC 20037 USA 240-515-6856 Email:Jzhu@orrick.Com Melissa Richards Smith ATTORNEY TO BE NOTICED Gillam & Smith, LLP 303 South Washington Avenue Marshall, TX 75670 USA 903-934-8450 Fax: 903-934-9257 Email:Melissa@gillamsmithlaw.Com Sten Anker Jensen ATTORNEY TO BE NOTICED Orrick, Herrington & Sutcliffe LLP - Washington DC Columbia Center 1152 15th Street, Nw Washington, DC 20005-1706

Litigants	Attorneys
	USA 202-339-8436 Fax: 202-339-8500 Email:Sjensen@orrick.Com
Kia Corporation Consolidated Civil Action 2:23cv219 | **Consol Defendant**	Douglas E Lumish LEAD ATTORNEY;ATTORNEY TO BE NOTICED Latham & Watkins LLP - Menlo Park 140 Scott Drive Menlo Park, CA 94025-1008 USA 650-328-4600 Fax: 650-463-2600 Email:Doug.Lumish@lw.Com Adam Michael Greenfield ATTORNEY TO BE NOTICED Latham & Watkins LLP - D.C. 555 Eleventh Street, Nw Suite 1000 Washington, DC 20004 USA 202.637.2200 Fax: 202.637.2201 Email:Adam.Greenfield@lw.Com Andrew Thompson (Tom) Gorham ATTORNEY TO BE NOTICED Gillam & Smith LLP 102 N College, Suite 800 Tyler, TX 75702 USA 9039348450 Fax: 9039349257 Email:Tom@gillamsmithlaw.Com Jacob Vannette ATTORNEY TO BE NOTICED Latham & Watkins LLP 555 Eleventh Street, Nw Suite 1000 Washington, DC 20004 USA 202-637-2200 Email:Jake.Vannette@lw.Com Melissa Richards Smith ATTORNEY TO BE NOTICED Gillam & Smith, LLP 303 South Washington Avenue Marshall, TX 75670 USA 903-934-8450 Fax: 903-934-9257 Email:Melissa@gillamsmithlaw.Com
Kia America, Inc. Consolidated Civil Action 2:23cv219 | **Consol Defendant**	Douglas E Lumish LEAD ATTORNEY;ATTORNEY TO BE NOTICED Latham & Watkins LLP - Menlo Park 140 Scott Drive Menlo Park, CA 94025-1008 USA 650-328-4600 Fax: 650-463-2600 Email:Doug.Lumish@lw.Com Adam Michael Greenfield ATTORNEY TO BE NOTICED Latham & Watkins LLP - D.C. 555 Eleventh Street, Nw Suite 1000

Litigants

Attorneys

Washington, DC 20004
USA
202.637.2200 Fax: 202.637.2201
Email:Adam.Greenfield@lw.Com

Andrew Thompson (Tom) Gorham
ATTORNEY TO BE NOTICED
Gillam & Smith LLP
102 N College, Suite 800
Tyler, TX 75702
USA
9039348450 Fax: 9039349257
Email:Tom@gillamsmithlaw.Com

Jacob Vannette
ATTORNEY TO BE NOTICED
Latham & Watkins LLP
555 Eleventh Street, Nw Suite 1000
Washington, DC 20004
USA
202-637-2200 Email:Jake.Vannette@lw.Com

Melissa Richards Smith
ATTORNEY TO BE NOTICED
Gillam & Smith, LLP
303 South Washington Avenue
Marshall, TX 75670
USA
903-934-8450 Fax: 903-934-9257
Email:Melissa@gillamsmithlaw.Com

Toyota Motor North America, Inc.
Consolidated Civil Action 2:23-321 |
[Terminated: 12/08/2023]
Consol Defendant

Steven Routh

[Terminated: 12/08/2023]
Orrick, Herrington & Sutcliffe LLP
Columbia Center 2100 Pennsylvania Ave, Nw
Washington, DC 20037
USA
202-339-8509 Fax: 202-339-8500 Email:Srouth@orrick.Com

Proceedings

#	Date	Proceeding Text	Source
1	04/05/2023	COMPLAINT against Denso Corporation, Toyota Motor North America, Inc. (Filing fee $ 402 receipt number ATXEDC-9437538.), filed by Foras Technologies Ltd.. (Attachments: # 1 Exhibit 1, # 2 Exhibit 2, # 3 Exhibit 3, # 4 Exhibit 4, # 5 Exhibit 5, # 6 Exhibit 6, # 7 Exhibit 7, # 8 Exhibit 8, # 9 Exhibit 9, # 10 Civil Cover Sheet)(Cooper, Brett) (Entered: 04/05/2023)	
2	04/05/2023	Notice of Filing of Patent/Trademark Form (AO 120). AO 120 mailed to the Director of the U.S. Patent and Trademark Office. (Cooper, Brett) (Entered: 04/05/2023)	
3	04/05/2023	CORPORATE DISCLOSURE STATEMENT filed by Foras Technologies Ltd. (Cooper, Brett) (Entered: 04/05/2023)	
	04/05/2023	Case Assigned to District Judge Rodney Gilstrap. (nkl,) (Entered: 04/05/2023)	
	04/05/2023	In accordance with the provisions of 28 USC Section 636(c), you	

#	Date	Proceeding Text	Source
		are hereby notified that a U.S. Magistrate Judge of this district court is available to conduct any or all proceedings in this case including a jury or non-jury trial and to order the entry of a final judgment. The form Consent to Proceed Before Magistrate Judge is available on our website. All signed consent forms, excluding pro se parties, should be filed electronically using the event Notice Regarding Consent to Proceed Before Magistrate Judge. (nkl,) (Entered: 04/05/2023)	
4	04/05/2023	SUMMONS Issued as to Denso Corporation c/o Denso International America, Inc.c/o Corporation Service Company d/b/a CSC Lawyers Incorporating Service Company, Toyota Motor North America, Inc. c/o CT Corporation System. (Attachments: # 1 Summons(es))(nkl,) (Entered: 04/05/2023)	
5	04/11/2023	SUMMONS Returned Executed by Foras Technologies Ltd.. Toyota Motor North America, Inc. served on 4/6/2023, answer due 4/27/2023. (Cooper, Brett) (Entered: 04/11/2023)	
6	04/12/2023	SUMMONS Returned Executed by Foras Technologies Ltd.. Denso Corporation served on 4/6/2023, answer due 4/27/2023. (Cooper, Brett) (Entered: 04/12/2023)	
7	04/27/2023	***DEFICIENT DOCUMENT***Joint MOTION for Extension of Time to File Response to Complaint by Denso Corporation. (Attachments: # 1 Text of Proposed Order)(Routh, Steven) Modified on 4/27/2023 (ch,). (Entered: 04/27/2023)	
8	04/27/2023	***DEFICIENT DOCUMENT***Joint MOTION for Extension of Time to File Response to Complaint by Toyota Motor North America, Inc.. (Attachments: # 1 Text of Proposed Order)(Routh, Steven) Modified on 4/27/2023 (ch,). (Entered: 04/27/2023)	
	04/27/2023	NOTICE of Deficiency regarding the 7 submitted Needs Certificate of Conference. Correction should be made by one business day (ch,) (Entered: 04/27/2023)	
	04/27/2023	NOTICE of Deficiency regarding the 8 submitted Needs Certificate of Conference. Correction should be made by one business day (ch,) (Entered: 04/27/2023)	
9	04/27/2023	Joint MOTION for Extension of Time to File Response to Complaint by Denso Corporation. (Attachments: # 1 Text of Proposed Order)(Routh, Steven) (Entered: 04/27/2023)	
10	04/27/2023	Joint MOTION for Extension of Time to File Response to Complaint by Toyota Motor North America, Inc.. (Attachments: # 1 Text of Proposed Order)(Routh, Steven) (Entered: 04/27/2023)	
11	04/28/2023	ORDER granting 9 Motion for Extension of Time to File Response to Complaint. Signed by District Judge Rodney Gilstrap on 4/28/2023. (nkl,) (Entered: 05/01/2023)	
12	04/28/2023	ORDER granting 10 Motion for Extension of Time to File Response to Complaint. Signed by District Judge Rodney Gilstrap on 4/28/2023. (nkl,) (Entered: 05/01/2023)	
13	05/23/2023	NOTICE of Attorney Appearance by Drew Bennett Hollander on behalf of Foras Technologies Ltd. (Hollander, Drew) (Entered: 05/23/2023)	
14	05/23/2023	NOTICE of Attorney Appearance by Seth Raymond Hasenour on behalf of Foras Technologies Ltd. (Hasenour, Seth) (Entered: 05/23/2023)	
15	05/23/2023	NOTICE of Attorney Appearance by Jonathan Randy Yim on behalf of Foras Technologies Ltd. (Yim, Jonathan) (Entered: 05/23/2023)	
16	06/13/2023	CONSOLIDATION ORDER - The above-captioned cases are hereby ORDERED to be CONSOLIDATED for all pretrial issues with the LEAD CASE, Case No. 2:23-cv-00150. All parties are instructed to file any future filings in the LEAD CASE. Individual	

#	Date	Proceeding Text	Source
		cases remain active for trial.. Signed by District Judge Rodney Gilstrap on 6/13/2023. (nkl,) (Entered: 06/13/2023)	
17	06/15/2023	ORDER setting Scheduling Conference for 7/20/2023 at 9:30 AM before District Judge Rodney Gilstrap. Signed by District Judge Rodney Gilstrap on 6/15/23. (tkd,) (Entered: 06/16/2023)	
18	06/22/2023	NOTICE of Attorney Appearance by Melissa Richards Smith on behalf of Kia America, Inc., Kia Corporation (Smith, Melissa) (Entered: 06/22/2023)	
19	06/22/2023	Joint MOTION for extension of time for Toyota and DENSO to respond to the complaint and for Foras to serve infringement contentions by Foras Technologies Ltd.. (Attachments: # 1 Proposed Order)(Cooper, Brett) (Entered: 06/22/2023)	
20	06/23/2023	ORDER granting 19 Joint MOTION for extension of time for Toyota and DENSO to respond to the complaint and for Foras to serve infringement contentions. Denso Corporation answer due 7/21/2023; Toyota Motor North America, Inc. answer due 7/21/2023.. Signed by District Judge Rodney Gilstrap on 6/23/2023. (ch,) (Entered: 06/23/2023)	
21	06/27/2023	NOTICE of Attorney Appearance by Douglas E Lumish on behalf of Kia America, Inc., Kia Corporation (Lumish, Douglas) (Entered: 06/27/2023)	
22	06/27/2023	NOTICE of Attorney Appearance by Adam Michael Greenfield on behalf of Kia America, Inc., Kia Corporation (Greenfield, Adam) (Entered: 06/27/2023)	
23	06/27/2023	NOTICE of Attorney Appearance by Jacob Vannette on behalf of Kia America, Inc., Kia Corporation (Vannette, Jacob) (Entered: 06/27/2023)	
24	06/27/2023	NOTICE of Attorney Appearance by Justin Kurt Truelove on behalf of Foras Technologies Ltd. (Truelove, Justin) (Entered: 06/27/2023)	
25	06/28/2023	NOTICE of Attorney Appearance by Justin Kurt Truelove on behalf of Foras Technologies Ltd. (Truelove, Justin) (Entered: 06/28/2023)	
26	07/03/2023	Joint MOTION FOR EXTENSIONS OF TIME REGARDING INFRINGEMENT AND INVALIDITY CONTENTIONS by Kia America, Inc., Kia Corporation. (Attachments: # 1 Proposed Order)(Smith, Melissa) (Entered: 07/03/2023)	
27	07/07/2023	ORDER granting 26 Motion for Extensions of Time Regarding Infringement and Invalidity Contentions. Signed by District Judge Rodney Gilstrap on 7/6/2023. (nkl,) (Entered: 07/07/2023)	
28	07/10/2023	ANSWER to 1 Complaint, by Toyota Motor North America, Inc..(Routh, Steven) (Entered: 07/10/2023)	
29	07/10/2023	Fed. R. Civ. P. 7.1(a)(1) Disclosure Statement filed by Toyota Motor North America, Inc. identifying Corporate Parent Toyota Motor Corporation for Toyota Motor North America, Inc.. (Routh, Steven) (Entered: 07/10/2023)	
30	07/10/2023	ANSWER to 1 Complaint, by Denso Corporation.(Routh, Steven) (Entered: 07/10/2023)	
31	07/10/2023	Fed. R. Civ. P. 7.1(a)(1) Disclosure Statement filed by Denso Corporation identifying Corporate Parent Toyota Motor Corporation for Denso Corporation. (Routh, Steven) (Entered: 07/10/2023)	
32	07/10/2023	Joint MOTION to Sever and Stay Claims Against Toyota by Denso Corporation, Toyota Motor North America, Inc.. (Attachments: # 1 Proposed Order)(Routh, Steven) (Additional attachment(s) added on 7/11/2023: # 2 Revised Proposed Order) (nkl,). (Entered: 07/10/2023)	
	07/10/2023	In accordance with the provisions of 28 USC Section 636(c), you	

#	Date	Proceeding Text	Source
		are hereby notified that a U.S. Magistrate Judge of this district court is available to conduct any or all proceedings in this case including a jury or non-jury trial and to order the entry of a final judgment. The form Consent to Proceed Before Magistrate Judge is available on our website. All signed consent forms, excluding pro se parties, should be filed electronically using the event Notice Regarding Consent to Proceed Before Magistrate Judge. (nkl,) (Entered: 07/10/2023)	
33	07/12/2023	ORDER granting 32 Motion to Sever and Stay Claims Against Toyota. Signed by District Judge Rodney Gilstrap on 7/11/2023. (nkl,) (Entered: 07/12/2023)	
34	07/12/2023	NOTICE of Attorney Appearance by Melissa Richards Smith on behalf of Denso Corporation (Smith, Melissa) (Entered: 07/12/2023)	
35	07/19/2023	NOTICE of Attorney Appearance by Andrew Thompson (Tom) Gorham on behalf of Denso Corporation, Kia America, Inc., Kia Corporation (Gorham, Andrew) (Entered: 07/19/2023)	
	07/20/2023	Minute Entry for proceedings held before District Judge Rodney Gilstrap (Lead 2:23-cv-150; Members 2:23-cv-219; 2:23-cv-321_stayed): Scheduling Conference/Case Management Conference held on 7/20/2023. Counsel for the parties appeared. Court asked whether they consented to trial before the U.S. Magistrate Judge. Court then gave counsel Claim Construction and Jury Selection/Trial dates. (Court Reporter Shawn McRoberts) (aeb) (Entered: 07/24/2023)	
36	07/21/2023	NOTICE by Foras Technologies Ltd. of Compliance Regarding Disclosure of Asserted Claims and Infringement Contentions (Cooper, Brett) (Entered: 07/21/2023)	
37	07/24/2023	ANSWER to Complaint by Kia America, Inc., Kia Corporation.(Smith, Melissa) (Entered: 07/24/2023)	
38	07/24/2023	Fed. R. Civ. P. 7.1(a)(1) Disclosure Statement filed by Kia America, Inc., Kia Corporation (Smith, Melissa) (Entered: 07/24/2023)	
39	07/24/2023	DEMAND for Trial by Jury by Kia America, Inc., Kia Corporation. (Smith, Melissa) (Entered: 07/24/2023)	
	07/24/2023	In accordance with the provisions of 28 USC Section 636(c), you are hereby notified that a U.S. Magistrate Judge of this district court is available to conduct any or all proceedings in this case including a jury or non-jury trial and to order the entry of a final judgment. The form Consent to Proceed Before Magistrate Judge is available on our website. All signed consent forms, excluding pro se parties, should be filed electronically using the event Notice Regarding Consent to Proceed Before Magistrate Judge. (nkl,) (Entered: 07/24/2023)	
40	08/03/2023	Joint MOTION for Entry of Agreed Docket Control Order by Foras Technologies Ltd.. (Attachments: # 1 Proposed Order)(Cooper, Brett) (Entered: 08/03/2023)	
41	08/03/2023	Joint MOTION for Entry of Agreed Discovery Order by Foras Technologies Ltd.. (Attachments: # 1 Proposed Order)(Cooper, Brett) (Entered: 08/03/2023)	
42	08/09/2023	DOCKET CONTROL ORDER re 40 Joint MOTION for Entry of Agreed Docket Control Order filed by Foras Technologies Ltd.., Pretrial Conference set for 12/2/2024 at 09:00 AM before District Judge Rodney Gilstrap., Amended Pleadings due by 5/2/2024.. Jury Selection set for 1/6/2025 at 09:00AM before District Judge Rodney Gilstrap., Markman Hearing set for 7/18/2024 at 09:00 AM before District Judge Rodney Gilstrap., Motions in Limine due by 11/12/2024., Proposed Pretrial Order due by 11/25/2024. Signed by District Judge Rodney Gilstrap on 8/8/2023. (nkl,) (Entered:	

#	Date	Proceeding Text	Source
		08/09/2023)	
43	08/09/2023	DISCOVERY ORDER re 41 Motion for Entry of Agreed Discovery Order. Signed by District Judge Rodney Gilstrap on 8/8/2023. (nkl,) (Entered: 08/09/2023)	
44	08/09/2023	Joint MOTION to Amend Docket Control Order re 42 Order,,, Terminate Motions,,, Scheduling Order,, by Foras Technologies Ltd.. (Attachments: # 1 Proposed Order)(Cooper, Brett) (Entered: 08/09/2023)	
45	08/10/2023	NOTICE of Attorney Appearance by Sten Anker Jensen on behalf of Denso Corporation (Jensen, Sten) (Entered: 08/10/2023)	
46	08/11/2023	FIRST AMENDED DOCKET CONTROL ORDER re 44 Motion to Amend Docket Control Order. Signed by District Judge Rodney Gilstrap on 8/11/2023. (nkl,) (Entered: 08/11/2023)	
47	08/17/2023	Joint MOTION for Entry of Agreed Protective Order by Foras Technologies Ltd.. (Attachments: # 1 Proposed Order)(Cooper, Brett) (Entered: 08/17/2023)	
48	08/17/2023	NOTICE by Foras Technologies Ltd. of Compliance Regarding Initial Disclosures (Cooper, Brett) (Entered: 08/17/2023)	
49	08/17/2023	NOTICE of Discovery Disclosure by Denso Corporation (Routh, Steven) (Entered: 08/17/2023)	
50	08/18/2023	NOTICE by Kia America, Inc., Kia Corporation of Compliance Regarding Initial and Additional Disclosures (Smith, Melissa) (Entered: 08/18/2023)	
51	08/21/2023	PROTECTIVE ORDER. Signed by District Judge Rodney Gilstrap on 8/21/2023. (nkl,) (Entered: 08/21/2023)	
52	09/15/2023	NOTICE by Denso Corporation OF SERVICE OF DENSO CORPORATIONS INVALIDITY CONTENTIONS (Routh, Steven) (Entered: 09/15/2023)	
53	09/18/2023	NOTICE by Kia America, Inc., Kia Corporation of Compliance Regarding Invalidity and Subject Matter Eligibility Contentions (Smith, Melissa) (Entered: 09/18/2023)	
54	09/19/2023	NOTICE of Attorney Appearance by Jasmine Wenxi Zhu on behalf of Denso Corporation (Zhu, Jasmine) (Entered: 09/19/2023)	
55	12/06/2023	JOINT MOTION to Dismiss Denso Corporation and Toyota Motor North America, Inc. by Foras Technologies Ltd.. (Attachments: # 1 Exhibit Joint Stipulation of Dismissal, # 2 Proposed Order)(Cooper, Brett) (Entered: 12/06/2023)	
56	12/08/2023	ORDER re 55 JOINT MOTION to Dismiss Denso Corporation and Toyota Motor North America, Inc. filed by Foras Technologies Ltd.., Toyota Motor North America, Inc. (Consolidated Civil Action 2:23-321) terminated., Attorney Steven Routh terminated.. Signed by District Judge Rodney Gilstrap on 12/8/2023. (nkl,) (Entered: 12/08/2023)	
57	01/11/2024	OPPOSED MOTION to Stay UNDER CUSTOMER-SUIT EXCEPTION by Kia America, Inc., Kia Corporation. (Attachments: # 1 Affidavit/Declaration Cho Ae Kim, # 2 Affidavit/Declaration A. Greenfield, # 3 Exhibit 1, # 4 Exhibit 2, # 5 Exhibit 3, # 6 Exhibit 4, # 7 Exhibit 5, # 8 Exhibit 6, # 9 Proposed Order)(Smith, Melissa) (Entered: 01/11/2024)	
58	01/24/2024	UNOPPOSED MOTION for Extension of Time to File Response/Reply as to 57 OPPOSED MOTION to Stay UNDER CUSTOMER-SUIT EXCEPTION by Foras Technologies Ltd.. (Attachments: # 1 Proposed Order)(Cooper, Brett) (Entered: 01/24/2024)	
59	01/24/2024	UNOPPOSED MOTION to Amend/Correct Docket Control Order by Kia America, Inc., Kia Corporation. (Attachments: # 1 Proposed Order)(Smith, Melissa) (Entered: 01/24/2024)	

#	Date	Proceeding Text	Source
60	01/29/2024	ORDER granting 58 UNOPPOSED MOTION for Extension of Time to File Response/Reply as to 57 OPPOSED MOTION to Stay UNDER CUSTOMER-SUIT EXCEPTION , 57 OPPOSED MOTION to Stay UNDER CUSTOMER-SUIT EXCEPTION Responses due by 2/8/2024. Signed by District Judge Rodney Gilstrap on 1/29/2024. (ch,) (Entered: 01/30/2024)	
61	01/30/2024	SECOND AMENDED DOCKET CONTROL ORDER re 59 Motion to Amend/Correct Docket Control Order. Signed by District Judge Rodney Gilstrap on 1/29/2024. (nkl,) (Entered: 01/30/2024)	
62	02/07/2024	JOINT MOTION to Stay Claims Against Kia by Foras Technologies Ltd.. (Attachments: # 1 Proposed Order)(Cooper, Brett) (Entered: 02/07/2024)	
63	02/10/2024	ORDER granting 62 Motion to Stay case 2:23cv219. ORDERED that Member Case No. 2:23-cv-00219-JRG is STAYED pending the earlier of the resolution1 of the Eastern District of 1 This means the entry of final judgment and exhaustion of appeals. Virginia action (Case No. 2:23-cv-00647-EWH-RJK), or by further Order of this Court. Signed by District Judge Rodney Gilstrap on 2/10/2024. (ch,) (Entered: 02/12/2024)	
64	05/02/2024	JOINT MOTION to Dismiss Kia Corporation and Kia America, Inc. by Foras Technologies Ltd.. (Attachments: # 1 Exhibit Joint Stipulation of Dismissal, # 2 Proposed Order)(Cooper, Brett) (Entered: 05/02/2024)	
65	05/03/2024	ORDER granting 64 Motion to Dismiss. Signed by District Judge Rodney Gilstrap on 5/3/2024. (NKL) (Entered: 05/03/2024)	
66	05/03/2024	Notice of Filing of Patent/Trademark Form (AO 120) at termination of case. AO 120 emailed to the Director of the U.S. Patent and Trademark Office. (NKL) (Entered: 05/03/2024)	

Patents

Number	Title	Issued	Class	Subclass
7,502,958	System and method for providing firmware recoverable lockstep protection	03/10/2009	714	12
7,624,302	System and method for switching the role of boot processor to a spare processor responsive to detection of loss of lockstep in a boot processor	11/24/2009	714	12
7,627,781	System and method for establishing a spare processor for recovering from loss of lockstep in a boot processor	12/01/2009	714	13

Copyright © LexisNexis CourtLink, Inc. All Rights Reserved.
*** THIS DATA IS FOR INFORMATIONAL PURPOSES ONLY ***

End of Document

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
90/019,243	09/01/2023	7624302	185945.012600	1079

199393 7590 06/26/2024

Atlantic IP
c/o Lombard Geliebter LLP
1325 Avenue of the Americas
28th Floor
New York, NY 10019

EXAMINER	
TON, MY TRANG	

ART UNIT	PAPER NUMBER
3992	

MAIL DATE	DELIVERY MODE
06/26/2024	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

DO NOT USE IN PALM PRINTER

(THIRD PARTY REQUESTER'S CORRESPONDENCE ADDRESS)

GREENBERG TRAURIG (NJ)
500 CAMPUS DRIVE, SUITE 400
P.O. BOX 677
FLORHAM PARK NJ 07932

EX PARTE REEXAMINATION COMMUNICATION TRANSMITTAL FORM

REEXAMINATION CONTROL NO. *90/019,243* .

PATENT UNDER REEXAMINATION *7624302* .

ART UNIT *3992* .

Enclosed is a copy of the latest communication from the United States Patent and Trademark Office in the above identified *ex parte* reexamination proceeding (37 CFR 1.550(f)).

Where this copy is supplied after the reply by requester, 37 CFR 1.535, or the time for filing a reply has passed, no submission on behalf of the *ex parte* reexamination requester will be acknowledged or considered (37 CFR 1.550(g)).

Notice of Intent to Issue Ex Parte Reexamination Certificate	**Control No.** 90/019,243	**Patent Under Reexamination** 7624302	
	Examiner MY TRANG TON	**Art Unit** 3992	**AIA Status** No

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

1. ☑ Prosecution on the merits is (or remains) closed in this *ex parte* reexamination proceeding. This proceeding is subject to reopening at the initiative of the Office or upon petition. *Cf.* 37 CFR 1.313(a). A Certificate will be issued in view of
   (a) ☑ Patent owner's communication(s) filed: 05/23/2024.
   (b) ☐ Patent owner's failure to file an appropriate timely response to the Office action mailed: _____.
   (c) ☐ Patent owner's failure to timely file an Appeal Brief (37 CFR 41.31).
   (d) ☐ The decision on appeal by the ☐ Board of Patent Appeals and Interferences ☐ Court dated _____
   (e) ☐ Other: _____.
2. The Reexamination Certificate will indicate the following:
   (a) Change in the Specification: ☐ Yes ☑ No
   (b) Change in the Drawing(s): ☐ Yes ☑ No
   (c) Status of the Claim(s):
      (1) Patent claim(s) confirmed: 1,5-8,13,17,21-22 and 25-26.
      (2) Patent claim(s) amended (including dependent on amended claim(s)): _____
      (3) Patent claim(s) canceled: _____.
      (4) Newly presented claim(s) patentable: _____.
      (5) Newly presented canceled claims: _____.
      (6) Patent claim(s) ☐ previously ☐ currently disclaimed: _____
      (7) Patent claim(s) not subject to reexamination: 2-4,9-12,14-16,18-20,23-24 and 27.
3. ☐ A declaration(s)/affidavit(s) under **37 CFR 1.130(b)** was/were filed on _____.
4. ☑ Note the attached statement of reasons for patentability and/or confirmation. Any comments considered necessary by patent owner regarding reasons for patentability and/or confirmation must be submitted promptly to avoid processing delays. Such submission(s) should be labeled: "Comments On Statement of Reasons for Patentability and/or Confirmation."
5. ☐ Note attached NOTICE OF REFERENCES CITED (PTO-892).
6. ☐ Note attached LIST OF REFERENCES CITED (PTO/SB/08 or PTO/SB/08 substitute).
7. ☐ The drawing correction request filed on _____ is: ☐approved ☐disapproved.
8. ☐ Acknowledgment is made of the priority claim under 35 U.S.C. § 119(a)-(d) or (f).
   a) ☐ All b) ☐ Some* c) ☐None of the certified copies have
   ☐been received.
   ☐not been received.
   ☐been filed in Application No. _____.
   ☐been filed in reexamination Control No. _____.
   ☐been received by the International Bureau in PCT Application No. _____.

   * Certified copies not received: _____.
9. ☐ Note attached Examiner's Amendment.
10. ☐ Note attached Interview Summary (PTO-474).
11. ☐ Other: _____.

All correspondence relating to this reexamination proceeding should be directed to the **Central Reexamination Unit** at the mail, FAX, or hand-carry addresses given at the end of this Office action.

/MY TRANG TON/ Primary Examiner, Art Unit 3992	

cc: Requester (if third party requester)

NOTICE OF INTENT TO ISSUE EX PARTE REEXAMINATION CERTIFICATE (NIRC)

This action addresses *Ex Parte* reexamination of claims 1, 5-8, 13, 17, 21-22 and 25-26 of U.S. Patent No. 7,624,302 ("**302 Patent**"). The 302 Patent issued on November 24, 2009 to Michaelis et al, and was titled "SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR".

PRIORITY

After careful review of the prosecution history of the 302 Patent, Examiners find that the 302 Patent was filed on October 25, 2004 and assigned U.S. Patent Application No. 10/972,588 ("**588 Application**"). The Examiners also find that the 302 Patent does not claim foreign priority.

LITIGATION

After reviewing a Request, performing a litigation search and reviewing the file history of the 302 Patent, the Examiners are aware of the following related proceeding and litigations:

- (Open) 1:23cv640 Foras Technologies Ltd. V. Nissan Motor Company, Ltd. Et Al;

- (Closed) 2:23cv150 Foras Technologies Ltd. V. Toyota Motor North America, Inc. Et Al.

RELEVANT PROSECUTION HISTORY

The following is a brief outline of the relevant prosecution history:

09/01/2023 The Requester filed a request for ex parte reexamination of claims 1-27 of the 302 Patent ("**Request**"). The Request was assigned reexamination control number 90/019,243 ("**Instant Reexamination Proceeding**").

10/17/2023 The Office issued an order ("**Order**") granting reexamination of the requested claims 1-27. The Order stated that there was a substantial new question of patentability affecting claims 1-27 of the 302 Patent.

02/23/2024 The Office issued a Non-Final Office action ("**February 2024 NF**"). The February 2024 NF indicated that the Office's determination will be limited solely to a review of Ground 1, claims 1, 5-8, 13, 17, 21-22 and 25-26. Grounds 2-3 proposed including dependent claims 2-4, 9-12, 14- 16, 18-20, 23-24 and 27 were not accepted in the Order, thus, were not reexamined. In the February 2024 NF, claims 1, 5-8, 13 and 17 were rejected under pre-AIA 35 U.S.C. 103(a) as being unpatentable over

Kalyanasundharam in combination with Kennedy; and claims 21, 22, 25 and 26 were indicate allowable.

05/23/2024 Patent Owner's response was filed ("**May 2024 Response**").

The Examiners acknowledge the May 2024 Response. This action is made in full consideration of the May 2024 Response.

STATUS OF CLAIMS

Claims 1, 5-8, 13, 17, 21-22, 25-26 are original. Of these, claims 1, 13, 21 and 26 are independent claims.

CLAIM INTERPRETATION

During examination, claims are given the broadest reasonable interpretation consistent with the specification and limitations in the specification are not read into the claims. See MPEP §2111, MPEP §2111.01 and *In re Yamamoto et al.*, 222 USPQ 934 (Fed. Cir. 1984). Under a broadest reasonable interpretation, words of the claim must be given their plain meaning, unless such meaning is inconsistent with the specification. See MPEP §2111.01(I). It is further noted it is improper to import claim limitations from the specification, i.e., a particular embodiment appearing in the written description may not be read into a claim when the claim language is broader than the embodiment. See MPEP §2111.01(II). Therefore, unless one of the exceptions applies below, the Examiners will interpret the limitations of the pending and examined claims using the broadest reasonable interpretation.

A. Lexicographic Definitions

A first exception to the prohibition of reading limitations from the specification into the claims is when the Applicant for patent has provided a lexicographic definition for the term. See MPEP §2111.01(IV). Following an independent review of the claims in view of the specification herein, the Examiners find that Patent Owner has not provided any lexicographic definitions related to claim terms with any reasonable clarity, deliberateness and precision.

B. Claim Interpretation Under 35 U.S.C. §112(6th ¶)

A second exception to the prohibition of reading limitations from the specification into the claims is when the claimed feature is written as a means-plus-function or a step-plus-function. See MPEP §2181-2183. As noted in MPEP §2181, a three prong test is used to determine the scope of a means-plus-function or step-plus-function limitation in a claim:

(A) the claim limitation uses the term "means" or "step" or a term used as a substitute for "means" that is a generic placeholder (also called a nonce term or a non-structural term having no specific structural meaning) for performing the claimed function
(B) the term "means" or "step" or the generic placeholder is modified by functional language, typically, but not always linked by the transition word "for" (e.g., "means for") or another linking word or phrase, such as "configured to" or "so that"
(C) the term "means" or "step" or the generic placeholder is not modified by sufficient structure, material, or acts for performing the claimed function.

B1. FL #1: "system firmware means for detecting ..." (Claim 26)

A first means-plus-function phrase is recited in claim 26, which recites "system firmware means for detecting..." or hereinafter FL #1. The Examiners determine herein that FL #1 meets the three prong test and thus will be interpreted as a means-plus-function limitation under 35 U.S.C. §112 ¶6th.

The Examiners find that FL #1 in claim 26 recites:

"*system firmware means for detecting loss of lockstep (LOL) for a processor in a multi-processor system*".

(B1)(a) 3-Prong Analysis: Prong (A)

FL #1 meets invocation prong (A) because "means ... for" language is recited and thus there is a presumption that interpretation under §112 ¶6th is invoked.

(B1)(b) 3-Prong Analysis: Prong (B)

FL #1 meets invocation prong (B) because it recites the function "*for detecting loss of lockstep (LOL) for a processor in a multi-processor system*".

(B1)(c) 3-Prong Analysis: Prong (C)

FL #1 meets invocation prong (C) because FL #1 does *not* recite sufficient structure for performing the claimed function. Based upon a review of claim 26, the Examiners find that FL #1 recites very little structure for performing the function as set forth of FL #1.

In view of the Examiners findings above that FL #1 meets invocation prongs (A)-(C), the Examiners conclude FL #1 invokes interpretation under 35 U.S.C. §112 ¶6th.

(B1)(d) Corresponding Structure

After a claimed phrase has been shown to invoke 35 U.S.C. §112 ¶6th as found above, the next step is to determine the corresponding structure or material as described in the specification for performing the recited function. See MPEP §2181(II).

Based upon a review of the specification of the 302 Patent, the Examiners find that the closest corresponding disclosed structure for performing the claimed function is a master processor 12A, a slave processor 12B and an error detect 13C with supporting firmware (18A) including a Processor Abstraction Layer (201), a System Abstraction Layer (202) shown in FIG. 2 of the 302 Patent, and an Extended Firmware Interface disclose in col. 8, line 40 of the 302 Patent, reprinted below. The software modules for performing the recite functions in association with this hardware are described in Figs. 5-6 in the 302 Patent.

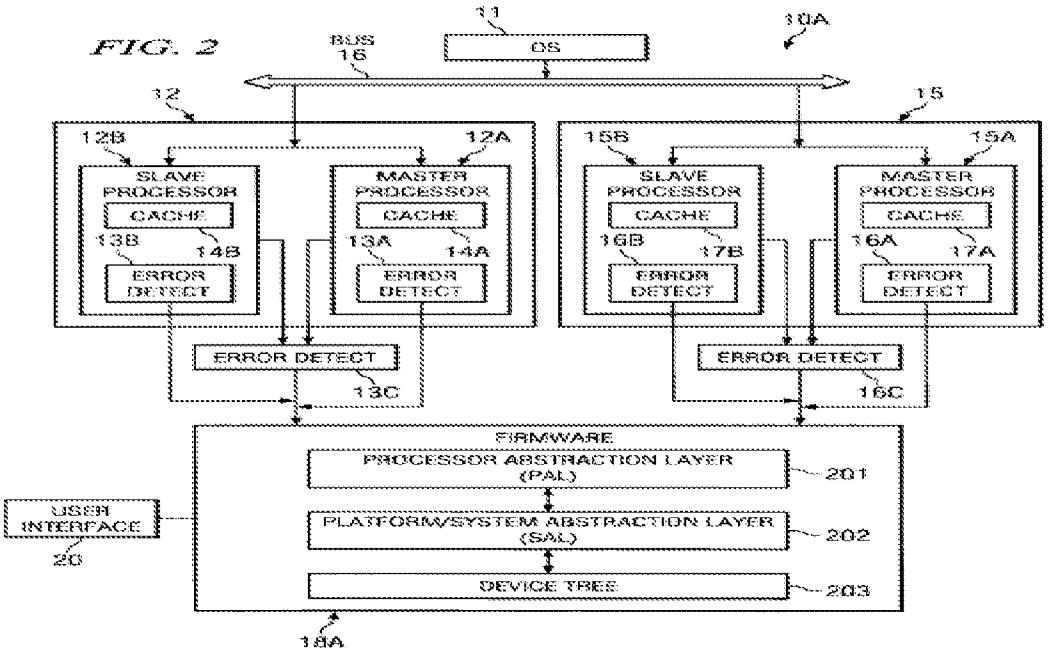
The 302 Patent states:

> *" In this example, master processor 12A includes error detect logic 13A, and slave processor 12B includes error detect logic 13B. While shown as included in each of the processors 12A and 12B in this example, in certain embodiments the error detect logic 13A and 13B may be implemented external to processors 12A and 12B. Error detect logic 13A and 13B*

include logic for detecting errors, such as data cache errors, present in their respective processors 12A and 12B. Examples of error detect logic 13A and 13B include known parity-based mechanisms and ECC mechanisms. Error detect logic 13C is also included, which may include an XOR (exclusive OR) gate, for detecting a lockstep mismatch between master processor 12A and slave processor 12B. As mentioned above, a lockstep mismatch refers to the output of master processor 12A and slave processor 12B failing to match. While shown as external to the lockstep processor pair 12 in this example, in certain embodiments error detect logic 13C may be implemented on a common silicon chip with processors 12A and 12B. Lockstep mismatch is one way of detecting a LOL between the master processor 12A and slave processor 12B. A detection of an error by either of error detect logic 13A and 13B also provides detection of LOL in the processors 12A and 12B. Because the detection of LOL by error detect logic 13A and 13B may occur before an actual lockstep mismatch occurs, the detection of LOL by error detect logic 13A and 13B may be referred to as a detection of a "precursor to lockstep mismatch". In other words, once an error (e.g., corrupt data) is detected by error detect logic 13A or 13B, such error may eventually propagate to a lockstep mismatch error that is detectable by error detect logic 13C." (col. 7:6-36)

"Processor architecture generally comprises corresponding supporting firmware, such as firmware 18 of system 10. For example, as described further below in conjunction with the specific example of FIG. 2, the IPF processor architecture comprises such Supporting firmware as Processor Abstraction Layer (PAL), System Abstraction Layer (SAL), and Extended Firmware Interface (EFI). Such supporting firmware may enable, for example, the OS to access a particular function implemented for the processor... Accordingly, the Supporting firmware. Such as the PAL, is implemented to provide an interface to the processor(s) for accessing the functionality provided by such processor(s). Each of those interfaces

provide standard, published procedure calls that are Supported." (col. 8:34-59)



302 Patent FIG. 2

In view of these findings, pursuant to 35 U.S.C. §112 ¶ 6th, FL #1 in claim 26 will be limited to the corresponding structures discussed above in conjunction with the appropriate software modules or programming noted above for detecting loss of lockstep (LOL) for a processor in the multi-processor system.

If applicant does not intend to have this limitation, FL #1 in claim 26 interpreted under 35 U.S.C. §112 ¶6th applicant may: (1) amend the claim limitation(s) to avoid it/them being interpreted under 35 U.S.C. §112 ¶6th (e.g., by reciting sufficient structure to perform the claimed function); or (2) present

a sufficient showing that the claim limitation recites sufficient structure to perform the claimed function so as to avoid it/them being interpreted under 35 U.S.C. §112 ¶6th.

B2. FL #2: "system firmware means for determining ..." (Claim 26)

A second means-plus-function phrase is recited in claim 26, which recites "system firmware means for determining ..." or hereinafter FL #2. The Examiners determine herein that FL #2 meets the three prong test and thus will be interpreted as a means-plus-function limitation under 35 U.S.C. §112 ¶6th.

The Examiners find that FL #2 in claim 26 recites:

"*system firmware means for determining whether the processor for which said LOL is detected is assigned the role of boot processor*".

(B2)(a) 3-Prong Analysis: Prong (A)

FL #2 meets invocation prong (A) because "means ... for" language is recited and thus there is a presumption that interpretation under §112 ¶6th is invoked.

(B2)(b) 3-Prong Analysis: Prong (B)

FL #2 meets invocation prong (B) because it recites the function "*for determining whether the processor for which said LOL is detected is assigned the role of boot processor*".

(B2)(c) 3-Prong Analysis: Prong (C)

FL #2 meets invocation prong (C) because FL #2 does *not* recite sufficient structure for performing the claimed function. Based upon a review of claim 26, the Examiners find that FL #2 recites very little structure for performing the function as set forth of FL #2.

In view of the Examiners findings above that FL #2 meets invocation prongs (A)-(C), the Examiners conclude FL #2 invokes interpretation under 35 U.S.C. §112 ¶6th.

(B2)(d) Corresponding Structure

After a claimed phrase has been shown to invoke 35 U.S.C. §112 ¶6th as found above, the next step is to determine the corresponding structure or material as described in the specification for performing the recited function. See MPEP §2181(II).

Based upon a review of the specification of the 302 Patent, the Examiners find that the closest corresponding disclosed structure for performing the claimed function is a device tree (203) in the firmware 18A, a Boot Processor 12 and a Spare Processor 15 for accessing the processor designators in the device tree (203) is shown in FIG. 2 of the 302 Patent, reprinted below. The software modules for performing the recite functions in association with this hardware are described in Figs. 5-6 in the 302 Patent. The 302 Patent states:

"*As described further herein, in certain embodiments, a device tree is stored to a data storage device (e.g., SRAM) such that it is accessible to the firmware 18, wherein the device tree includes information identifying processor 12 as the system's boot processor and identifying processor 15 as the spare. The information specifying the boot processor and/or square may be received as input from a user in certain embodiment.*" (col. 9:4-11)

"*Further, device tree 203 includes information designating which processor is the boot processor, as well as a spare processor for the boot processor.*" (col. 13:21-23)

The 302 Patent further discloses:

"*In operational : block 501, the system firmware, responsive to such detection of LOL for the boot processor, examines the device tree to determine which processor module is assigned the role of active spare. In operational block 502 the system firmware (e.g., SAL) asserts a General Purpose Event Interrupt (GPE) by writing to the appropriate GPE register in the ACPI register space. This causes the OS to execute its interrupt handler, which will run an AML Method associated with this particular interrupt. In this example, this AML method that is executed responsive to the interrupt generated by the GPE includes an AML instruction called "Notify". This "Notify" operation indicates the determined active spare processor object and that an Eject is being requested as arguments. Accordingly, an AML method is used to request the OS to eject the active spare that was determined in block 501. In response, the OS idles the active spare processor and returns control over the active spare processor to the system firmware, in block 503.*" (col. 17:15-33)

"In operational block 601, LOL is detected for a processor in a multi-processor system. In operational block 602, the system (e.g., system firmware) determines whether the processor for which LOL is detected is the system's boot processor. As described above, this is determined in certain embodiments by accessing the node of the device tree corresponding to the processor for which LOL is detected and determining if the information in that node of the device tree indicates that the processor is the system's boot processor. (col. 18: 50-58)

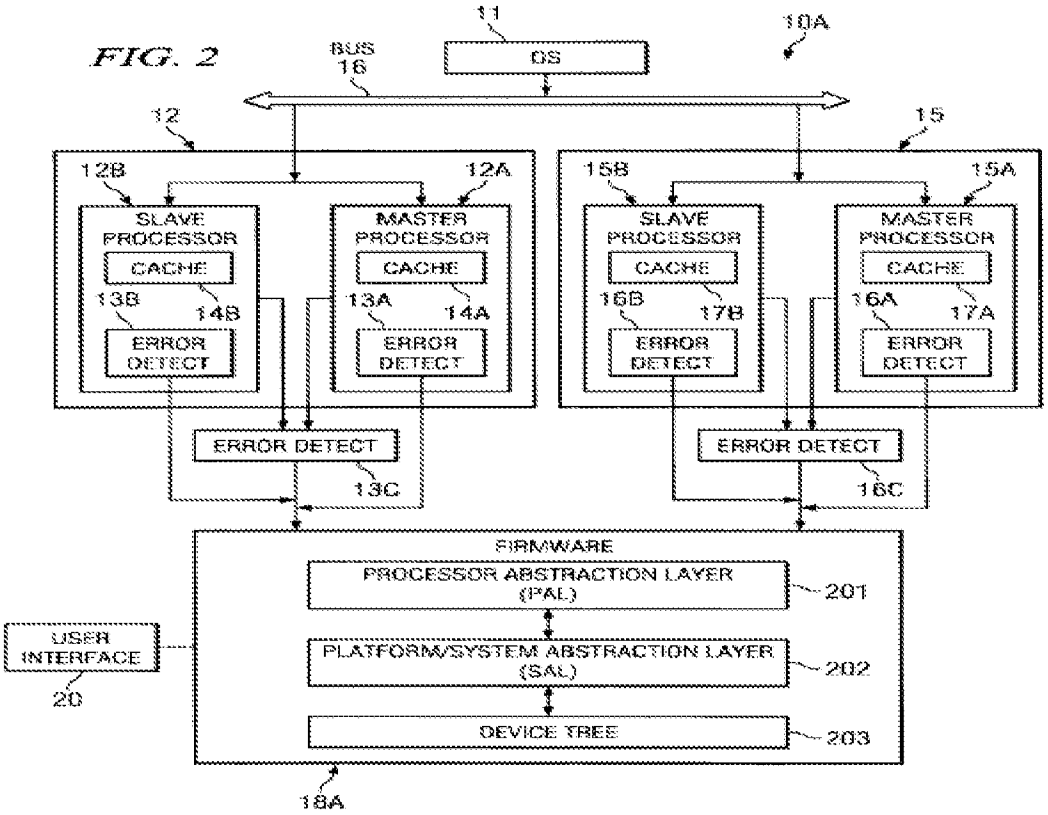
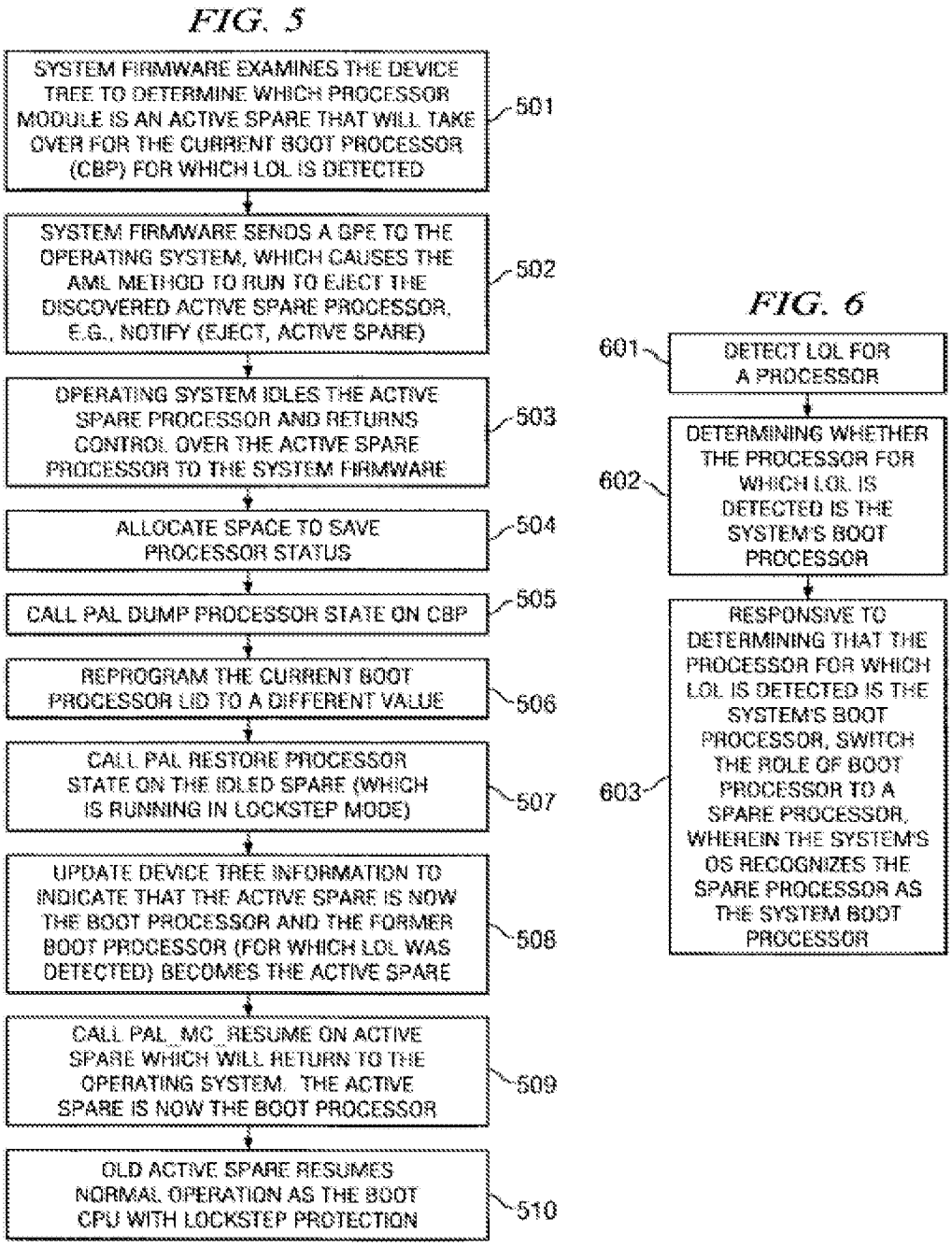


Fig. 2 of the 302 Patent



Figs. 5-6 of the 302 Patent

In view of these findings, pursuant to 35 U.S.C. §112 ¶ 6th, FL #2 in claim 26 will be limited to the corresponding structures discussed above in

conjunction with the appropriate software modules or programming noted above for determining whether the processor for which the LOL is detected and assigned the role of boot processor.

If applicant does not intend to have this limitation, FL #2 in claim 26 interpreted under 35 U.S.C. §112 ¶6th applicant may: (1) amend the claim limitation(s) to avoid it/them being interpreted under 35 U.S.C. §112 ¶6th (e.g., by reciting sufficient structure to perform the claimed function); or (2) present a sufficient showing that the claim limitation recites sufficient structure to perform the claimed function so as to avoid it/them being interpreted under 35 U.S.C. §112 ¶6th.

B3. FL #3: "system firmware means for switching..." (Claim 26)

A third means-plus-function phrase is recited in claim 26, which recites "system firmware means for switching..." or hereinafter FL #3. The Examiners determine herein that FL #3 meets the three prong test and thus will be interpreted as a means-plus-function limitation under 35 U.S.C. §112 ¶6th.

The Examiners find that FL #3 in claim 26 recites:

"*system firmware means for switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which said LOL is detected is assigned the role of boot processor.*"

(B3)(a) 3-Prong Analysis: Prong (A)

FL #3 meets invocation prong (A) because "means ... for" language is recited and thus there is a presumption that interpretation under §112 ¶6th is invoked.

(B3)(b) 3-Prong Analysis: Prong (B)

FL #3 meets invocation prong (B) because it recites the function "*for switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which said LOL is detected is assigned the role of boot processor.*"

(B3)(c) 3-Prong Analysis: Prong (C)

FL #3 meets invocation prong (C) because FL #3 does *not* recite sufficient structure for performing the claimed function. Based upon a review of claim 26, the Examiners find that FL #3 recites very little structure for performing the function as set forth of FL #3.

In view of the Examiners findings above that FL #3 meets invocation prongs (A)-(C), the Examiners conclude FL #3 invokes interpretation under 35 U.S.C. §112 ¶6th.

(B3)(d) Corresponding Structure

After a claimed phrase has been shown to invoke 35 U.S.C. §112 ¶6th as found above, the next step is to determine the corresponding structure or

material as described in the specification for performing the recited function. See MPEP §2181(II).

Based upon a review of the specification of the 302 Patent, the Examiners find that the corresponding disclosed structure for performing the claimed function is the Boot Processor (12), the Spare Processor (15), the error detects 13C, 16C and supporting firmware 18A including a Processor Abstraction Layer (201), a System Abstraction Layer (202) shown in FIG. 2 of the 302 Patent, and an Extended Firmware Interface discloses in col. 8, line 40 of the 302 Patent, reprinted below. The software modules for performing the recite functions in association with the hardware are described in FIGS. 1 and 4-5 in the 302 Patent.

The 302 Patent states:

"Processor architecture generally comprises corresponding supporting firmware, such as firmware 18 of system 10. For example, as described further below in conjunction with the specific example of FIG. 2, the IPF processor architecture comprises such supporting firmware as Processor Abstraction Layer (PAL), System Abstraction Layer (SAL), and Extended Firmware Interface (EFI). Such supporting firmware may enable, for example, the OS to access a particular function implemented for the processor. For instance, the OS may query the PAL as to the size of the cache implemented for the processor, etc. Other well-known functions provided by the supporting firmware (SAL, EFT) include, for example: (a) performing I/O configuration accesses to discover and program the I/O Hardware (SAL_PCILCONFIG_READ and SAL_PCI_CONFIG_WRITE); (b) retrieving error log data from the platform following a Machine Check Abort (MCA) event (SAL_GET_STATE_INFO); (c) accessing persistent store configuration data stored in non-volatile memory (EFI variable services:

GetNextVariableName, GetVariable and SetVariable); and accessing the battery-backed real-time clock/calendar (EFI GetTime and SetTime). Accordingly, the supporting firmware, such as the PAL, is implemented to provide an interface to the processor(s) for accessing the functionality provided by such processor(s). Each of those interfaces provide standard, published procedure calls that are supported." (col. 8:34-59)

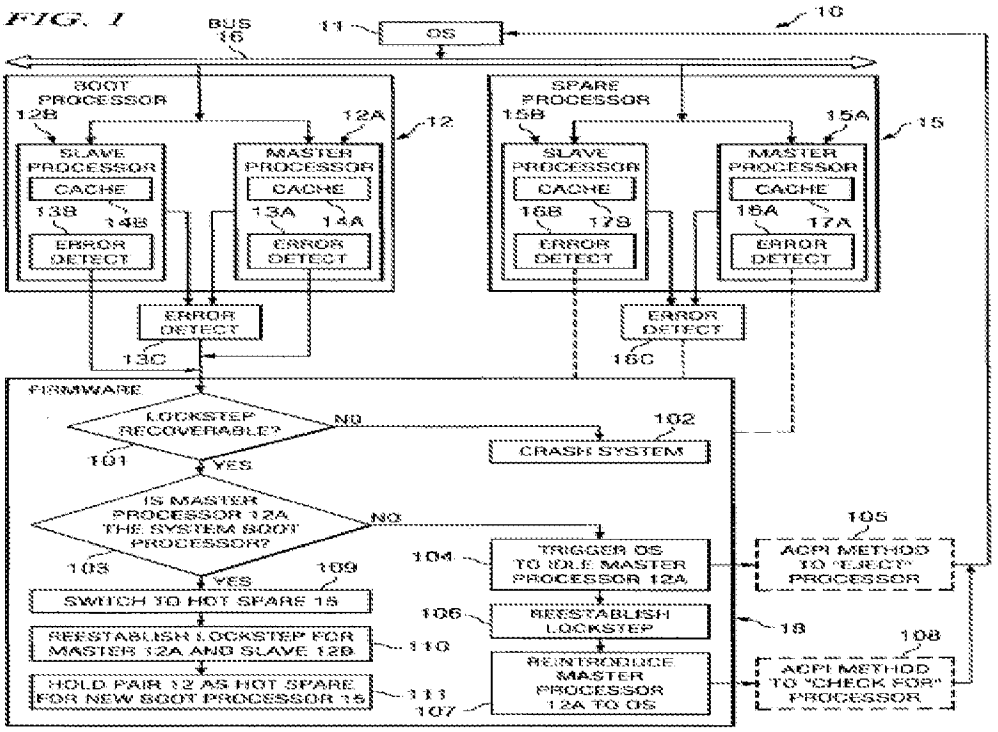
"For instance, in the example of FIG. 1, if determined in block 103 that the master processor 12A is the system's boot processor, operation advances to block 109 whereat the state of the "good" processor in the lockstep processor pair 12 is copied over to each of processors 15A and 15B in the spare lockstep processor pair 15. Exemplary techniques that may be utilized by firmware 18 for switching the role of boot processor to the hot spare are described further herein below." (col. 11:11-18)

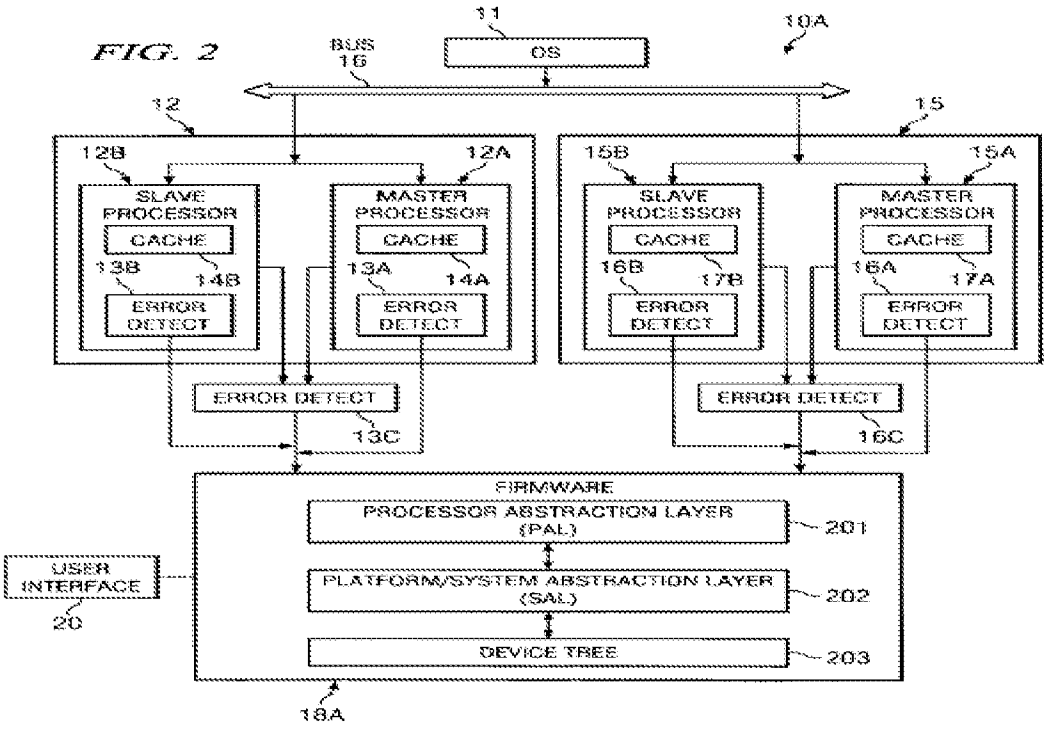
"In operational block 403, the LID of the current boot processor (for which LOL was detected) is reprogrammed to a different value. For instance, the current boot processor may be assigned a LID that is reserved just for this situation (i.e., for switching the role of boot processor). Accordingly, the boot processor for which LOL was detected will not respond to the interrupts addressed for its original LID because the boot processor's LID value is reprogrammed to a different value." (col. 16, 27-35)

"Turning to FIG. 5, an exemplary operational flow diagram for switching the role of boot processor to an active spare processor according to one embodiment is shown. Again, an LOL has been detected for the boot processor. In operational block 501, the system firmware, responsive to such detection of LOL for the boot processor, examines the device tree to determine which processor module is assigned the role of active spare. In operational block 502 the system firmware (e.g., SAL) asserts a General Purpose Event Interrupt (GPE) by writing to the appropriate GPE register in

the ACPI register space. This causes the OS to execute its interrupt handler, which will run. an AML Method associated with this particular interrupt. In this example, this AML method that is executed responsive to the interrupt generated by the GPE includes an AML instruction called "Notify". This "Notify" operation indicates the determined active spare processor object and that an Eject is being requested as arguments. Accordingly, an AML method is used to request the OS to eject the active spare that was determined in block 501. In response, the OS idles the active spare processor and returns control over the active spare processor to the system firmware, in block 503." (col. 17:12-33)

"In operational block 603, responsive to determining that the processor for which LOL is detected is the system's boot processor, the system's firmware switches the role of boot processor to a spare processor, which may be either a hot spare or an active spare, as described above. Accordingly, the system's OS recognizes the spare processor as the system boot processor. In view of the above, embodiments hereof enable the role of boot processor to be dynamically switched, during system runtime, to a spare processor responsive to a detected LOL for the boot processor, thus enabling recovery from the LOL without requiring that the system be crashed or re-booted." (col. 18:59- col. 19:3)





FIGs. 1-2 of the 302 Patent.

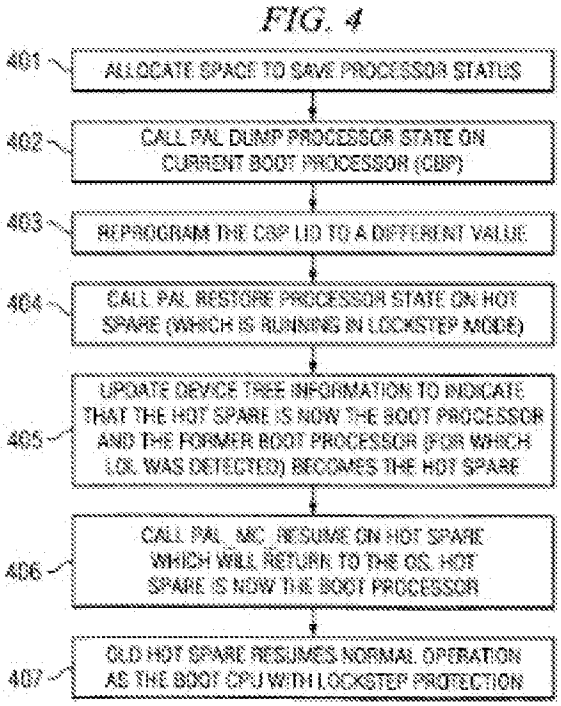


Fig. 4 of the 302 Patent

FIG. 5

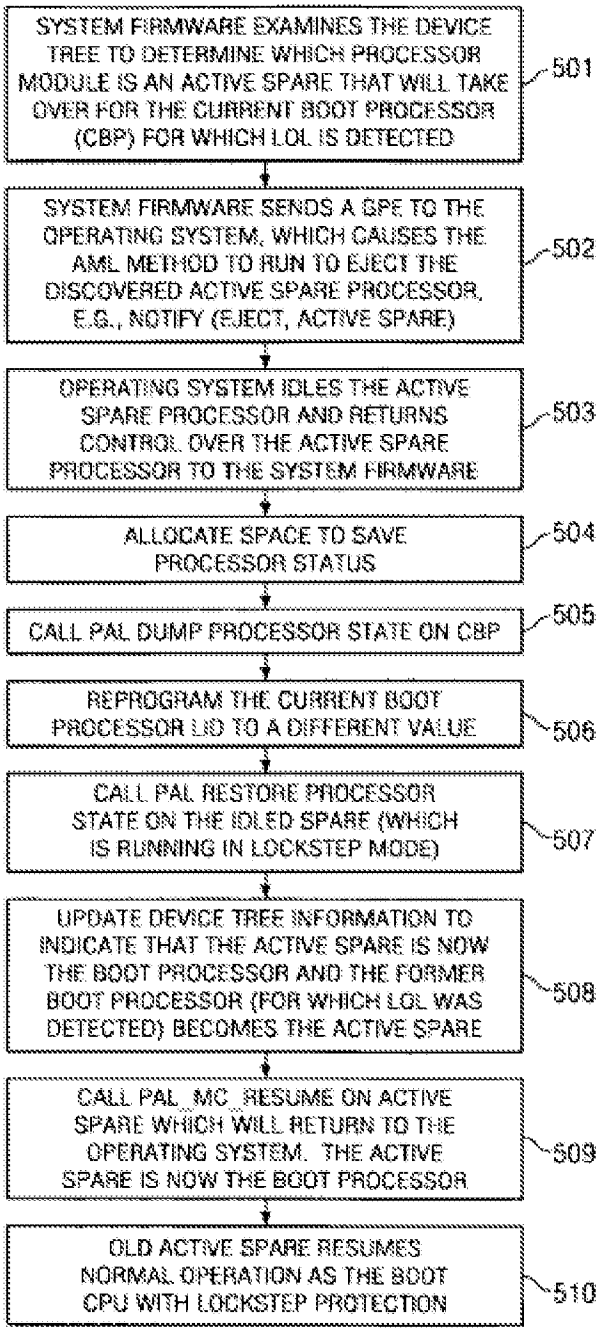
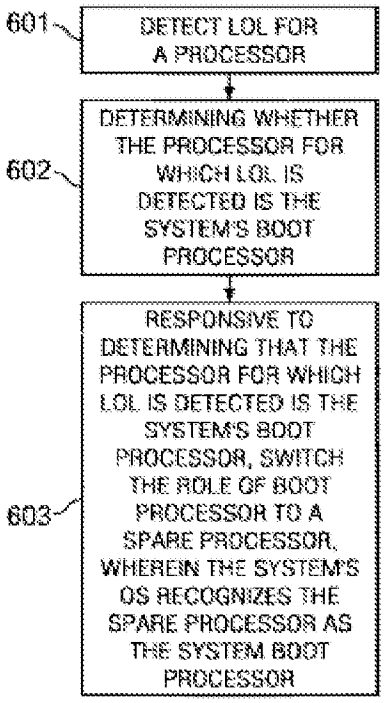


FIG. 6



Figs. 5-6 of the 302 Patent

In view of these findings, pursuant to 35 U.S.C. §112 ¶ 6th, FL #3 in claim 26 will be limited to the corresponding structures discussed above in conjunction with the appropriate software modules or programming noted above for switching the role of boot processor to the spare processor without shutting down the system's operating system if determined that the processor for which the LOL is detected and assigned the role of boot processor.

If applicant does not intend to have this limitation, FL #3 in claim 26 interpreted under 35 U.S.C. §112 ¶6th applicant may: (1) amend the claim limitation(s) to avoid it/them being interpreted under 35 U.S.C. §112 ¶6th (e.g., by reciting sufficient structure to perform the claimed function); or (2) present a sufficient showing that the claim limitation recites sufficient structure to perform the claimed function so as to avoid it/them being interpreted under 35 U.S.C. §112 ¶6th.

Furthermore, Examiners find herein that claim 21 include one or more claim limitations that do not use the word "means," but are nonetheless being interpreted under 35 U.S.C. §112 ¶6th because the claim limitations uses a generic placeholder that is coupled with functional language without reciting sufficient structure to perform the recited function and the generic placeholder is not preceded by a structural modifier. Each such limitation will be discussed in turn as follows:

B4. FL #4: "error detection logic" (Claim 21)

A fourth means-plus-function phrase is recited in claim 21 (and dependent claims 22 and 25), which recites "error detection logic" or hereinafter FL #4. The Examiners determine herein that FL #4 meets the three prong test and thus will be interpreted as a means-plus-function limitation under 35 U.S.C. §112 ¶6th.

The Examiners find that FL #4 in claim 21 recites:

"*error detection logic operable to detect loss of lockstep (LOL) for at least one of said processor modules*".

(B4)(a) 3-Prong Analysis: Prong (A)

FL #4 meets invocation prong (A) because "means ... for" language is recited. The Examiners find that "error detection logic" as used in claim 21 is a generic placeholder or nonce term equivalent to "means" because the phrase does not convey any particular sufficient structure for performing the recited functions.

Furthermore, there is no disclosure or suggestion from the prior art or the 302 Patent that a "error detection logic" is sufficient structure to perform the functions recited in FL #4. Rather, the prior art uses the phrase "error detection logic" in association with numerous structures and perform numerous functions which are distinct from those functions in FL #4. For example, U.S. Patent No. 7,370,243, claim 15 recites "error detection logic" comprising various means and functions; U.S. Publication No. 2006/0020850

claim 1 recites "*error detection logic configured to scan the memory regions for a latent processing error maintained in a memory region as an erroneous data output from a processor element*"; U.S. Patent No. 6,882,620, claim 2 recites "*error detection logic configured to monitor at least one of the plurality of clients for an anomaly in the operation thereof*". The Examiners further find these error detection logics have different structures and functions than those recited in FL #4. Accordingly, based on the diverse functional uses of "error detection logic" which would have diverse structures to perform these functions, the Examiners find that "error detection logic" does not imply or require a specific structure across these references and FL #4.

Accordingly, the Examiners find nothing in the specification, prosecution history or the prior art to construe "error detection logic" in FL #4 as the name of a sufficiently definite structure for performing the functions recited in FL #4 so as to take the overall claim limitation out of the ambit of §112(f). *See Williamson v. Citrix Online, L.L.C.*, 115 USPQ2d 1105, 1112 (Fed. Cir. 2015).

In light of the above findings, the Examiners conclude that the term "error detection logic" is a generic placeholder having no specific structure associated therewith. Because "error detection logic" is merely a generic placeholder having no specific structure associated therewith, the Examiners conclude that FL #4 meets invocation Prong (A).

(B4)(b) 3-Prong Analysis: Prong (B)

FL #4 meets invocation prong (B) because it recites the function "*operable to detect loss of lockstep (LOL) for at least one of said processor modules.*"

(B4)(c) 3-Prong Analysis: Prong (C)

FL #4 meets invocation prong (C) because FL #4 does *not* recite sufficient structure for performing the claimed function. Based upon a review of claim 21, the Examiners find that FL #4 recites very little structure, if any, for performing the function as set forth of FL #4.

In view of the Examiners findings above that FL #4 meets invocation prongs (A)-(C), the Examiners conclude FL #4 invokes interpretation under 35 U.S.C. §112 ¶6th.

(B4)(d) Corresponding Structure

After a claimed phrase has been shown to invoke 35 U.S.C. §112 ¶6th as found above, the next step is to determine the corresponding structure or material as described in the specification for performing the recited function. See MPEP §2181(II).

Based upon a review of the specification of the 302 Patent, the Examiners find that the corresponding hardware structure for the FL#4, i.e. the error detection logic is the error detection logics 13A-13C, 16A-16C shown in Fig. 1 of the 302 Patent.

The 302 Patent discloses at col. 7:6-44:

"*In this example, master processor 12A includes error detect logic 13A, and slave processor 12B includes error detect logic 13B. While shown as included in each of the processors 12A and 12B in this example, in certain embodiments the error detect logic 13A and 13B may be implemented external to processors 12A and 12B. Error detect logic 13A and 13B include logic for detecting errors, such as data cache errors, present in their respective processors 12A and 12B. Examples of error detect logic 13A and 13B include known parity-based mechanisms and ECC mechanisms. Error detect logic 13C is also included, which may include an XOR (exclusive OR) gate, for detecting a lockstep mismatch between master processor 12A and slave processor 12B. As mentioned above, a lockstep mismatch refers to the output of master processor 12A and slave processor 12B failing to match. While shown as external to the lockstep processor pair 12 in this example, in certain embodiments error detect logic 13C may be implemented on a common silicon chip with processors 12A and 12B.*

Lockstep mismatch is one way of detecting a LOL between the master processor 12A and slave processor 12B. A detection of an error by either of error detect logic 13A and 13B also provides detection of LOL in the processors 12A and 12B. Because the detection of LOL by error detect logic 13A and 13B may occur before an actual lockstep mismatch occurs, the detection of LOL by error detect logic 13A and 13B may be referred to as a detection of a "precursor to lockstep mismatch". In other words, once an error (e.g., corrupt data) is detected by error detect logic 13A or 13B, such error may eventually propagate to a lockstep mismatch error that is detectable by error detect logic 13C.

System 10 further includes a second lockstep processor pair 15 that is communicatively coupled to bus 16. Lockstep processor pair 15 includes master processor 15A and slave processor 15B. Master processor 15A includes cache 17A, and slave processor 15B includes cache 17B. As with error detect logic 13A-13C implemented for lockstep processor pair 12, lockstep processor pair 15 has error detect logic 16A-16C."

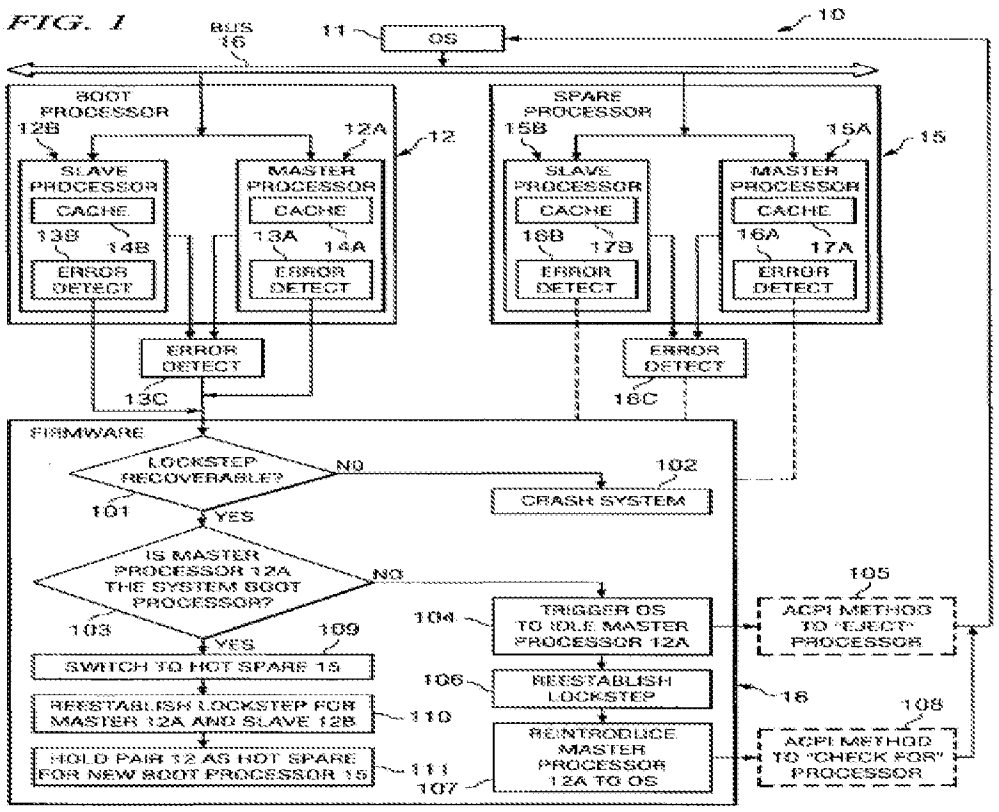


Fig. 1 of 302 Patent

In view of these findings, pursuant to 35 U.S.C. §112 ¶ 6th, FL #4 in claim 21 will be limited to the corresponding structures discussed above.

If applicant does not intend to have this limitation, FL #4 in claim 21 interpreted under 35 U.S.C. §112 ¶6th applicant may: (1) amend the claim limitation(s) to avoid it/them being interpreted under 35 U.S.C. §112 ¶6th (e.g., by reciting sufficient structure to perform the claimed function); or (2) present a sufficient showing that the claim limitation recites sufficient structure to perform the claimed function so as to avoid it/them being interpreted under 35 U.S.C. §112 ¶6th.

B5. FL #5: "system firmware" (Claim 21)

A fifth means-plus-function phrase is recited in claim 21, which recites "system firmware" or hereinafter FL #5. The Examiners determine herein that FL #5 meets the three prong test and thus will be interpreted as a means-plus-function limitation under 35 U.S.C. §112 ¶6th.

The Examiners find that FL #5 in claim 21 recites:

"*system firmware operable, responsive to the detection of LOL for a first of said processor modules, to determine whether said first processor module is assigned a role of system boot processor; wherein if determined that said first processor module is assigned the role of system boot processor, said system firmware is further operable to cause said operating system to recognize another processor module as system boot processor without shutting down said operating system.*"

(B5)(a) 3-Prong Analysis: Prong (A)

FL #5 meets invocation prong (A) because "means ... for" language is recited. The Examiners find that "system firmware" as used in claim 21 is a generic placeholder or nonce term equivalent to "means" because the phrase does not convey any particular sufficient structure for performing the recited functions.

Furthermore, there is no disclosure or suggestion from the prior art or the 302 Patent that the "system firmware" is sufficient structure to perform the functions recited in FL #5. Rather, the prior art uses the phrase "system firmware" in association with numerous structures and perform numerous

functions which are distinct from those functions in FL #5. For example, U.S. Patent No. 5,524,245 discloses in col. 1:29-34 "*While manufacturers have designed components such as the CPU and I/O controllers that can be configured to operate in either byte-order mode, firmware is normally one mode or the other. Thus, the configurable components are normally statically hard-wired by the manufacturers to operate in the same byte order as the computer firmware*"; U.S. Patent No. 4,413,322 discloses in col. 12:56-66 "Firmware stored in memory 34" comprising various functions. The Examiners further find these firmwares have different structures and functions than those recited in FL #5. Accordingly, based on the diverse functional uses of "system firmware" which would have diverse structures to perform these functions, the Examiners find that "system firmware" does not imply or require a specific structure across these references and FL #5.

Accordingly, the Examiners find nothing in the specification, prosecution history or the prior art to construe "system firmware" in FL #5 as the name of a sufficiently definite structure for performing the functions recited in FL #5 so as to take the overall claim limitation out of the ambit of §112(f). *See Williamson v. Citrix Online, L.L.C.*, 115 USPQ2d 1105, 1112 (Fed. Cir. 2015).

In light of the above findings, the Examiners conclude that the term "system firmware" is a generic placeholder having no specific structure associated therewith. Because "system firmware" is merely a generic placeholder having no specific structure associated therewith, the Examiners conclude that FL #5 meets invocation Prong (A).

(B5)(b) 3-Prong Analysis: Prong (B)

FL #5 meets invocation prong (B) because it recites the function: "*operable, responsive to the detection of LOL for a first of said processor modules, to determine whether said first processor module is assigned a role of system boot processor; wherein if determined that said first processor module is assigned the role of system boot processor, said system firmware is further operable to cause said operating system to recognize another processor module as system boot processor without shutting down said operating system.*"

(B5)(c) 3-Prong Analysis: Prong (C)

FL #5 meets invocation prong (C) because FL #5 does *not* recite sufficient structure for performing the claimed function. Based upon a review of claim 21, the Examiners find that FL #5 recites very little structure for performing the function as set forth of FL #5.

In view of the Examiners findings above that FL #5 meets invocation prongs (A)-(C), the Examiners conclude FL #5 invokes interpretation under 35 U.S.C. §112 ¶6th.

(B5)(d) Corresponding Structure

After a claimed phrase has been shown to invoke 35 U.S.C. §112 ¶6th as found above, the next step is to determine the corresponding structure or

material as described in the specification for performing the recited function. See MPEP §2181(II).

Based upon a review of the specification of the 302 Patent, the Examiners find that the corresponding disclosed structure for performing the claimed function is the firmware 18A including a Processor Abstraction Layer (201), a System Abstraction Layer (202), a device tree (203) shown in FIG. 2 of the 302 Patent, reprinted below, and an Extended Firmware Interface discloses in col. 8, line 40 of the 302 Patent,. The software modules for performing the recite functions in association with the hardware are described in FIGS. 1 and 4-5 in the 302 Patent.

The 302 Patent states:

"Processor architecture generally comprises corresponding supporting firmware, such as firmware 18 of system 10. For example, as described further below in conjunction with the specific example of FIG. 2, the IPF processor architecture comprises such supporting firmware as Processor Abstraction Layer (PAL), System Abstraction Layer (SAL), and Extended Firmware Interface (EFI). Such supporting firmware may enable, for example, the OS to access a particular function implemented for the processor. For instance, the OS may query the PAL as to the size of the cache implemented for the processor, etc. Other well-known functions provided by the supporting firmware (SAL, EFT) include, for example: (a) performing I/O configuration accesses to discover and program the I/O Hardware (SAL_PCILCONFIG_READ and SAL_PCI_CONFIG_WRITE); (b) retrieving error log data from the platform following a Machine Check Abort (MCA) event (SAL_GET_STATE_INFO); (c) accessing persistent store configuration data stored in non-volatile memory (EFI variable services: GetNextVariableName, GetVariable and SetVariable); and accessing the

battery-backed real-time clock/calendar (EFI GetTime and SetTime). Accordingly, the supporting firmware, such as the PAL, is implemented to provide an interface to the processor(s) for accessing the functionality provided by such processor(s). Each of those interfaces provide standard, published procedure calls that are supported." (col. 8:34-59)

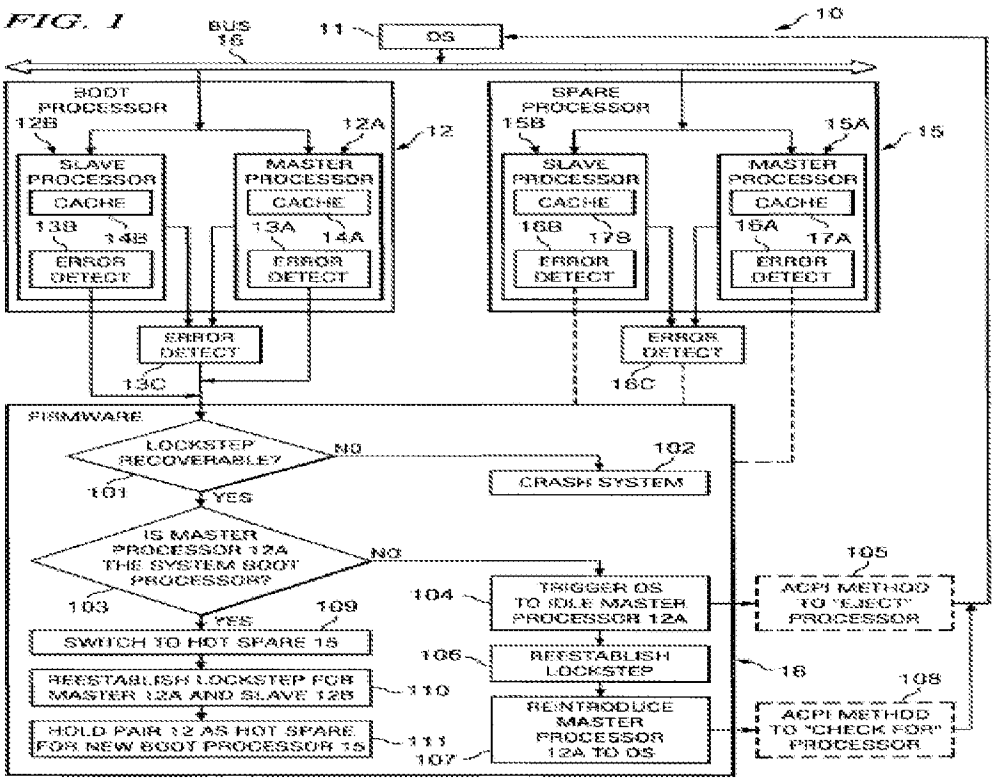
"For instance, in the example of FIG. 1, if determined in block 103 that the master processor 12A is the system's boot processor, operation advances to block 109 whereat the state of the "good" processor in the lockstep processor pair 12 is copied over to each of processors 15A and 15B in the spare lockstep processor pair 15. Exemplary techniques that may be utilized by firmware 18 for switching the role of boot processor to the hot spare are described further herein below." (col. 11:11-18)

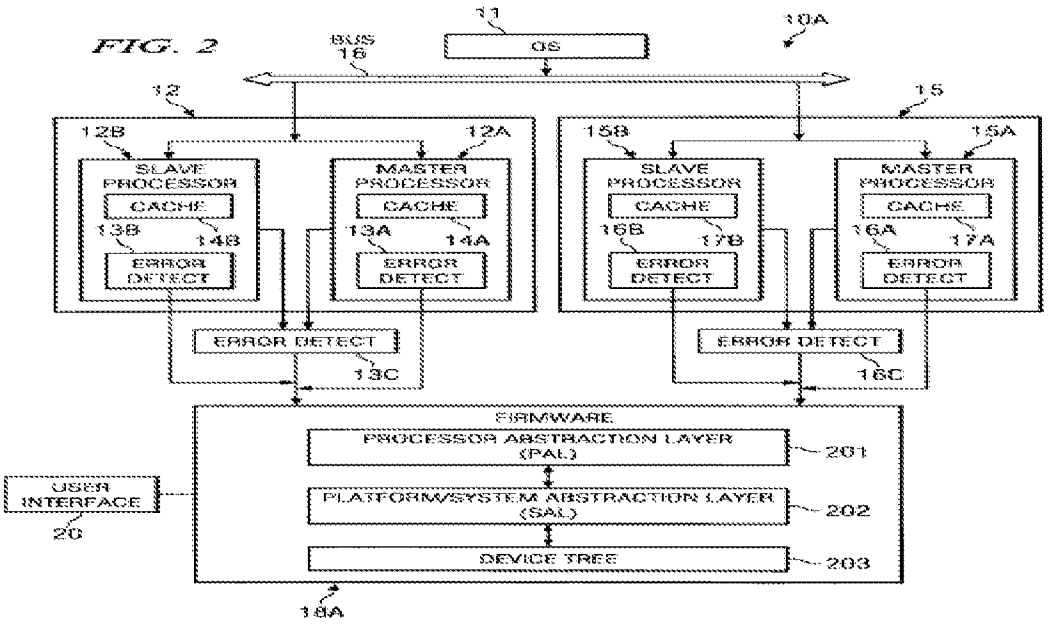
"In operational block 403, the LID of the current boot processor (for which LOL was detected) is reprogrammed to a different value. For instance, the current boot processor may be assigned a LID that is reserved just for this situation (i.e., for switching the role of boot processor). Accordingly, the boot processor for which LOL was detected will not respond to the interrupts addressed for its original LID because the boot processor's LID value is reprogrammed to a different value." (col. 16, 27-35)

"Turning to FIG. 5, an exemplary operational flow diagram for switching the role of boot processor to an active spare processor according to one embodiment is shown. Again, an LOL has been detected for the boot processor. In operational block 501, the system firmware, responsive to such detection of LOL for the boot processor, examines the device tree to determine which processor module is assigned the role of active spare. In operational block 502 the system firmware (e.g., SAL) asserts a General Purpose Event Interrupt (GPE) by writing to the appropriate GPE register in the ACPI register space. This causes the OS to execute its interrupt

handler, which will run. an AML Method associated with this particular interrupt. In this example, this AML method that is executed responsive to the interrupt generated by the GPE includes an AML instruction called "Notify". This "Notify" operation indicates the determined active spare processor object and that an Eject is being requested as arguments. Accordingly, an AML method is used to request the OS to eject the active spare that was determined in block 501. In response, the OS idles the active spare processor and returns control over the active spare processor to the system firmware, in block 503" (col. 17:12-33).

"In operational block 603, responsive to determining that the processor for which LOL is detected is the system's boot processor, the system's firmware switches the role of boot processor to a spare processor, which may be either a hot spare or an active spare, as described above. Accordingly, the system's OS recognizes the spare processor as the system boot processor. In view of the above, embodiments hereof enable the role of boot processor to be dynamically switched, during system runtime, to a spare processor responsive to a detected LOL for the boot processor, thus enabling recovery from the LOL without requiring that the system be crashed or re-booted." (col. 18:59- col. 19:3)





FIGs. 1-2 of the 302 Patent.

FIG. 5

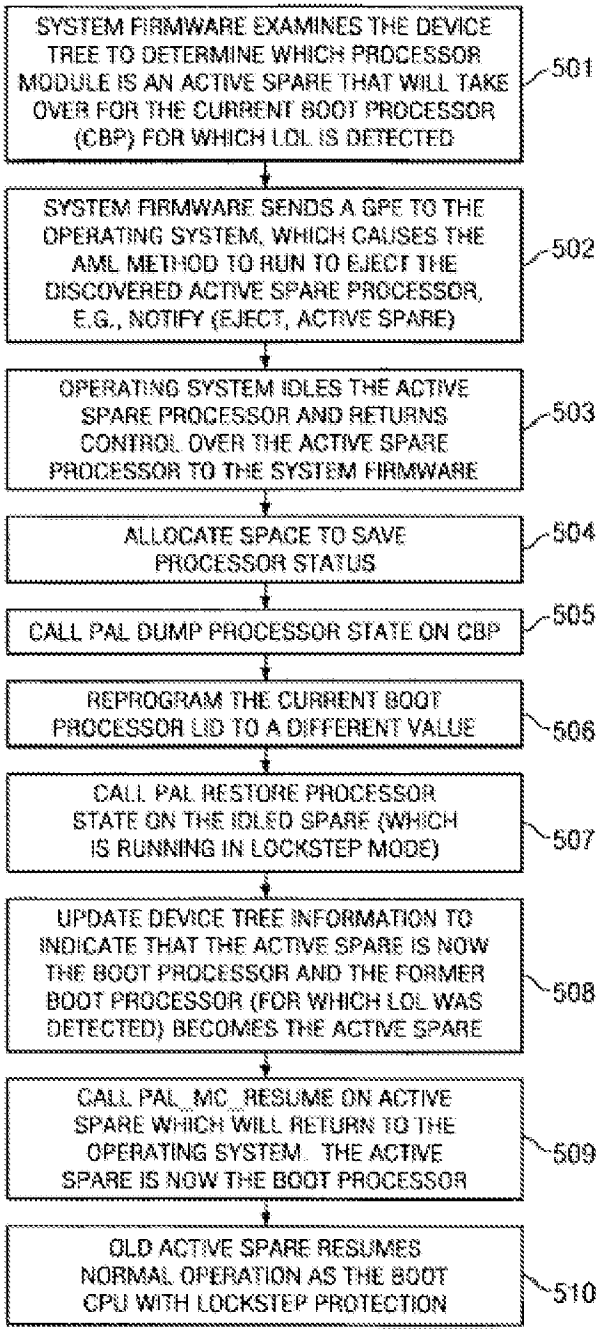
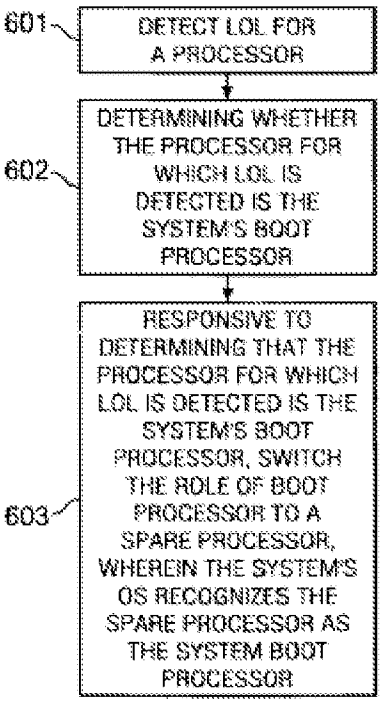


FIG. 6



Figs. 5-6 of the 302 Patent

In view of these findings, pursuant to 35 U.S.C. §112 ¶ 6th, FL #5 in claim 21 will be limited to the corresponding structures discussed above in conjunction with the appropriate software modules or programming noted above for switching the role of boot processor to the spare processor without shutting down the system if determined that the processor for which the LOL is detected and assigned the role of boot processor.

If applicant does not intend to have this limitation, FL #5 in claim 21 interpreted under 35 U.S.C. §112 ¶6th applicant may: (1) amend the claim limitation(s) to avoid it/them being interpreted under 35 U.S.C. §112 ¶6th (e.g., by reciting sufficient structure to perform the claimed function); or (2) present a sufficient showing that the claim limitation recites sufficient structure to perform the claimed function so as to avoid it/them being interpreted under 35 U.S.C. §112 ¶6th.

RESPONSE TO ARGUMENT

After reconsider, Examiners generally agree in part with the Patent Owner's arguments presented on the May 2024 Response, pages 5-14. Although Examiners do not agree that "*Kalyanasundharam is devoid of any discussion to teach or suggest the act of determining that the processor for which the loss of lockstep is detected is assigned a particular role, namely the boot processor role, as claimed*" (argument page 8, last paragraph). Examiners only agree that Kalyanasundharam fails to specifically indicate that the processor is a "boot processor"; Kennedy is used as secondary reference to teach the feature

"boot processor". Kalyanasundharam does in fact teach in Figs. 1, 2 and 7 an error detecting mechanism (26A) that checks for errors within processors 18A-18B that execute in lockstep in a multi-processor system (Fig. 1): "*In embodiments in which the processor cores 18A-18B execute in lockstep, the redundancy check circuit 26A may check that corresponding communications are concurrently received from the processor cores 18A-18B (e.g. received on the same clock cycle). For example, if a valid communication is received from one of the processor cores 18A-18B and a valid communication is not concurrently received from the other processor core 18A- 18B, an execution error may be detected*" (Kalyanasundharam at col. 4:27-35). Nevertheless, Examiners agree with the Patent Owner that the prior art references of record (Kalyanasundharam and Kennedy) fail to teach or suggest the combination of the claimed limitations "**determining that the processor for which said LOL is detected is assigned a role of boot processor; changing a system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected, to a different value, which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor**" (claim 1) and "**system firmware determining that loss of lockstep (LOL) is detected for a lockstep pair of processors that are assigned the role of boot processor in a system; ... changing a system identifier, used by the system's operating system to recognize the boot processor, of said lockstep pair of processors that are assigned the role of**

boot processor, to a different value, which causes the s system's operating system to not recognize the lock step pair of processors for which said LOL is detected as the processor" (claim 13) in view of the May 2024 Response to independent claims 1 and 13.

Explanation as to why Kalyanasundharam and Kennedy, the previously applied references, lack these features is given in the May 2024 Response, which the Examiners agree with. See May 2024 Response, pages 5-14.

Therefore, the previous rejection of claims 1, 13 (and dependent claims 5-8, 17) as being unpatentable over Kalyanasundharam in view of Kennedy has been withdrawn.

STATEMENT OF REASONS FOR PATENTABILITY AND/OR CONFIRMATION

The following is an examiner's statement of reasons for patentability and/or confirmation of the claims found patentable in this reexamination proceeding:

Regarding claim 1: The prior arts of record fail to specifically suggest or disclose "**determining that the processor for which said LOL is detected is assigned a role of boot processor; changing a system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected, to a different value, which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor**" method steps in

combination with "*changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to a spare processor without shutting down the system's operating system*" method step.

Dependent claims 5-8 are allowable for the same reason as independent claim 1 from which they depend.

Regarding claim 13: The prior arts of record fail to specifically suggest or disclose "**system firmware determining that loss of lockstep (LOL) is detected for a lockstep pair of processors that are assigned the role of boot processor in a system; ... changing a system identifier, used by the system's operating system to recognize the boot processor, of said lockstep pair of processors that are assigned the role of boot processor, to a different value, which causes the s system's operating system to not recognize the lock step pair of processors for which said LOL is detected as the processor**" in combination with "copying the state ..." and "changing a system identifier of the spare processor...".

Dependent claim 17 is allowable for the same reason as independent claim 13 from which it depends.

Regarding claim 21: The prior art of record fails to specifically suggest or disclose the corresponding structure for FL #4 - FL#5 as discussed above in the Claim Interpretation section: "**error detection logic operable to detect loss of lockstep (LOL) for at least one of said processor modules**" in combination

with "**system firmware operable, responsive to the detection of LOL for a first of said processor modules, to determine whether said first processor module is assigned a role of system boot processor; wherein if determined that said first processor module is assigned the role of system boot processor, said system firmware is further operable to cause said operating system to recognize another processor module as system boot processor without shutting down said operating system**."

Claims 22 and 25 are allowable at least by virtue of their dependency from independent claim 21.

Regarding claim 26: The prior art of record fails to specifically suggest or disclose the corresponding structure for FL #1 - FL#3 as discussed above in the Claim Interpretation section: "**system firmware means for detecting loss of lockstep (LOL) for a processor in a multi-processor system**"; "**means for determining whether the processor for which said LOL is detected is assigned the role of boot processor**" in combination with "**system firmware means for switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which said LOL is detected is assigned the role of boot processor**".

Any comments considered necessary by PATENT OWNER regarding the above statement must be submitted promptly to avoid processing delays. Such submission by the Patent Owner should be labeled: "Comments on Statement

of Reasons for Patentability and/or Confirmation" and will be placed in the reexamination file.

Information regarding the status of reexamination proceedings may be obtained from the USPTO's Patent Center. To file and manage patent submissions in Patent Center, visit: https://patentcenter.uspto.gov.

Visit https://www.uspto.gov/patents/apply/patent-center for more information about Patent Center and https://www.uspto.gov/patents/docx for information about filing in DOCX format. For additional questions, contact the Electronic Business Center (EBC) at (866) 217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative, call (800) 786-9199 (IN USA OR CANADA) or (571) 272-1000.

Any inquiry concerning this communication or earlier communications from the Reexamination Legal Advisor or Examiner, or as to the status of this proceeding, should be directed to the Central Reexamination Unit at telephone number (571) 272-7705.

Signed:

/MY TRANG TON/
Primary Examiner, Art Unit 3992

Conferees:

/KENNETH WHITTINGTON/
Primary Examiner, Art Unit 3992

/ANDREW J. FISCHER/
Supervisory Patent Reexamination Specialist, Art Unit 3992

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	90/019,243	7624302
	Examiner	Art Unit
	MY TRANG TON	3992

CPC - Searched*		
Symbol	Date	Examiner

CPC Combination Sets - Searched*		
Symbol	Date	Examiner

US Classification - Searched*			
Class	Subclass	Date	Examiner

* See search history printout included with this form or the SEARCH NOTES box below to determine the scope of the search.

Search Notes		
Search Notes	Date	Examiner
P-TACTS searched	09/18/2023	mt
PDP searched	09/18/2023	mt
Reviewed US Pat. No. 7624302 & App. No. 10/972588 file history	09/18/2023	mt
Litigation reviewed	09/18/2023	mt
update P-TACTS, PDP, litigation search	06/06/2024	mt

Interference Search			
US Class/CPC Symbol	US Subclass/CPC Group	Date	Examiner
	n/a		

Reexamination	Application/Control No.	Applicant(s)/Patent Under Reexamination
	90/019,243	7624302
	Certificate Date	Certificate Number
		C1

Requester Correspondence Address: ☐ Patent Owner ☑ Third Party
GREENBERG TRAURIG (NJ) 500 CAMPUS DRIVE, SUITE 400 P.O. BOX 677 FLORHAM PARK NJ 07932

LITIGATION REVIEW ☑	mt (examiner initials)	18 September 2023 (date)
Case Name		Director Initials
1:23cv640 | Foras Technologies Ltd. V. Nissan Motor Company, Ltd. Et Al (Open)		
2:23cv150 Foras Technologies Ltd. V. Toyota Motor North America. Inc. Et Al (closed)		

COPENDING OFFICE PROCEEDINGS	
TYPE OF PROCEEDING	**NUMBER**
None	

Issue Classification	Application/Control No.	Applicant(s)/Patent Under Reexamination
	90/019,243	7624302
	Examiner	Art Unit
	MY TRANG TON	3992

CPC														
Symbol													Type	Version
	/			/										

CPC Combination Sets														
Symbol													Type	Set	Ranking	Version
	/			/												

NONE		Total Claims Allowed:	
(Assistant Examiner)	(Date)	11	
/MY TRANG TON/ Primary Examiner, Art Unit 3992	10 June 2024	O.G. Print Claim(s)	O.G. Print Figure
(Primary Examiner)	(Date)	1	1

Issue Classification	Application/Control No.	Applicant(s)/Patent Under Reexamination
	90/019,243	7624302
	Examiner	**Art Unit**
	MY TRANG TON	3992

INTERNATIONAL CLASSIFICATION					
CLAIMED					
G06F	/	11	/	00	
NON-CLAIMED					
	/		/		

US ORIGINAL CLASSIFICATION	
CLASS	**SUBCLASS**
714	12

CROSS REFERENCES(S)							
CLASS	**SUBCLASS (ONE SUBCLASS PER BLOCK)**						
714	11						

NONE (Assistant Examiner) (Date)	**Total Claims Allowed:** 11	
/MY TRANG TON/ Primary Examiner, Art Unit 3992 (Primary Examiner) 10 June 2024 (Date)	O.G. Print Claim(s) 1	O.G. Print Figure 1

Issue Classification	Application/Control No.	Applicant(s)/Patent Under Reexamination
	90/019,243	7624302
	Examiner	Art Unit
	MY TRANG TON	3992

☑ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIMS

Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original
	1		10		19										
	2		11		20										
	3		12		21										
	4		13		22										
	5		14		23										
	6		15		24										
	7		16		25										
	8		17		26										
	9		18		27										

NONE		Total Claims Allowed:	
(Assistant Examiner)	(Date)	11	
/MY TRANG TON/ Primary Examiner, Art Unit 3992	10 June 2024	O.G. Print Claim(s)	O.G. Print Figure
(Primary Examiner)	(Date)	1	1

Bibliographic Data

Application No: 90/019,243

Foreign Priority claimed: ○ Yes ◉ No

35 USC 119 (a-d) conditions met: ☐ Yes ☑ No ☐ Met After Allowance

Verified and Acknowledged: /MY TRANG TON/ Examiner's Signature | Initials

Title: SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR

FILING or 371(c) DATE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.
09/01/2023	714	3992	
RULE			

APPLICANTS

INVENTORS

7624302,

CONTINUING DATA

This application is a REX of 10972588 10/25/2004 PAT 7624302

FOREIGN APPLICATIONS

IF REQUIRED, FOREIGN LICENSE GRANTED**

STATE OR COUNTRY

ADDRESS

FILING FEE RECEIVED

$275

Issue Classification	Application/Control No.	Applicant(s)/Patent Under Reexamination
	90/019,243	7624302
	Examiner	Art Unit
	MY TRANG TON	3992

CPC								
Symbol							Type	Version
G06F	/	11		00			F	2013-01-01

CPC Combination Sets									
Symbol							Type	Set	Ranking	Version
	/			/						

NONE		Total Claims Allowed:	
(Assistant Examiner)	(Date)	11	
/MY TRANG TON/ Primary Examiner, Art Unit 3992	10 June 2024	O.G. Print Claim(s)	O.G. Print Figure
(Primary Examiner)	(Date)	1	1

U.S. Patent and Trademark Office — Part of Paper No.: 20240610

Issue Classification	Application/Control No.	Applicant(s)/Patent Under Reexamination
	90/019,243	7624302
	Examiner	Art Unit
	MY TRANG TON	3992

INTERNATIONAL CLASSIFICATION					
CLAIMED					
G06F	/	11	/	00	
NON-CLAIMED					
	/		/		

US ORIGINAL CLASSIFICATION	
CLASS	SUBCLASS
714	12

CROSS REFERENCES(S)							
CLASS	SUBCLASS (ONE SUBCLASS PER BLOCK)						
714	11						

NONE		Total Claims Allowed:	
(Assistant Examiner)	(Date)	11	
/MY TRANG TON/ Primary Examiner, Art Unit 3992	10 June 2024	O.G. Print Claim(s)	O.G. Print Figure
(Primary Examiner)	(Date)	1	1

Issue Classification	Application/Control No.	Applicant(s)/Patent Under Reexamination
	90/019,243	7624302
	Examiner	Art Unit
	MY TRANG TON	3992

☑ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIMS

Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original
	1		10		19										
	2		11		20										
	3		12		21										
	4		13		22										
	5		14		23										
	6		15		24										
	7		16		25										
	8		17		26										
	9		18		27										

NONE		Total Claims Allowed:	
(Assistant Examiner)	(Date)	11	
/MY TRANG TON/ Primary Examiner, Art Unit 3992	10 June 2024	O.G. Print Claim(s)	O.G. Print Figure
(Primary Examiner)	(Date)	1	1

US007624302C1

(12) EX PARTE REEXAMINATION CERTIFICATE (12688th)

United States Patent

Michaelis et al.

(10) Number: US 7,624,302 C1

(45) Certificate Issued: Aug. 27, 2024

(54) **SYSTEM AND METHOD FOR SWITCHING THE ROLE OF BOOT PROCESSOR TO A SPARE PROCESSOR RESPONSIVE TO DETECTION OF LOSS OF LOCKSTEP IN A BOOT PROCESSOR**

(75) Inventors: **Scott L. Michaelis**, Plano, TX (US); **Anurupa Rajkumari**, Round Rock, TX (US); **William B. McHardy**, Fort Collins, CO (US)

(73) Assignee: **Foras Technologies Ltd.**

Reexamination Request:
No. 90/019,243, Sep. 1, 2023

Reexamination Certificate for:

Patent No.:	**7,624,302**
Issued:	**Nov. 24, 2009**
Appl. No.:	**10/972,588**
Filed:	**Oct. 25, 2004**

Certificate of Correction issued Feb. 16, 2010

(51) **Int. Cl.**
G06F 11/00 (2006.01)

(52) **U.S. Cl.**
CPC ***G06F 11/00*** (2013.01)

(58) **Field of Classification Search**
None
See application file for complete search history.

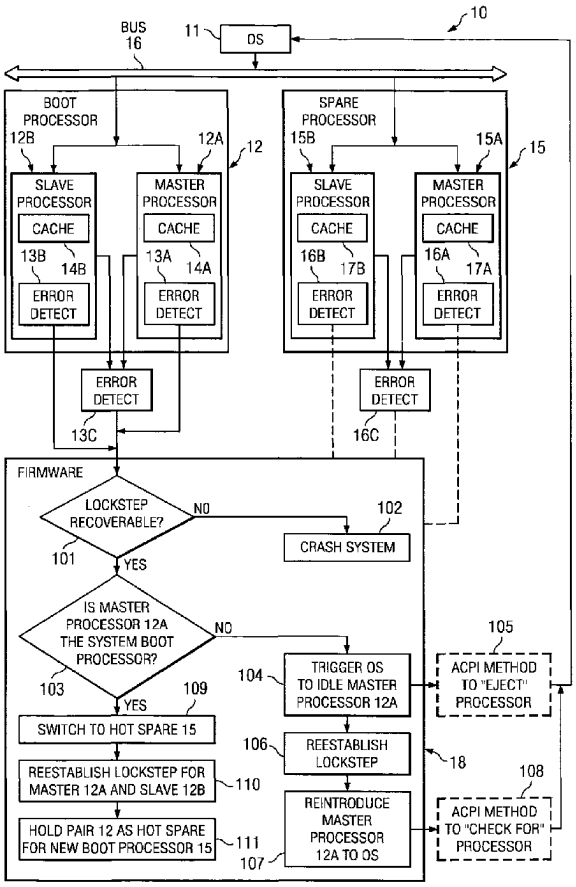
(56) **References Cited**

To view the complete listing of prior art documents cited during the proceeding for Reexamination Control Number 90/019,243, please refer to the USPTO's Patent Electronic System.

Primary Examiner — My Trang Ton

(57) **ABSTRACT**

According to one embodiment, a method comprises detecting loss of lockstep (LOL) for a processor in a multiprocessor system. The method further comprises determining that the processor for which the LOL is detected is assigned the role of boot processor, and switching the role of boot processor to a spare processor without shutting down the system's operating system. In another embodiment, a method comprises system firmware determining that an LOL is detected for a lockstep pair of processors that are assigned the role of boot processor in a system. The method further comprises determining one of the lockstep pair of processors that is not the cause of the LOL, and copying the state of the determined one of the lockstep pair of processors that is not the cause of the LOL to a spare processor. The method further comprises switching the role of boot processor to the spare processor.



EX PARTE REEXAMINATION CERTIFICATE

NO AMENDMENTS HAVE BEEN MADE TO THE PATENT

AS A RESULT OF REEXAMINATION, IT HAS BEEN DETERMINED THAT:

The patentability of claims **1**, **5-8**, **13**, **17**, **21**, **22**, **25**, and **26** is confirmed.

Claims **2-4**, **9-12**, **14-16**, **18-20**, **23-24**, and **27** were not reexamined.

* * * * *