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.
90/019,245	09/01/2023	7502958	FTL-REX245	6989

89220 7590 03/19/2024
Antonio Papageorgiou
Lombard Geliebter LLP
230 PARK AVENUE 4TH FL WEST
New York, NY 10169

EXAMINER
SORRELL, ERON J

ART UNIT	PAPER NUMBER
3992	

MAIL DATE	DELIVERY MODE
03/19/2024	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.

PTOL-90A (Rev. 04/07)

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, NEW JERSEY 07932

EX PARTE REEXAMINATION COMMUNICATION TRANSMITTAL FORM

REEXAMINATION CONTROL NO. *90/019,245* .

PATENT UNDER REEXAMINATION *7502958* .

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. 90/019,245	Patent Under Reexamination 7502958	
	Examiner ERON J SORRELL	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 1-25 are subject to reexamination.

1b. ☐ Claims _____ are not subject to reexamination.

2. ☐ Claims _____ have been canceled in the present reexamination proceeding.

3. ☐ Claims _____ are patentable and/or confirmed.

4. ☑ Claims 1-25 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)

DETAILED ACTION

Summary of the Proceeding to Date

On September 1, 2023, Third-Party Requester, Unified Patents, LLC, (Requester) requested Ex Parte Reexamination of claims 1-25 of U.S. Patent No. 7,502,958 to Michaelis et al. ("the Request").

On November 29, 2023, an Order granting the Request was mailed indicating the Request raised at least one substantial new question of patentability.

Listing of Prior Art

The following references are cited in this proceeding:

i. U.S. Patent Application Publication No. 2003/0126498 to Bigbee et al. ("Bigbee");

ii. ii. U.S. Patent No. 7,085,959 to Safford ("Safford");

iii. dil. U.S. Patent Application Publication No. 2003/0051190 to Marisetty et al. ('"Marisetty");

iv. iv. U.S. Patent No. 5,491,788 to Cepulis et al. ("Cepulis").

Order Granting Reexamination

In the Order, the Examiner determined that Bigbee *did not* raise a SNQ with respect to claims 13-18 citing a distinction

between "idling" processors and "virtually ejecting" them. Upon further review Bigbee, the Examiner finds Bigbee does reasonably disclose idling the processors. The Examiners prior finding was based on a misunderstanding of certain portions of the reference.

Statutory Basis for Rejections

35 USC § 102

The following is a quotation of the appropriate paragraphs of pre-AIA 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 -
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

35 USC § 103

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 negated by the manner in which the invention was made.

Bigbee-based Rejections

Claim(s) 1, 4-8, and 19-22 are rejected under pre-AIA 35 U.S.C. 102(b) as being anticipated by Bigbee.

Referring to claim 1, Bigbee teaches a method comprising:

detecting loss of lockstep for a lockstep pair of processors (see ¶ 19, "*FRC logic detects errors within individual processor cores of the described embodiment which causes hardware 102 to break lock step between at least two processor cores within a processor package, disabling FRC mode operation.*");

using firmware to trigger an operating system to idle the lockstep pair of processors (see ¶ 20, "…*the processor error handler causes a platform error handler within a SAL associated with the package to be executed which initiates a platform-independent device removal sequence for a processor associated with the processor package by generating a system control interrupt (SCI) to operating system 106.*" See also ¶ 22, "*Operating system 106 receives or detects the firmware-generated SCI and utilizes ACPI control methods to determine that the interrupt was caused by a request for processor removal or*

ejection." See also ¶ 23, *"Operating system 106 next utilizes ACPI control methods to virtually eject the processor from data processing system 100… …In another embodiment, virtually ejecting the processor results in the physical removal of power from its associated processor package."*),

recover lockstep for the lockstep pair of processors (see ¶ 24, "*Firmware 104, via ACPI BIOS program code then resets the package to its original state at the time of operating system 106 initial load/boot. In alternative embodiments of the invention, this reset may comprise a "cold" reset or "RESET" operation or a "warm" or "INIT" reset operation. An associated SAL reinitializes the package and re-enables FRC mode lock step operation between two or more of the package's associated processor cores in response to detecting the ACPI BIOS-initiated reset operation."*), and

trigger the operating system to recognize the lockstep pair of processors having recovered lockstep (see ¶ 24, "*…In yet another embodiment, the package's associated SAL initiates a platform-independent device insertion sequence for the processor package's associated processor by generating a system control interrupt (SCI) to operating system 106 via the previously disclosed INT_OUT command capability."* See also ¶ 25, *"Operating system 106 receives the firmware-generated SCI and utilizes ACPI*

control methods to determine that the interrupt was caused by a request for processor insertion. Operating system 106 responsively modifies its internal data structures to reflect that the package's associated processor is being brought online and then wakes and configures the associated processor for use.”); and

responsive to said detecting loss of lockstep, using said firmware to determine if lockstep is recoverable for the lockstep pair of processors, wherein said using said firmware to determine if lockstep is recoverable further comprises determining if a lockstep mismatch has occurred between the lockstep pair of processors (see ¶ 33, *“FRC logic 308 monitors processor core store operations or "writes" to shared cache 306 to detect inconsistencies in the generated results which indicate an FRC mismatch error. Such inconsistencies can occur as a result of partial or total hardware failure, or the effect of an alpha particle or other form of natural or artificial radiation impacting one or more processor cores.”* see also ¶ 34, *“In alternative embodiments, FRC logic 308 may be implemented as executable firmware or dedicated hardware coupled with or incorporated into one or more of the processor package's 204a components.”*).

Referring to claim 4, Bigbee teaches the method of claim 1, as shown above, and Bigbee further teaches detecting loss of lockstep comprises detecting corrupt data in one of said lockstep pair of processors that would lead to a lockstep mismatch if acted upon (see ¶ 18, *"For example, single bit data error correcting code (ECC) errors occurring in processor cache may be handled by processor hardware while multi-bit data ECC or parity errors occurring in processor cache may be handled by an associated PAL."* See also ¶ 35 *"The illustrated process begins (block 402) and thereafter hardware (e.g FRC logic) detects an error within a first processor core of a plurality of processor cores operating in an FRC mode as a single logical processor…"*).

Referring to claim 5, Bigbee teaches the method of claim 1, as shown above, and Bigbee further teaches detecting loss of lockstep comprises detecting a precursor to lockstep mismatch (see ¶ 18, *"For example, single bit data error correcting code (ECC) errors occurring in processor cache may be handled by processor hardware while multi-bit data ECC or parity errors occurring in processor cache may be handled by an associated PAL."* See also ¶ 35 *"The illustrated process begins (block 402) and thereafter hardware (e.g FRC logic) detects an error within*

a first processor core of a plurality of processor cores operating in an FRC mode as a single logical processor…").

Referring to claim 6, Bigbee teaches the method of claim 1 (as shown above) and Bigbee further teaches detecting loss of lockstep comprises detecting lockstep mismatch between the lockstep pair of processors (see ¶ 33, *"FRC logic 308 monitors processor core store operations or "writes" to shared cache 306 to detect inconsistencies in the generated results which indicate an FRC mismatch error."*).

Referring to claim 7, Bigbee teaches the method of claim 1, as shown above, and Bigbee further teaches if determined that lockstep mismatch has not occurred between the lockstep pair of processors (see paragraph 19, *"FRC logic detects errors within individual processor cores of the described embodiment which causes hardware 102 to break lock step between at least two processor cores within a processor package, disabling FRC mode operation."*), then determining that the loss of lockstep is recoverable (see ¶ 21 *"…attempts to correct processor-related errors where possible…"*).

Referring to claim 8, Bigbee teaches the method of claim 1, as shown above, and Bigbee further teaches if determined that a lockstep mismatch has occurred between the lockstep pair of processors then determining that the loss of lockstep is not recoverable (see ¶ 21 "*…attempts to correct processor-related errors where possible…*").

Referring to claim 19, Bigbee teaches a system comprising a pair of lockstep processors and computer-executable firmware code stored to a computer-readable medium, the computer-executable firmware code comprising:

firmware code, responsive to detection of loss of lockstep between the pair of lockstep processors, for determining if the lockstep is recoverable for the pair of lockstep processors, wherein said firmware code for determining if the lockstep is recoverable further comprises firmware code for determining if a lockstep mismatch has occurred between the pair of lockstep processors (see ¶ 33, "*FRC logic 308 monitors processor core store operations or "writes" to shared cache 306 to detect inconsistencies in the generated results which indicate an FRC mismatch error. Such inconsistencies can occur as a result of partial or total hardware failure, or the effect of an alpha*

particle or other form of natural or artificial radiation impacting one or more processor cores." see also ¶ 34, *"In alternative embodiments, FRC logic 308 may be implemented as executable firmware or dedicated hardware coupled with or incorporated into one or more of the processor package's 204a components."*); and

firmware code, responsive to determining that the lockstep is recoverable, for triggering an operating system to idle the processors (see ¶ 36, *"OS software then… …queries firmware (e.g. ACPI BIOS) to determine the source of the SCI (block 408). In one embodiment, this may be accomplished by the OS executing an _Lxx control method. Firmware responds to the OS query by notifying the operating system of a request for processor hot removal (block 410). This may be accomplished using an ACPI Notify command. The operating system then modifies its internal data structures to indicate that the processor is offline and requests firmware ejection of the processor (block 412). In one embodiment, the operating system requests firmware ejection of the processor by executing an _EJx control method." See also ¶ 23, "…In another embodiment, virtually ejecting the processor results in the physical removal of power from its associated processor package."*),

attempting to recover lockstep between the pair of processors, and if lockstep is successfully recovered between the pair of processors (see ¶ 37, *"resets the processor, re-enables FRC operation of at least the first and second processor cores and generates an SCI to the Operating System"*),

triggering said operating system to recognize the processors as being available for receiving instructions (see ¶ 37, *"Thereafter, in the process illustrated in FIG. 4b, operating system software queries firmware (e.g. ACPI BIOS) to determine the source of the previously-generated SCI (block 416). Firmware responds to the OS query by notifying the operating system of a request for processor hot insertion (block 418)… …Once the processor has been inserted and/or activated as described, the OS modifies its internal data structures to indicate that the processor is online and available before the illustrated process terminates (block 430)."* See also ¶ 25, *"Operating system 106 receives the firmware-generated SCI and utilizes ACPI control methods to determine that the interrupt was caused by a request for processor insertion. Operating system 106 responsively modifies its internal data structures to reflect that the package's associated processor is being brought online and then wakes and configures the associated processor for use."*).

Referring to claim 20, Bigbee teaches the system of claim 19, as shown above, and Bigbee further teaches the code for attempting to recover lockstep further comprises code for resetting the pair of processors (see ¶ 37, "…*firmware virtually ejects the processor from the system, resets the processor, re-enables FRC operation of at least the first and second processor cores…*").

Referring to claim 21, Bigbee teaches the system of claim 19, as shown above, and Bigbee further teaches the operating system is an Advanced Configuration and Power Interface (ACPI)-compatible operating system, said code for triggering the operating system to idle the processors comprises code for using an ACPI method to idle the processors (see ¶ 23, "*Operating system 106 next utilizes ACPI control methods to virtually eject the processor from data processing system 100.*").

Referring to claim 22, Bigbee teaches the system of claim 19 wherein said operating system is an Advanced Configuration and Power Interface (ACPI)-compatible operating system, said code for triggering said operating system to recognize the

processors as being available for receiving instructions comprises code for using an ACPI method to cause the operating system to check for the processors (see ¶ 25, *"Operating system 106 receives the firmware-generated SCI and utilizes ACPI control methods to determine that the interrupt was caused by a request for processor insertion. Operating system 106 responsively modifies its internal data structures to reflect that the package's associated processor is being brought online and then wakes and configures the associated processor for use."* Note the OS waking and configuring the processor are being construed as "checking for" the processor).

Claims 2-5 and 13-18 are rejected under pre-AIA 35 U.S.C. 103(a) as being unpatentable over Bigbee in view of Safford.

Referring to claim 2, Bigbee teaches the method of claim 1, as shown above, and Bigbee further teaches using firmware to trigger an operating system to idle the lockstep pair of processors further comprises using the firmware to trigger the operating system to idle a first processor of the lockstep pair of processors (See ¶ 23, *"Operating system 106 next utilizes ACPI control methods to virtually eject the processor from data*

processing system 100… …In another embodiment, virtually ejecting the processor results in the physical removal of power from its associated processor package.").

Bigbee fails to explicitly disclose the first processor is designated as a master processor.

Safford teaches, in an analogous system, a computer system having first and second processors that operate in lockstep, wherein the first processor is a master processor and the second processor is a slave processor, wherein such configuration improves overall reliability of the system (see lines 55-60 of column 2). Safford is analogous to the claimed invention as both are directed toward recovering from a loss of lock step in a multiprocessor system.

It would have been obvious to one of ordinary skill in the art at the time of the Patent Owner's invention to modify the teachings of Bigbee with the above teachings of Safford such that Bigbee's first and second processors are configured as master processor and slave processor, respectively. Bigbee's system differs from the claimed system only in that Bigbee does not explicitly disclose the pair of processors are a master and slave, which configuration is well known in the art as shown by Safford. A person of ordinary skill could have easily

substituted the dual core architecture of Bigbee with the dual core architecture of Safford and the results of such substitution would have been reasonably predictable (See MPEP 2143(I)).

Referring to claim 3, the combination of Bigbee and Safford teaches the method of claim 2, as shown above, and the combination as applied further teaches using firmware to trigger the operating system to recognize the lockstep pair of processors having recovered lockstep further comprises using the firmware to trigger the operating system to recognize the master of the lockstep pair of processors (See Bigbee ¶ 37, *"Once the relevant status and resource data is received, the operating system requests insertion of the processor (block 424) and the firmware virtually inserts the processor into the system (block 426). In one embodiment, the operating system may request insertion of the processor using a _PS0 control method which puts a device into its D0 or "active" state. Once the processor has been inserted and/or activated as described, the OS modifies its internal data structures to indicate that the processor is online and available before the illustrated process terminates (block 430)."*).

Referring to claim 13, Bigbee teaches a system (see figure 2) comprising:

an Advanced Configuration and Power Interface (ACPI)-compatible operating system (see ¶ 16, "*In one embodiment, system 100 comprises an advanced configuration and power interface (ACPI)-compliant data processing system. Accordingly, operating system 106 includes operating system-directed power management (OSPM) program code, and an ACPI device driver and firmware 104 comprises an ACPI BIOS including ACPI machine language (AML) program code describing what hardware devices are present within data processing system 100, as well as their configuration and interfaces, via ACPI control methods, objects, registers, and tables. The disclosed ACPI device driver within operating system 106 acts as an AML interpreter to interpret and execute AML program code.*");

a first processor (see item 302A in figure 3) operable to process instructions scheduled by said ACPI-compatible operating system (see ¶ 33, ¶ "*In one embodiment, the first and second processor cores 302a-b within the depicted processor package 204a may be bound together to function as a single logical*

processor from the point of view of operating system software. Both cores 302a and 302b operate in an FRC mode in which identical instructions may be provided to each core and concurrently executed on identical data utilizing FRC logic 308… Each processor core 302a-b executes instructions and processes data to generate results which may then be stored within an associated private (e.g. 304a-b) and/or shared (e.g. 306) cache.");

a second processor (see item 302B in figure 3) lockstepped with the first processor operable to perform redundant processing of said instructions scheduled for the first processor (see ¶ 33, ¶ "*In one embodiment, the first and second processor cores 302a-b within the depicted processor package 204a may be bound together to function as a single logical processor from the point of view of operating system software. Both cores 302a and 302b operate in an FRC mode in which identical instructions may be provided to each core and concurrently executed on identical data utilizing FRC logic 308…*"); and

firmware (see item 104 in figure 1) operable, responsive to detection of loss of lockstep between the first and second processors, to use an ACPI method to trigger the operating system to idle the first processor (see ¶ 35, "*…(e.g FRC logic)*

detects an error within a first processor core of a plurality of processor cores operating in an FRC mode as a single logical processor, disables the first processor core, and generates a system-transparent machine check exception (block 404) which in turn causes firmware (e.g. SAL program code) to generate a system-visible system control interrupt (SCI) to operating system (OS)…" see also ¶ 36, "Firmware responds to the OS query by notifying the operating system of a request for processor hot removal (block 410). This may be accomplished using an ACPI Notify command. The operating system then modifies its internal data structures to indicate that the processor is offline and requests firmware ejection of the processor (block 412). In one embodiment, the operating system requests firmware ejection of the processor by executing an _EJx control method."),

attempt to recover lockstep between the first and second processors (see ¶ 37, "*…firmware virtually ejects the processor from the system, resets the processor, re-enables FRC operation of at least the first and second processor cores and generates an SCI to the Operating System (block 414)…"*), and

if recovery of lockstep is successful then use an ACPI method to reintroduce the first processor to the operating system (see ¶ 37, "*Firmware responds to the OS query by notifying the operating system of a request for processor hot*

insertion (block 418). Next, the operating system queries firmware to determine the status and to identify the associated resources of the processor to be inserted (block 420) and firmware provides the operating system with the requested data (block 422). In one embodiment, this may be accomplished by executing one or more control methods (e.g., _STA, _MAT, and _CRS). Once the relevant status and resource data is received, the operating system requests insertion of the processor (block 424) and the firmware virtually inserts the processor into the system (block 426).”).

Bigbee fails to explicitly disclose the first processor is a master processor and the second processor is a slave processor.

Safford teaches, in an analogous system, a computer system having first and second processors that operate in lockstep, wherein the first processor is a master processor and the second processor is a slave processor, wherein such configuration improves overall reliability of the system (see lines 55-60 of column 2). Safford is analogous to the claimed invention as both are directed toward recovering from a loss of lock step in a multiprocessor system.

It would have been obvious to one of ordinary skill in the art at the time of the Patent Owner's invention to modify the teachings of Bigbee with the above teachings of Safford such that Bigbee's first and second processors are configured as master processor and slave processor, respectively. Bigbee's system differs from the claimed system only in that Bigbee does not explicitly disclose the pair of processors are a master and slave, which configuration is well known in the art as shown by Safford. A person of ordinary skill could have easily substituted the dual core architecture of Bigbee with the dual core architecture of Safford and the results of such substitution would have been reasonably predictable (See MPEP 2143(I)).

Referring to claim 14, the combination of Bigbee and Safford teaches the system of claim 13, as shown above, and Bigbee teaches the system further comprising error detection logic (see item 308 in figure 3) for detecting loss of lockstep between the master and slave processors (see ¶ 33, *"FRC logic 308 monitors processor core store operations or "writes" to shared cache 306 to detect inconsistencies in the generated results which indicate an FRC mismatch error."*)

Referring to claim 15, the combination of Bigbee and Safford teaches the system of claim 13, as shown above, and Safford further teaches error detection logic for detecting a precursor to a loss of lockstep (see lines 21-31 of column 3, *"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."* see also lines 35-40 of column 4, *"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.""*).

It would have been obvious to one of ordinary skill in the art at the time of the Patent Owner's invention to further

modify the Bigbee with the above teachings Safford of in order to further improve the availability of processor assets as suggested by Safford (see lines 21-24 of column 4).

Referring to claim 16, the combination of Bigbee and Safford teaches the system of claim 15, as shown above, and the applied combination, via Safford, further teaches the precursor to loss of lockstep is treated as said detection of loss of lockstep (see Safford, see also lines 35-40 of column 4, "*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.""*).

It would have been obvious to one of ordinary skill in the art at the time of the Patent Owner's invention to further modify the Bigbee with the above teachings of Safford in order to further improve the availability of processor assets as suggested by Safford (see lines 21-24 of column 4).

Referring to claim 17, the combination of Bigbee and Safford teaches the system of claim 15, as shown above, and Bigbee further teaches error detection logic comprises a parity-based mechanism (see ¶ 33, *"FRC logic 308 monitors processor core store operations or "writes" to shared cache 306 to detect inconsistencies in the generated results which indicate an FRC mismatch error."* Note the FRC logic determines if the writes from both processor cores are equal which is a parity-based operation.)

Referring to claim 18, the combination of Bigbee as modified by Safford teaches the system of claim 13, as shown above, and the combination also teaches error detection logic for detecting lockstep mismatch between processors (see Bigbee item 308, and Safford, lines 55-60 of column 2).

As mentioned above, it would have been obvious to one of ordinary skill in the art at the time of the Patent Owner's invention to modify the teachings of Bigbee with the above teachings of Safford such that Bigbee's first and second processors are a master processor and slave processor, respectively. A person of ordinary skill could have easily substituted the dual core architecture of Bigbee with the dual

core architecture of Safford and the results of such substitution would have been reasonably predictable (See MPEP 2143(I)).

Safford-based Rejections

Claims 1-8 and 19-22 are rejected under pre-AIA 35 U.S.C. 103(a) as being unpatentable over Safford in view of Bigbee.

Referring to claim 1, Safford teaches a method comprising:

detecting loss of lockstep for a lockstep pair of processors (see lines 1-45 of column 3, "*External logic circuit 115 monitors outputs of the processors 111 and 113 and may be used to detect any differences in the outputs… …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.""*);

using logic to trigger the computer system to idle the lockstep pair of processors (see lines 25-63 of column 4, "*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 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."),

recover lockstep for the lockstep pair of processors (see lines 54-57 of column 4, "*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).*"), and

trigger the computer system to recognize the lockstep pair of processors having recovered lockstep (see lines 25-63 of column 4, "*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." Note that just prior to the reboot the system operates without "hot standby" processors, then after the reboot processor 111/113 are recognized as the "hot standby" processors.*).

Safford further teaches responsive to said detecting loss of lockstep, determining if lockstep is recoverable for the lockstep pair of processors, wherein determining if lockstep is

recoverable further comprises determining if a lockstep mismatch has occurred between the lockstep pair of processors (see lines 10-20 of column 3, *"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."* See also lines 61-66 of column 3, *"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."*);

Safford does not use the terms "firmware" or "operating system", therefore does not specifically disclose using firmware to trigger an Operating System (OS) to idle the lockstep processors, recover lockstep for the lockstep pair of processors, or trigger the OS to recognize the recovery. Safford also fails to teach using firmware to determine if lockstep is recoverable for the lockstep pair of processors comprises determining if a lockstep mismatch has occurred. Safford, instead, uses external logic to the determination including determining if a lockstep mismatch has occurred (see lines 1-45 of column 3).

Bigbee discloses a method for providing lock step protection in a system comprising multiple processors, comprising using firmware to trigger an operating system to idle a processor (see ¶ 20, "*…the processor error handler causes a platform error handler within a SAL associated with the package to be executed which initiates a platform-independent device removal sequence for a processor associated with the processor package by generating a system control interrupt (SCI) to operating system 106." See also ¶ 22, "Operating system 106 receives or detects the firmware-generated SCI and utilizes ACPI control methods to determine that the interrupt was caused by a request for processor removal or ejection." See also ¶ 23 "Operating system 106 next utilizes ACPI control methods to virtually eject the processor from data processing system 100… …In another embodiment, virtually ejecting the processor results in the physical removal of power from its associated processor package.*"),

recover lockstep for the lockstep pair of processors (see ¶ 24, "*Firmware 104, via ACPI BIOS program code then resets the package to its original state at the time of operating system 106 initial load/boot. In alternative embodiments of the invention, this reset may comprise a "cold" reset or "RESET" operation or a "warm" or "INIT" reset operation. An associated*

SAL reinitializes the package and re-enables FRC mode lock step operation between two or more of the package's associated processor cores in response to detecting the ACPI BIOS-initiated reset operation.”), and to

trigger the OS to recognize the recovery (see ¶ 24, “*In yet another embodiment, the package's associated SAL initiates a platform-independent device insertion sequence for the processor package's associated processor by generating a system control interrupt (SCI) to operating system 106 via the previously disclosed INT_OUT command capability.*”).

It would have been obvious to one of ordinary skill in the art at the time of the Patent Owner’s invention to combine the teachings of Safford and Bigbee to such that Safford system uses firmware to trigger an operating system to idle the processors having the detected error, recover lockstep, and trigger the OS to recognize the recovery. One of ordinary skill at the time of the Patent Owner’s invention would have been motivated to make such modification because Bigbee suggests that in advanced architecture computer systems which is type of system disclosed by Safford, error resolutions requiring processors be idled are handled with firmware to be platform independent (See ¶ 20).

Bigbee also teaches in response to detecting a loss of lockstep using said firmware to determine if lockstep is recoverable for the lockstep pair of processors, wherein said using said firmware to determine if lockstep is recoverable further comprises determining if a lockstep mismatch has occurred between the lockstep pair of processors (see ¶ 33-34, *"FRC logic 308 monitors processor core store operations or "writes" to shared cache 306 to detect inconsistencies in the generated results which indicate an FRC mismatch error. Such inconsistencies can occur as a result of partial or total hardware failure, or the effect of an alpha particle or other form of natural or artificial radiation impacting one or more processor cores… … In alternative embodiments, FRC logic 308 may be implemented as executable firmware or dedicated hardware coupled with or incorporated into one or more of the processor package's 204a components."*).

It would have been obvious to one of ordinary skill in the art at the time of the Patent Owner's invention to further modify the method of Safford with the above teachings of Bigbee such that the system uses firmware to determine if lockstep is recoverable comprising determining if a lockstep mismatch has occurred. Such a modification would merely require substituting the hardware-implemented lockstep logic in Safford with the

firmware-implemented FRC logic disclosed by Bigbee. One of ordinary skill would recognize that such substitution would have predictable result.

Referring to claim 2, the combination of Safford and Bigbee teaches the method of claim 1, as shown above, and Bigbee further teaches wherein using firmware to trigger an operating system to idle the lockstep pair of processors further comprises: using the firmware to trigger the operating system to idle a master processor of the lockstep pair of processors (*See also ¶ 22, "Operating system 106 receives or detects the firmware-generated SCI and utilizes ACPI control methods to determine that the interrupt was caused by a request for processor removal or ejection." See also ¶ 23 "Operating system 106 next utilizes ACPI control methods to virtually eject the processor from data processing system 100… …In another embodiment, virtually ejecting the processor results in the physical removal of power from its associated processor package."*).

It would have been obvious to one of ordinary skill in the art at the time of the Patent Owner's invention to combine the teachings of Safford and Bigbee to such that Safford system uses

firmware to trigger an operating system to idle the processors because Safford suggests both processors should be idled and Bigbee suggests that in advanced architecture computer systems which is type of system disclosed by Safford, error resolutions requiring processors be idled are handled by firmware to be platform independent (See ¶ 20).

Referring to claim 3, the combination of Safford and Bigbee teaches the method of claim 2, as shown above, and the combination further teaches wherein said using firmware to trigger the operating system to recognize the lockstep pair of processors having recovered lockstep further comprises using the firmware to trigger the operating system to recognize the master of the lockstep pair of processors (see ¶ 24, "*In yet another embodiment, the package's associated SAL initiates a platform-independent device insertion sequence for the processor package's associated processor by generating a system control interrupt (SCI) to operating system 106 via the previously disclosed INT_OUT command capability.*").

It would have been obvious to one of ordinary skill in the art at the time of the Patent Owner's invention to combine the teachings of Safford and Bigbee to such that Safford system uses

firmware to trigger an operating system to recognize the processors because Safford suggests both processors should be recognized (*See also ¶ 23 "Operating system 106 next utilizes ACPI control methods to virtually eject the processor from data processing system 100… …In another embodiment, virtually ejecting the processor results in the physical removal of power from its associated processor package."*) and Bigbee suggests that in advanced architecture computer systems which is type of system disclosed by Safford, error resolution requiring processors be idled, rebooted, and recognized as available are handled by firmware to be platform independent (See ¶ 20).

Referring to claim 4, Safford in view of Bigbee teaches the method of claim 1 (as shown above) and Safford further teaches detecting loss of lockstep comprises detecting corrupt data in one of said lockstep pair of processors that would lead to a lockstep mismatch if acted upon (see lines 35-40 of column 4, *"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.""*).

Referring to claim 5, the combination of Safford and Bigbee teaches the method of claim 1, as shown above, and Safford further teaches detecting loss of lockstep comprises detecting a precursor to lockstep mismatch (see lines 35-40 of column 4, "*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.""*).

Referring to claim 6, the combination of Safford and Bigbee teaches the method of claim 1, as shown above, and Safford further teaches detecting loss of lockstep comprises detecting lockstep mismatch between the lockstep pair of processors (see lines 10-20 of column 3, "*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.*").

Referring to claim 7, the combination of Safford and Bigbee teaches the method of claim 1, as shown above, and Safford further teaches if determined that a lockstep mismatch has not occurred between the lockstep pair of processors, then determining that the loss of lockstep is recoverable (see lines 30-63 of column 4).

Referring to claim 8, the combination of Safford and Bigbee teaches the method of claim 1, as shown above, and Safford further teaches if determined that a lockstep mismatch has occurred between the lockstep pair of processors, then determining that the loss of lockstep is not recoverable (see lines 10-20 of column 3, *"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."*).

Referring to claim 19, Safford teaches a system comprising a pair of lockstep processors (see figure 2), and logic comprising,

logic responsive to detection of loss of lockstep between the pair of lockstep processors, determining if the lockstep is recoverable comprising determining if a lockstep mismatch has occurred between the pair of lockstep processors (see lines 10-20 of column 3, "*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.*" See also lines 61-66 of column 3, "*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.*");

logic responsive to determining that the lockstep is recoverable, triggering the computer system to idle the processors (see lines 25-63 of column 4, "*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 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."),

attempting to recover lockstep between the pair of processors (see lines 54-57 of column 4, *"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)."*), and

if lockstep is successfully recovered between the pair of processors, recognizing the processors as being available for receiving instructions (see lines 25-63 of column 4, *"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." Note that just prior to the reboot the system operates without "hot standby" processors, then after the reboot processor 111/113 are recognized as the "hot standby" processors.*).

Safford does not use the terms "firmware" or "operating system", therefore does not specifically disclose the system comprising computer-executable firmware code stored to a computer-readable mediuem or using the firmware code to respond to the detection loss of lock step by determining if lockstep is recoverable for the lockstep pair of processors by determining if a lockstep mismatch has occurred. Safford also does not

specifically disclose the firmware code, in response to determining lockstep is recoverable, triggering an Operating System (OS) to idle the lockstep processors, attempting to recover lockstep for the lockstep pair of processors, and if successfully recovered, triggering the OS to recognize the processors as being available for receiving instructions.

Bigbee discloses a method for providing lock step protection in a system comprising multiple processors. Bigbee teaches using firmware to trigger an OS to idle the processors (see ¶ 20, "*…the processor error handler causes a platform error handler within a SAL associated with the package to be executed which initiates a platform-independent device removal sequence for a processor associated with the processor package by generating a system control interrupt (SCI) to operating system 106." See also ¶ 22, "Operating system 106 receives or detects the firmware-generated SCI and utilizes ACPI control methods to determine that the interrupt was caused by a request for processor removal or ejection." See also ¶ 23 "Operating system 106 next utilizes ACPI control methods to virtually eject the processor from data processing system 100… …In another embodiment, virtually ejecting the processor results in the physical removal of power from its associated processor package."*),

recover lockstep for the lockstep pair of processors (see ¶ 24, "*Firmware 104, via ACPI BIOS program code then resets the package to its original state at the time of operating system 106 initial load/boot. In alternative embodiments of the invention, this reset may comprise a "cold" reset or "RESET" operation or a "warm" or "INIT" reset operation. An associated SAL reinitializes the package and re-enables FRC mode lock step operation between two or more of the package's associated processor cores in response to detecting the ACPI BIOS-initiated reset operation.*"), and to

trigger the operating system to recognize the recovery (see ¶ 24, "*In yet another embodiment, the package's associated SAL initiates a platform-independent device insertion sequence for the processor package's associated processor by generating a system control interrupt (SCI) to operating system 106 via the previously disclosed INT_OUT command capability.*").

It would have been obvious to one of ordinary skill in the art at the time of the Patent Owner's invention to combine the teachings of Safford and Bigbee to such that Safford system uses firmware to trigger an operating system to idle the processors having the detected error, recover lockstep, and trigger the OS to recognize the recovery. One of ordinary skill at the time of the Patent Owner's invention would have been motivated to make

such modification because Bigbee suggests in advanced architecture computer system which is type of system disclosed by Safford, error resolution requiring processors be idled are handled with firmware to be platform independent (See ¶ 20).

Bigbee also teaches in response to detecting a loss of lockstep, using firmware to determine if lockstep is recoverable for the lockstep pair of processors, wherein said using said firmware to determine if lockstep is recoverable further comprises determining if a lockstep mismatch has occurred between the lockstep pair of processors (see ¶ 33-34, *"FRC logic 308 monitors processor core store operations or "writes" to shared cache 306 to detect inconsistencies in the generated results which indicate an FRC mismatch error. Such inconsistencies can occur as a result of partial or total hardware failure, or the effect of an alpha particle or other form of natural or artificial radiation impacting one or more processor cores… … In alternative embodiments, FRC logic 308 may be implemented as executable firmware or dedicated hardware coupled with or incorporated into one or more of the processor package's 204a components."*).

It would have been obvious to one of ordinary skill in the art at the time of the Patent Owner's invention to further modify the method of Safford with the above teachings of Bigbee

such that the system uses firmware to determine if lockstep is recoverable comprising determining if a lockstep mismatch has occurred. Such a modification would merely require substituting the hardware-implemented lockstep logic in Safford with the firmware-implemented FRC logic disclosed by Bigbee. One of ordinary skill would recognize that such substitution would have predictable result.

Referring to claim 20, the combination of Safford and Bigbee teaches the system of claim 19, as shown above, and Safford further teaches the code for attempting to recover lockstep further comprises code for resetting the pair of processors (see lines 57-60 of column 4, "*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.*").

Referring to claim 21, the combination of Safford and Bigbee teaches the system of claim 19, as shown above, and Bigbee further teaches the operating system is an Advanced Configuration and Power Interface (ACPI)-compatible operating system, said code for triggering the operating system to idle

the processors comprises: code for using an ACPI method to idle the processors (see ¶ 20, "*…the processor error handler causes a platform error handler within a SAL associated with the package to be executed which initiates a platform-independent device removal sequence for a processor associated with the processor package by generating a system control interrupt (SCI) to operating system 106." See also ¶ 22, "Operating system 106 receives or detects the firmware-generated SCI and utilizes ACPI control methods to determine that the interrupt was caused by a request for processor removal or ejection." See also ¶ 23 "Operating system 106 next utilizes ACPI control methods to virtually eject the processor from data processing system 100… …In another embodiment, virtually ejecting the processor results in the physical removal of power from its associated processor package."*).

It would have been obvious to one of ordinary skill in the art at the time of the Patent Owner's invention to modify the teachings of Safford with the above teachings of Bigbee because Bigbee teaches advanced architecture multiprocessor systems of the type disclosed by Safford typically have a ACPI compatible operating system in order to provide a layer of abstraction between the software applications the user interfaces with and the underlying hardware (see ¶¶ 14-16).

Referring to claim 22, the combination of Safford and Bigbee teaches the system of claim 19, as shown above, and Bigbee further teaches the operating system is an Advanced Configuration and Power Interface (ACPI)-compatible operating system, said code for triggering said operating system to recognize the processors as being available for receiving instructions comprises: code for using an ACPI method to cause the operating system to check for the processors (see ¶ 24, *"In yet another embodiment, the package's associated SAL initiates a platform-independent device insertion sequence for the processor package's associated processor by generating a system control interrupt (SCI) to operating system 106 via the previously disclosed INT_OUT command capability."*).

It would have been obvious to one of ordinary skill in the art at the time of the Patent Owner's invention to modify the teachings of Safford with the above teachings of Bigbee because Bigbee teaches advanced architecture multiprocessor systems of the type disclosed by Safford typically have a ACPI compatible methods called from firmware in order to provide a layer of abstraction between the software applications the user interfaces with and the underlying hardware (see ¶¶ 14-16).

Claims 9-12 is/are rejected under pre-AIA 35 U.S.C. 103(a) as being unpatentable over Safford as evidenced by Cepulis in view of Bigbee.

Referring to claim 9, the combination of Safford and Bigbee teaches the method of claim 1, as shown above, and Safford as evidenced by Cepulis further teaches determining if said lockstep pair of processors for which said loss of lockstep is detected comprises a system boot processor (see lines 52-54 of column 4, Note Safford teaches designating a new pair of processors as logical CPU 0. Cepulis discloses "logical CPU 0" is a term commonly used in the art to refer to the system boot processor in a multiprocessor system).

Referring to claim 10, the combination of Safford as evidenced by Cepulis in view of Bigbee teaches the method of claim 9, as shown above, and Safford further teaches if determined that said lockstep pair of processors for which said loss of lockstep is detected comprises a system boot processor, further comprising swapping the role of system boot processor to another processor in the system (see lines 52-54 of column 4,

"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.").

Referring to claim 11, the combination of Safford as evidenced by Cepulis in view of Bigbee teaches the method of claim 10, as shown above, and Safford further teaches after said swapping, recovering the lockstep for the lockstep pair of processors (see lines 54-60 of column 4, *"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."*).

Referring to claim 12, the combination of Safford as evidenced by Cepulis in view of Bigbee teaches the method of claim 11, as shown above, and Safford further teaches the method comprising after recovering lockstep for the lockstep pair of processors, holding the recovered pair of processors in idle as a spare for the processor to which the role of system boot processor was swapped (see lines 60-64 of column 4, *"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.").

Claims 13-18 are rejected under pre-AIA 35 U.S.C. 103(a) as being unpatentable over Safford in view of Marisetty.

Referring to claim 13, Safford teaches a system comprising:

master processor operable to process instructions (see item 111 in figure 2 and lines 55-60 of column 2) scheduled by the system;

slave processor lockstepped with the master processor operable to perform redundant processing of said instructions scheduled for the master processor (see item 111 in figure 2 and lines 55-60 of column 2); and

logic operable, responsive to detection of loss of lockstep between the master and slave processors, trigger the system to idle the master processor (see lines 25-63 of column 4, "*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 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."),

attempt to recover lockstep between the master and slave processors (see lines 54-57 of column 4, "*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).*"), and

if recovery of lockstep is successful to reintroduce the master processor to the operating system (see lines 25-63 of column 4, "*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." Note that just prior to the reboot the system operates without "hot standby" processors, then after the reboot processor 111/113 are recognized as the "hot standby" processors.*).

Safford does not specifically disclose an "ACPI compatible Operating system" or "firmware using ACPI methods", therefore does not specifically disclose firmware using ACPI method to trigger the Operating System (OS) to idle the lockstep processors, recover lockstep for the lockstep pair of

processors, or using ACPI methods to trigger the OS to reintroduce the master processor to the operating system.

Marisetty discloses that it is old and well known for a multiprocessor computer system to employ multiple layers of software including firmware and an ACPI compliant operating Operating System which operates to schedule commands for the processor (see paragraphs 6-7).

It would have been obvious to modify the teachings of Safford with the above teachings of Marisetty such that it includes firmware and an operating system because Marisetty discloses such software layering is useful as it allows for a user applications to use the system as if it were a single processor system (see paragraph 8).

Marisetty further teaches the firmware using ACPI methods to trigger an operating system to idle processors experiencing an error (see ¶ 34, see also ¶ 42), attempting to correct the error (see ¶ 34; see also ¶ 42), and if successfully recovering triggering an operating system to recognize the processors are available (see ¶ 34; see also ¶ 42).

It would have been obvious to one of ordinary skill in the art at the time of the Patent Owner's invention to modify the teachings of Safford with the above teachings Marisetty such

that an Operating system is triggered to idle the lockstep pair of processors having lost lock step, attempt to recover lockstep, and trigger the operating system to recognize the processors are available because Marisetty suggests these errors requiring a processor be idled before being corrected are best handled by firmware in conjunction with the operating system (see ¶ 34).

Referring to claim 14, the combination of Safford and Marisetty teaches the system of claim 13, as shown above, and Safford further teaches error detection logic for detecting loss of lockstep between the master and slave processors (see lines 10-20 of column 3, "*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.*").

Referring to claim 15, the combination of Safford and Marisetty teaches the system of claim 13, as shown above, and Safford further teaches error detection logic for detecting a precursor to loss of lockstep (see lines 21-51 of column 3, "*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.").

Referring to claim 16, the combination of Safford and Marisetty teaches the system of claim 15, as shown above, and Safford further teaches the precursor to loss of lockstep is treated as said detection of loss of lockstep (see lines 21-51 of column 3, *"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."*).

Referring to claim 17, the combination of Safford and Marisetty teaches the system of claim 15, as shown above and Safford further teaches said error detection logic comprises a parity-based mechanism (see lines 10-20 of column 3, *"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."*).

Referring to claim 18, the combination of Safford and Bigbee teaches the system of claim 13, as shown above, and Safford further teaches error detection logic for detecting lockstep mismatch between the master and slave processors (see lines 10-20 of column 3, *"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."*).

Claims 23-25 are rejected under pre-AIA 35 U.S.C. 103(a) as being unpatentable over Safford as evidenced by Cepulis in view of Marisetty.

Referring to claim 23, Safford as evidenced by Cepulis teaches a method comprising:

establishing, in a multi-processor system, a hot spare processor for a system boot processor (see lines 1-5 of column 3, "*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.*"; see lines 25-29 of column 4, "*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.*" While Safford does not use term "system boot processor", Safford does disclose identifying a processor as a logical CPU 0. Cepulis discloses a "logical CPU 0" is a designation given to a processor which is tasked with booting a multiprocessor system. Thus one ordinary skill in the art at the

time of the Patent Owner's invention would understand Safford's disclosure of a logical CPU 0 is a system boot processor.);

detecting loss of lockstep for a lockstep pair of processors in said multi-processor system (see lines 35-40 of column 4, *"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.""*);

determining if said lockstep pair of processors for which the loss of lockstep is detected is the system boot processor (see lines 52-54 of column 4, *"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."* Note the reassignment of logical CPU 0 (i.e. the system boot processor) is being interpreted as a detection of the failure occurring with the prior logical CPU 0).

if determined that said lockstep pair of processors for which the loss of lockstep is the system boot processor, copying a state of system boot processor to the hot spare processor (see lines 35-46 of column 4, *"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.”); and

if determined that said lockstep pair of processors for which the loss of lockstep is not the system boot processor (see lines 25-29 of column 4, “*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.”* Note that processors 121 and 123 are not designated as logical CPU 0 and would be replaced by the hot standby processors), then triggering the computer system to a) idle the lockstep pair of processors (see lines 25-63 of column 4, Note only the example of 111/113 is explicitly detailed but the recovery steps would be the same of all of the processors), b) attempt to recover lockstep between the lockstep pair of processors (see lines 25-63 of column 4), and c) if lockstep is successfully recovered between the lockstep pair of processors, trigger said computer system to recognize the processors as being available for receiving instructions (see lines 25-63 of column 4).

Safford fails to explicitly disclose an operating system, therefore doesn't specifically teach triggering an operating system to idle the lockstep pair of processors, attempt to recover lockstep between the lockstep pair of processors and recognize the processors as being available for receiving instructions.

Marisetty teaches, in an analogous system, triggering an operating system to idle processors experiencing an error (see ¶ 34, see also ¶ 42), attempting to correct the error (see ¶ 34; see also ¶ 42), and if successfully recovering triggering an operating system to recognize the processors are available (see ¶ 34; see also ¶ 42).

It would have been obvious to one of ordinary skill in the art at the time of the Patent Owner's invention to modify the teachings of Safford with the above teachings Marisetty such that an Operating system is triggered to idle the lockstep pair of processors having lost lock step, attempt to recover lockstep, and trigger the operating system to recognize the processors are available because Marisetty suggests these errors requiring a processor be idled before being corrected are best handled by firmware in conjunction with the operating system (see ¶ 34).

Referring to claim 24, Safford as evidenced by Cepulis in view of Marisetty teaches the method of claim 23, and Safford further teaches wherein if determined that said lockstep pair of processors for which the loss of lockstep is the system boot processor further comprising: attempting to recover lockstep between the lockstep pair of processors after copying the state of the system boot processor to the hot spare processor (see items 220 followed by item 225 in figure 4 and lines 40-60 of column 4).

Referring to claim 25, Safford as evidenced by Cepulis in view of Marisetty teaches the method of 24, as shown above, and Safford further teaches if determined that said lockstep pair of processors for which the loss of lockstep is the system boot processor further comprising: if lockstep is successfully recovered between the lockstep pair of processors, establishing the lockstep pair of processors for which lockstep was recovered as a hot spare processor (see lines 57-60 of column 4, "*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.*").

Submissions

In order to ensure full consideration of any amendments, affidavits or declarations, or other documents as evidence of patentability, such documents must be submitted in response to this Office action. Submissions after the next Office action, which is intended to be a final action, will be governed by the requirements of 37 CFR 1.116, after final rejection and 37 CFR 41.33 after appeal, which will be strictly enforced.

Amendments

Patent owner is notified that any proposed amendment to the specification and/or claims in this reexamination proceeding must comply with 37 CFR 1.530(d)-(j), must be formally presented pursuant to 37 CFR 1.52(a) and (b), and must contain any fees required by 37 CFR 1.20(c).

Extensions of Time

Extensions of time under 37 CFR 1.136(a) will not be permitted in these proceedings because the provisions of 37 CFR 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 CFR 1.550(a)). Extension of time in *ex parte* reexamination proceedings are provided for in 37 CFR 1.550(c).

Conclusion

All correspondence relating to this ex parte reexamination proceeding should be directed:

Electronically: Registered users may submit via Patent Center at https://patentcenter.uspto.gov/.

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 electronic transmissions, 37 CFR 1.8(a)(1)(i)(C) and (ii) states that 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 USPTO patent 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 should be directed to Primary Examiner Eron Sorrell at telephone number 571-272-4160.

Signed:

/ERON J SORRELL/
Primary Examiner, Art Unit 3992

Conferees: /JOSEPH R POKRZYWA/
Primary Examiner, Art Unit 3992

/MICHAEL FUELLING/
Supervisory Patent Examiner, Art Unit 3992