

Filed: August 30, 2024

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

BMW of North America, LLC,
Petitioner,

v.

Foras Technologies Ltd.,
Patent Owner.

Case IPR2024-01346
U.S. Patent 7,627,781

PETITION FOR *INTER PARTES* REVIEW

TABLE OF CONTENTS

I.	Relief Requested and Grounds	1
II.	Overview of the '781 Patent.....	2
III.	Level of Ordinary Skill.....	5
IV.	Statement of Facts.....	5
V.	Claim Construction.....	6
	A. “detecting loss of lockstep”.....	7
	B. “logic operable, responsive to detecting loss of lockstep for said boot processor, for transferring, during system runtime, the role of boot processor to said spare processor”.....	7
VI.	The Cited Art Is Analogous Art to the '781 Patent	7
VII.	Fox in View of Safford and Arai Renders Obvious Claim 1 (Ground 1)	9
	A. Overview of the Combination.....	9
	B. A POSITA Would Have Found It Obvious to Combine Fox, Safford, and Arai	10
	1. A POSITA would have found it obvious to implement Fox’s processors as lockstep processors with error detection logics that detect loss of lockstep.....	11
	2. Fox’s multiprocessor system would have one designated boot processor and replaced the failing boot processor with the spare processor.....	15
	3. A POSITA would have found it obvious to incorporate Arai’s processor-switching implementation details to minimize system interruption.	18
	C. Independent Claim 1	20

1.	[1preamble]: “A system comprising:”	20
2.	[1a]: “a plurality of processors;”	21
3.	[1b]: “data storage storing information that assigns a role of boot processor to one of said plurality of processors and assigns a role of spare processor to another of said plurality of processors; and”	22
4.	[1c]: “logic operable, responsive to detecting loss of lockstep for said boot processor, for transferring, during system runtime, the role of boot processor to said spare processor,”	24
5.	[1d]: “wherein said another of said plurality of processors that is assigned said role of spare processor is held in reserve as a hot spare processor until said loss of lockstep is detected for said boot processor.”	30
VIII.	Fox in View of Safford, Landry, and Arai Renders Obvious Claim 1 (Ground 2)	33
A.	Overview of the Combination	33
B.	A POSITA Would Have Found It Obvious to Combine Fox, Safford, Landry, and Arai	33
1.	A POSITA would have found it obvious to identify Fox’s boot processor as logical CPU0.	33
2.	A POSITA would have found it obvious to switch logical CPU0 (boot processor) to the spare processor in response to detecting loss of lockstep in the logical CPU0.	35
C.	Independent Claim 1	36
1.	[1preamble]	36
2.	[1a]	36
3.	[1b]	36
4.	[1c]	37

5.	[1d]	37
IX.	Fox in View of Safford and Arai (Ground 1) or Fox in View of Safford, Landry, and Arai (Ground 2) Renders Obvious Claims 2, 5-7, 9-18, 22-23, 25-26, and 29-30	38
A.	Independent Claim 9	38
1.	[9preamble]: “A method comprising:”	38
2.	[9a]: “assigning a role of boot processor to one of a plurality of processors in a system;”	38
3.	[9b]: “assigning a role of spare processor to another of said plurality of processors;”	38
4.	[9c]: “responsive to detecting loss of lockstep for said boot processor, changing, during system runtime, the assignment of the role of boot processor to said spare processor; and”	38
5.	[9d]: “responsive to changing the assignment of the role of boot processor to said spare processor, an operating system of the system recognizing said spare processor as the boot processor without requiring shutdown of said operating system.”	39
B.	Independent Claim 16	40
1.	[16preamble]: “A system comprising:”	40
2.	[16a]: “a plurality of lockstep processor modules that each includes a master processor and a slave processor that is in lockstep with said master processor;”	40
3.	[16b]: “a first communication bus with which at least a first two of said plurality of lockstep processor modules are communicatively coupled;”	43
	[16c]: “a second communication bus with which at least a second two of said plurality of lockstep processor modules are communicatively coupled; and”	43

4.	[16d]: “data storage storing information that assigns a role of boot processor to one of said plurality of processors and assigns a role of spare processor to another of said plurality of processors,”	45
5.	[16e]: “wherein said spare processor is communicatively coupled to the same one of said first and second communication buses as said boot processor.”.....	45
C.	Independent Claim 25	46
1.	[25preamble]: “A method comprising:”	46
2.	[25a]: “selecting a first one of a plurality of lockstep processor modules in a system as a boot processor,”	46
3.	[25b]: “wherein each of said plurality of lockstep processor modules includes a master processor and a slave processor that is in lockstep with said master processor, and”	47
4.	[25c]: “wherein said boot processor is communicatively coupled to a first of a plurality of communication buses;”	47
5.	[25d]: “selecting a second one of said plurality of lockstep processor modules as a spare processor,”	48
6.	[25e]: “wherein said spare processor is selected as one that is communicatively coupled to the same communication bus as said boot processor; and”	49
7.	[25f]: “responsive to detecting loss of lockstep for said boot processor, said system causing said spare processor to dynamically assume the role of boot processor.”	49
D.	Dependent Claim 2: “wherein said data storage is non-volatile data storage.”	50
E.	Dependent Claim 5: “wherein each of said plurality of processors is a processor module that includes a master processor and a slave processor.”	50

F.	Dependent Claim 6: “wherein said another of said plurality of processors that is assigned said role of spare processor is an active spare before being transferred said role of boot processor.”	50
G.	Dependent Claim 7: “an operating system, wherein, responsive to the logic transferring the role of boot processor to the spare processor, the operating system recognizes the spare processor as the boot processor.”	52
H.	Dependent Claims 10-11	52
I.	Dependent Claim 12	53
1.	[12a]: “determining at least one processor that is coupled to a common communication bus with said one of said plurality of processors assigned the role of boot processor; and”	53
2.	[12b]: “selecting said another of said plurality of processors to be assigned the role of spare processor from said determined at least one processor.”	55
J.	Dependent Claim 13: “during system boot-up, system firmware identifying the spare processor and holding said spare processor in reserve as a hot spare.”	55
K.	Dependent Claim 14: “during system boot-up, system firmware booting the spare processor as an active spare.”	56
L.	Dependent Claim 15: “system firmware performs said changing operation.”	56
M.	Dependent Claim 17: “system firmware operable, responsive to detecting loss of lockstep for said boot processor, changing the assignment of the role of boot processor to said spare processor.”	57
N.	Dependent Claim 18: “wherein said system firmware performs said changing during system runtime without requiring an operating system of said system to be shutdown.”	57

O.	Dependent Claim 22: “system firmware that, during system boot-up, identifies the spare processor and holds said spare processor in reserve as a hot spare.”	57
P.	Dependent Claim 23: “system firmware that, during system boot-up, boots the spare processor as an active spare.”	57
Q.	Dependent Claim 26: “wherein said system causing said spare processor to dynamically assume the role of boot processor comprises: said system causing said spare processor to dynamically assume the role of boot processor without shutting down an operating system of said system.”	58
R.	Dependent Claim 29: “storing information identifying said selected first one of said plurality of lockstep processor modules as said boot processor and identifying said selected second one of said plurality of lockstep processor modules as said spare processor.”	58
S.	Dependent Claim 30	58
	1. [30a]: “system firmware determining said spare processor from said stored information; and”	58
	2. [30b]: “said system firmware causing an operating system to recognize said spare processor as said boot processor.”	61
X.	Fox in View of Safford, Arai, and Jin (Ground 3) or Fox in View of Safford, Landry, Arai, and Jin (Ground 4) Renders Obvious Claims 3-4, 19-21, and 27-28	61
A.	Overview of the Combinations	61
B.	A POSITA Would Have Found It Obvious to Include a User Interface for Receiving Input Specifying the Selection of the Boot Processor and Spare Processor to Improve System Customizability and Availability	62
C.	Dependent Claims 3-4, 19-20, and 27-28	63

D.	Dependent Claim 21: “wherein said user interface presents for selection as said spare processor at least one processor that is communicatively to the same one of said first and second communication buses as said boot processor.”	64
XI.	Fox in View of Safford, Arai, and Bigbee (Ground 5) or Fox in View of Safford, Landry, Arai, and Bigbee (Ground 6) Renders Obvious Claims 26 and 27	65
A.	Overview of the Combinations	65
B.	A POSITA Would Have Found It Obvious to Configure Fox-Safford-Arai’s or Fox-Safford-Landry-Arai’s System with Itanium Family Processors as Taught in Bigbee to Improve System Performance and Capacity.....	65
C.	Dependent Claims 8 and 24	66
XII.	<i>Fintiv</i> Discretionary Denial Is Inappropriate.....	67
XIII.	Discretionary Denial Is Inappropriate Under 35 U.S.C. § 325(d).....	67
A.	The Grounds of This Petition Were Not Considered During Prosecution	67
B.	This Petition Is Not Cumulative of Unified Patents’ <i>Ex Parte</i> Reexamination.....	70
XIV.	Mandatory Notices.....	72
A.	Real Parties-in-Interest.....	72
B.	Related Matters.....	72
C.	Lead and Back-Up Counsel.....	73
D.	Service Information.....	74
XV.	Standing	74
XVI.	Conclusion	74

TABLE OF AUTHORITIES

	Page(s)
Cases	
<i>Advanced Bionics, LLC v. MED-EL Elektromedizinische Geräte GmbH,</i> IPR2019-01469, Paper 6 (PTAB Feb. 13, 2020).....	67, 68, 69
<i>Apple Inc. v. Fintiv, Inc.,</i> IPR2020-00019, Paper 11 (PTAB Mar. 20, 2020)	67
<i>Becton, Dickinson & Co. v. B. Braun Melsungen AG,</i> IPR2017-01586, Paper 8 (PTAB Dec. 15, 2017)	67
<i>Billerud Aktiebolag (publ) v. WestRock MWV, LLC,</i> IPR2023-01206, Paper 14 (PTAB Mar. 13, 2024)	69
<i>Peloton Interactive, Inc. v. Mad Dogg Athletics, Inc.,</i> IPR2022-00016, Paper 9 (PTAB Apr. 8, 2022)	69
<i>Phillips v. AWH Corp.,</i> 415 F.3d 1303 (Fed. Cir. 2005) (en banc)	6
Statutes	
35 U.S.C. § 102(a)	1
35 U.S.C. § 102(b)	1
35 U.S.C. § 102(e)	1
35 U.S.C. § 103	2, 70, 71, 72
35 U.S.C. § 314(a)	67
35 U.S.C. § 325(d)	67
Regulations	
37 C.F.R. § 42.22(c).....	5

TABLE OF EXHIBITS

Exhibit	Description
Ex. 1001	U.S. Patent No. 7,627,781 B2 to Michaelis et al. (“the ’781 Patent”)
Ex. 1002	Declaration of Dr. Michael C. Brogioli (“Brogioli”)
Ex. 1003	Curriculum Vitae of Dr. Michael C. Brogioli
Ex. 1004	Prosecution History of the ’781 Patent
Ex. 1005	US 2005/0172164 A1 (“Fox”), filed January 21, 2004, published August 4, 2005
Ex. 1006	US 2004/0006722 A1 (“Safford”), filed July 3, 2002, published January 8, 2004
Ex. 1007	US 2006/0036889 A1 (“Arai”), filed August 16, 2004, published February 16, 2006
Ex. 1008	IA-32 Intel® Architecture Software Developer’s Manual, Vol. 3: System Programming Guide (“IA-32”), publicly available at least as of February 14, 2002
Ex. 1009	Declaration of Nathaniel E. Frank-White regarding Internet Archive copy of “IA-32 Intel® Architecture Software Developer’s Manual, Vol. 3: System Programming Guide,” dated July 5, 2024
Ex. 1010	U.S. Patent No. 7,296,181 B2 to Safford, filed April 6, 2004, granted November 13, 2007 (“Safford-181”)
Ex. 1011	U.S. Patent No. 5,408,647 (“Landry”), filed October 2, 1992, granted April 18, 1995
Ex. 1012	US 2003/0126498 A1 (“Bigbee”), filed January 2, 2002, published July 3, 2003
Ex. 1013	U.S. Patent No. 5,491,788 (“Cepulis”), filed September 10, 1993, granted February 13, 1996

Ex. 1014	US 2003/0233492 A1 (“Schelling”), filed June 13, 2002, published December 18, 2003
Ex. 1015	U.S. Patent No. 6,189,112 B1 (“Slegel”), filed April 30, 1998, granted February 13, 2001
Ex. 1016	U.S. Patent No. 7,426,657 B2 (“Zorek”), filed July 9, 2004, granted September 16, 2008
Ex. 1017	WO 2001080007 A2 (“Suffin”), filed April 12, 2001, published October 25, 2001
Ex. 1018	U.S. Patent No. 6,108,781 (“Jayakumar”), filed February 23, 1999, granted August 22, 2000
Ex. 1019	US 2004/0221193 A1 (“Armstrong”), filed April 17, 2003, published November 4, 2004
Ex. 1020	US 2002/0144177 A1 (“Kondo”), filed January 31, 2002, published October 3, 2002
Ex. 1021	US 2004/0123201 A1 (“Nguyen”), filed December 19, 2002, published June 24, 2004
Ex. 1022	U.S. Patent No. 5,915,082 (“Marshall”), filed June 7, 1996, granted June 22, 1999
Ex. 1023	U.S. Patent No. 6,081,890 (“Datta”), filed November 30, 1998, granted June 27, 2000
Ex. 1024	U.S. Patent No. 5,784,599 (“Elkhoury”), filed December 15, 1995, granted July 21, 1998
Ex. 1025	US 2005/0086667 A1 (“Jin”), filed September 30, 2003, published April 21, 2005
Ex. 1026	L. Spainhower and T.A. Gregg, <i>IBM S/390 Parallel Enterprise Server G5 fault tolerance: A historical perspective</i> , IBM Journal of Research and Development, vol. 43, no. 5/6, pp. 863-873, September/November 1999, doi: 10.1147/rd.435.0863.

Ex. 1027	M. Mueller et al., <i>RAS strategy for IBM S/390 G5 and G6</i> , IBM Journal of Research and Development, vol. 43, no. 5/6, pp. 875-888, September/November 1999, doi: 10.1147/rd.435.0875.
Ex. 1028	MultiProcessor Specification (“MP Specification”), Version 1.4, May 1997
Ex. 1029	Advanced Configuration and Power Interface Specification, Revision 2.0, July 27, 2000 (“ACPI 2.0”)
Ex. 1030	Intel® Itanium® Architecture Software Developer’s Manual, Volume 2: System Architecture, Revision 2.1, October 2002
Ex. 1031	Prosecution History of <i>Ex Parte</i> Reexamination of the ’302 Patent, Control No. 90/019,243
Ex. 1032	Prosecution History of <i>Ex Parte</i> Reexamination of the ’781 Patent, Control No. 90/019,244
Ex. 1033	U.S. Patent No. 7,502,958 B2 to Michaelis et al. (“the ’958 Patent”), filed October 25, 2004, granted March 10, 2009
Ex. 1034	U.S. Patent No. 7,624,302 B2 to Michaelis et al. (“the ’302 Patent”), filed October 25, 2004, granted November 24, 2009
Ex. 1035	<i>Bayerische Motoren Werke AG v. Foras Techs. Ltd.</i> , Case No. 1:24-cv-363 (LMB/WEF), Dkt. 54 (E.D. Va. July 12, 2024) (“EDVA Stay Order”)
Ex. 1036	<i>Foras Techs. Ltd. v. Nissan Motor Co., Ltd.</i> , Case No. 1:23-cv-00640-RP, Dkt. 51 (W.D. Tex. May 21, 2024) (“WDTX Nissan Stay Order”)

I. Relief Requested and Grounds

BMW of North America, LLC (“Petitioner”) requests *inter partes* review and cancellation of claims 1-30 (“the Challenged Claims”) of U.S. Patent No. 7,627,781 (“the ’781 Patent”) based on:

Exhibit	Reference	Pre-AIA Prior Art
Ex. 1005	US 2005/0172164 A1 (“Fox”), filed January 21, 2004, published August 4, 2005	§ 102(e)
Ex. 1006	US 2004/0006722 A1 (“Safford”), filed July 3, 2002, published January 8, 2004	§ 102(a)
Ex. 1007	US 2006/0036889 A1 (“Arai”), filed August 16, 2004, published February 16, 2006	§ 102(e)
Ex. 1011	US 5,408,647 (“Landry”), filed October 2, 1992, issued April 18, 1995	§ 102(b)
Ex. 1025	US 2005/0086667 A1 (“Jin”), filed September 30, 2003, published April 21, 2005	§ 102(e)
Ex. 1012	US 2003/0126498 A1 (“Bigbee”), filed January 2, 2002, published July 3, 2003	§ 102(b)

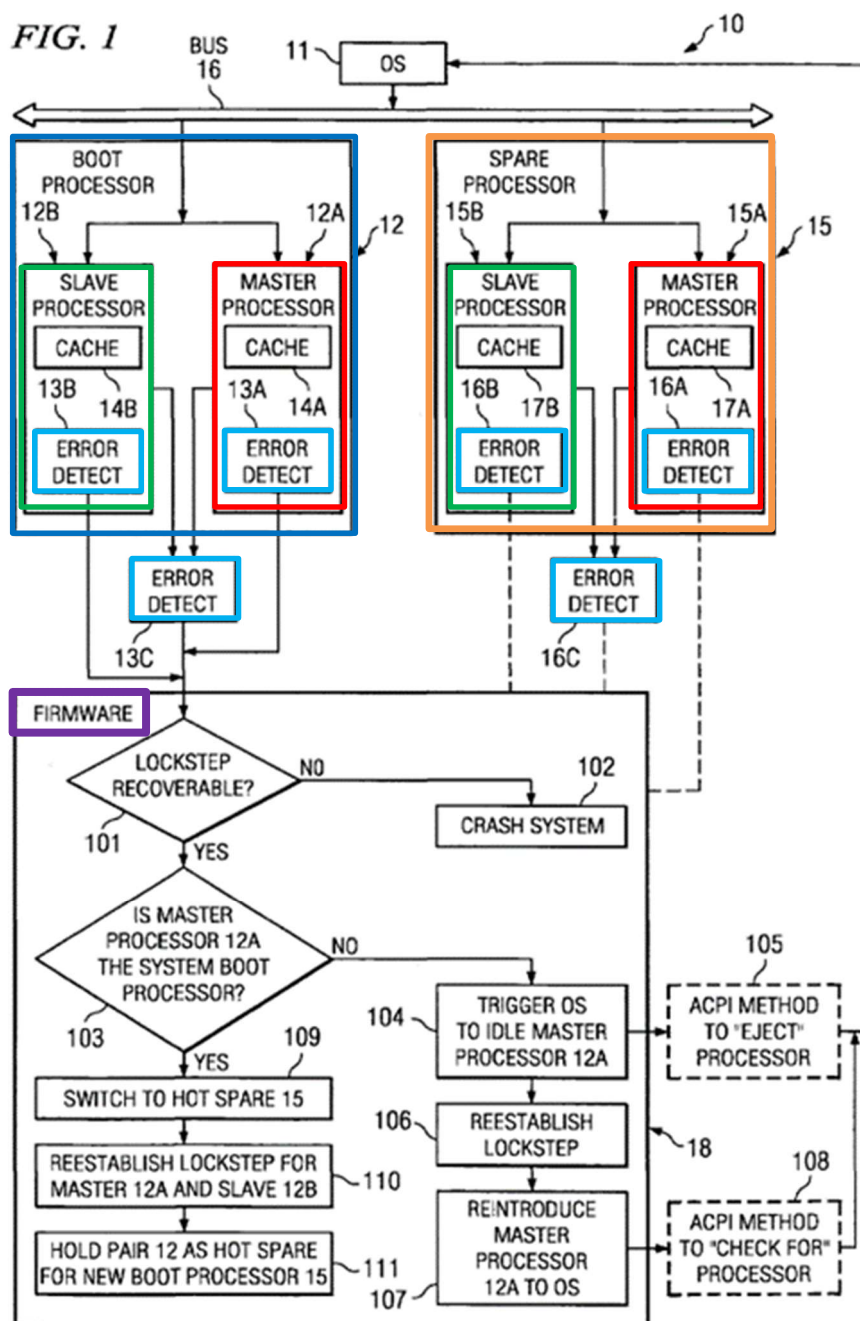
Claims 1-30 are unpatentable under the following grounds:

Ground	Claims	Reference(s)	Basis
1	1-2, 5-7, 9-18, 22-23, 25-26, 29-30	Fox in view of Safford and Arai	§ 103
2	1-2, 5-7, 9-18, 22-23, 25-26, 29-30	Fox in view of Safford, Landry, and Arai	§ 103
3	3-4, 19-21, 27-28	Fox in view of Safford, Arai, and Jin	§ 103
4	3-4, 19-21, 27-28	Fox in view of Safford, Landry, Arai, and Jin	§ 103
5	8, 24	Fox in view of Safford, Arai, and Bigbee	§ 103
6	8, 24	Fox in view of Safford, Landry, Arai, and Bigbee	§ 103

II. Overview of the '781 Patent

The '781 Patent claims priority to October 25, 2004. Ex. 1001, cover. The '781 Patent discloses a multiprocessor system that includes a **boot processor**, a **spare processor**, and processors other than the boot processor and spare processor. *Id.*, 3:62-4:9, 7:12-20, 12:21-34, 13:20-39. Each of the processors includes a lockstep processor pair, including a **master processor** and a **slave processor**, and **error detect**

logic to detect loss of lockstep (“LOL”) between the processor pair. *Id.*, 3:62-4:9, 6:10-26, 6:41-59, FIG. 1 (reproduced below). The multiprocessor system includes firmware that determines that the processor having LOL is the boot processor and switches the designation of the boot processor to the spare processor. *Id.*, 7:21-34, 7:51-53, 8:4-34, 9:18-10:12, 10:45-11:16, 11:44-59, 13:60-14:3, 14:65-15:20, 16:22-18:36, FIGS. 2-6.



Id., FIG. 1;¹ Brogioli, ¶¶55-57.

¹ Annotations added unless otherwise noted.

III. Level of Ordinary Skill

A person of ordinary skill in the art (“POSITA”) would have at least a bachelor’s degree in computer science, computer engineering, or equivalent, and at least two years of prior work experience with computer architecture design, including fault-tolerant computer architecture design, as of the earliest priority date of the ’781 Patent—October 25, 2004. Brogioli, ¶¶28-30. Additional education could substitute for professional experience and vice versa. *Id.*

IV. Statement of Facts

Pursuant to 37 C.F.R. § 42.22(c), Petitioner submits the following statement of facts. These facts are supported by Dr. Brogioli’s declaration that provides an overview of the state of the art and a background of the technology at issue. *Id.*, ¶¶39-54.

1. On or before October 25, 2004, using lockstep processors in a multiprocessor system was known. *Id.*, ¶¶39-43.

2. On or before October 25, 2004, it was known that a pair of lockstep processors are often differentiated by designating one as a master processor and the other as the slave processor. *Id.*, ¶42.

3. On or before October 25, 2004, it was known that a multiprocessor system includes a boot processor that performs system initialization. *Id.*, ¶¶44-47.

4. On or before October 25, 2004, it was known that a multiprocessor system configured with IA-32 architecture includes a boot processor. *Id.*, ¶¶45-46.

5. On or before October 25, 2004, it was known that a boot processor of a multiprocessor system could be identified as logical CPU0. *Id.*, ¶47.

6. On or before October 25, 2004, it was known to use a spare processor to recover a failing processor in a multiprocessor. *Id.*, ¶¶48-51.

7. On or before October 25, 2004, using both lockstep processors and spare processors in a multiprocessor system was known. *Id.*, ¶51.

8. On or before October 25, 2004, using firmware to switch the role of a failing processor to a spare processor during system runtime was known. *Id.*, ¶¶52-54.

9. On or before October 25, 2004, it was known that the BIOS of a computer system is firmware typically stored in a non-volatile memory. *Id.*, ¶52.

V. Claim Construction

The Board construes claims per *Phillips v. AWH Corp.*, according to their ordinary and customary meaning as understood by a POSITA in the context of the specification. 415 F.3d 1303, 1312-13 (Fed. Cir. 2005) (en banc). Aside from the terms addressed below, all terms are given their plain and ordinary meaning as understood by a POSITA and consistent with the specification. Brogioli, ¶¶31-38.

A. “detecting loss of lockstep”

The terms “detecting loss of lockstep” and “loss of lockstep is detected” should be construed to include both mismatch errors and errors that may eventually result in mismatch errors constituting a “precursor to lockstep mismatch”—including at least one of (1) detecting a lockstep mismatch error; and (2) detecting an error that may eventually cause a lockstep mismatch error. The specification provides detecting these two types of errors as examples of detection of LOL. *See e.g.*, Ex. 1001, 2:37-3:3, 6:60-7:4; Brogioli, ¶¶34-36.

This is consistent with the undisputed construction proposed by Petitioners in the IPR challenging U.S. Patent No. 7,502,958. *BMW of N. Am., LLC v. Foras Techs. Ltd.*, IPR2023-01373, Paper 18, 2 (PTAB July 12, 2024).

B. “logic operable, responsive to detecting loss of lockstep for said boot processor, for transferring, during system runtime, the role of boot processor to said spare processor”

This term should be construed to include “firmware logic” because the specification only discloses that the firmware assigns the role of boot processor to the spare processor responsive to detecting loss of lockstep for the boot processor. *See* Ex. 1001, 4:58-62, 8:32-34, 15:4-20, FIGS. 1-2; Brogioli, ¶37.

VI. The Cited Art Is Analogous Art to the ’781 Patent

Like the ’781 Patent, Fox, Safford, Arai, Landry, and Bigbee all relate to fault-tolerant systems that can operate despite the occurrence of processor error in a

multiprocessor system. **Fox's** multiprocessor system detects a processor failure and, in response to detecting the processor failure, replaces the failing processor with a spare processor. Fox, Abstract, [0006], [0014], [0016], [0019], [0023]-[0024], [0036], [0044], [0047], FIG. 3. **Safford's** multiprocessor system detects loss of lockstep in a pair of lockstep processors and, in response to detecting the loss of lockstep, replaces the lockstep processor pair in error with a spare processor pair. Safford, [0018]-[0021], FIGS. 3-4. **Arai's** multiprocessor system detects an error in an active processor and, in response to detecting the error, replaces the active processor in error with a reserved inactive processor. Arai, [0022]-[0023], [0025], FIGS. 4a-4b. **Landry's** system designates another processor as the boot processor to take the place of the originally designated boot processor if it fails. Landry, Abstract, 1:6-10, 2:9-41, 6:44-7:52, Figs. 1, 4. **Bigbee's** multiprocessor system includes a plurality of processors, each including two processor cores operating in Functional Redundancy Check ("FRC") or lockstep mode. Bigbee, [0026]-[0027], [0033], FIG. 2. Bigbee discloses detecting loss of lockstep in a lockstep pair of processors and recovering the loss of lockstep in response to the detection. *Id.*, [0018]-[0023], [0035]-[0037], FIG. 4A; Brogioli, ¶¶58-76, 79. **Jin's** multiprocessor computer system includes a boot processor that boots the operating system and input devices (e.g., touchscreen, keyboard, keypad, mouse) for selecting processors. Jin,

Abstract, [0014]-[0015], [0019], [0025], [0048]-[0049], [0061]; Brogioli, ¶¶78, 80.

All art is in the same field of endeavor as the '781 Patent. Brogioli, ¶¶58-81.

VII. Fox in View of Safford and Arai Renders Obvious Claim 1 (Ground 1)

A. Overview of the Combination

Fox discloses a multiprocessor system with firmware that detects a processor failure and switches the failing processor with a spare processor without interrupting the operating system. *See id.*, ¶¶58-62. Fox's multiprocessor system does not expressly include lockstep processors or detecting loss of lockstep, but Safford discloses a multiprocessor system including pairs of lockstep processors with error detection logics and switching a pair of lockstep processors detected to have loss of lockstep to a spare processor pair. *See id.*, ¶¶63-65.

A POSITA would have found it obvious to implement the processors of Fox's multiprocessor system as lockstep processor pairs with error detection logics, taught in Safford, to improve the reliability and availability of Fox's server system. Fox, [0016], [0044], [0047] (providing high availability of the system); Suffin, 6:4-12 (lockstep processors provide high level of availability); Safford, [0002] (lockstep processors enhance reliability of multiprocessor systems), [0017] (both internal and external error detection logics improve availability of processor assets); Brogioli, ¶¶85-86.

A POSITA would have understood that Fox's multiprocessor system would have a boot processor to initialize the system and load the operating system. *See Brogioli*, ¶¶44-47. A POSITA also would have understood that Fox's BIOS would have determined whether the failing processor is the boot processor. A POSITA also would have understood that Fox's spare processor would be used for replacing the boot processor in response to determining that the failing processor is the boot processor. A POSITA would have further understood that in Fox-Safford's system, detecting failure in a processor would include detecting loss of lockstep between the pair of lockstep processors. *Id.*, ¶87.

Arai discloses switching the role of a failing processor with a spare processor. Arai discloses switching the identifications between the failing processor and the spare processor, and transferring the context of the failing processor to the spare processor. *See id.*, ¶¶66-68. A POSITA would have understood that Arai provides implementation details for switching the failing processor and the spare processors, as suggested by Fox. *Id.*, ¶88.

B. A POSITA Would Have Found It Obvious to Combine Fox, Safford, and Arai

A POSITA would have found it obvious to combine Fox's multiprocessor system with Safford's lockstep processor and error detection logics and Arai's processor-switching implementation details. *Id.*, ¶¶89-112.

1. A POSITA would have found it obvious to implement Fox's processors as lockstep processors with error detection logics that detect loss of lockstep.

As Dr. Brogioli explains, it was well known and common to use lockstep processors in multiprocessor systems to improve system reliability and availability. *Id.*, ¶¶39-43. And it was also known to use both lockstep processors and spare processors in a fault-tolerant multiprocessor system. *See id.*, ¶51. A POSITA would have been motivated to implement Fox's processors as lockstep processors to improve reliability and availability of Fox's system for critical processes when higher reliability and availability of the system is desired. Fox, [0006], [0016], [0044], [0047]. A POSITA also would have been motivated to implement Fox with Safford's lockstep processors and error detection logics that detect loss of lockstep together to improve system availability. Safford, [0005], [0017]. A POSITA would have understood that when Fox's processors are implemented as lockstep processors, a loss of lockstep indicates processor failure to be detected and handled to achieve system high reliability and availability. Brogioli, ¶90.

A POSITA would have understood how to implement Fox's processors as lockstep processors with error detection logics and expected success. A POSITA would have applied well-known configurations and ordinary hardware and software configuration skills, like implementing Fox's processors as dual-core processors

with two processor cores operating in lockstep mode. *See* Bigbee, [0002]; Nguyen, [0020]. Alternatively, a POSITA also would have configured pairs of processors in Fox's system in lockstep mode and reserved at least one processor pair as spare processors. *See* Fox, [0028], [0034], [0040]; Safford, [0015], [0018], [0020]-[0021]; Ex. 1027, 5-6; Arai, [0019]. Indeed, a POSITA would have known that processors of a multiprocessor system configured with IA-32 architecture, like Fox's, would operate in lockstep mode with lockstep error detection. IA-32, 470-471; Suffin, 6:13-22; Elkhoury, 6:36-47; Brogioli, ¶¶91-92.

Fox also teaches other system resources can be added or removed. Fox, [0028]. A POSITA would have understood how to add the error detection logics (well known and taught by Safford) to improve system fault tolerance. Nguyen, [0021], [0028], [0032] (similar error detection logics); Bigbee, [0032]-[0033] (similar error detection logics); Brogioli, ¶93.

A POSITA would have understood that when implementing Fox's processors as lockstep processors with error detection logics, Fox's BIOS processor failure response logic would generate a systems management interrupt ("SMI") in response to the detection of the loss of lockstep by the error detection logics, as Fox teaches. Fox, [0032], [0041]; IA-32, 142, 269-270, 287-288. A POSITA also would have understood that when implementing Fox's processors as lockstep processors, each

lockstep processor pair would operate as a single logical processor having the same logical identification (well known and taught by Safford). Safford, [0015] (the processors 111 and 113 will appear to the computer system 100 to be a single processor core, or a logical CPU0); Bigbee, [0002], [0019], [0027], [0033], [0035] (two or more lockstep processor cores function as a single logical processor); Brogioli, ¶¶94-95.

Fox's multiprocessor system is configured with IA-32 architecture, which includes the APICs—each processor is connected to the APIC. Fox, [0024]-[0026], FIG. 2A. A POSITA would have understood that each APIC would be associated with a unique physical identification, e.g., the local APIC ID. IA-32, 277-278; MP Specification, 33, 43. A POSITA would have understood that in Fox-Safford's system, APIC-related identifications also would be available and both processors of a lockstep pair would share the same logical identification (e.g., logical APIC ID) as taught in Safford. *See* IA-32, 273-274, 294, 307; Brogioli, ¶96.

A POSITA would have further understood that the processor logical identification (e.g., logical APIC ID) of each processor would be stored in the register or table of the BIOS that is known to the operating system and BIOS after system initialization. *See* Brogioli, ¶46. A POSITA would have understood that the processor logical identification would be used for Fox's interrupt communication.

Fox, [0026], [0030] (all interrupt transactions on the system bus contain target node and IA-32 processor identification); *see also* IA-32, 267, 269, 289-295; Brogioli, ¶¶96-97.

A POSITA would have understood that the processor logical identification (e.g., logical APIC ID) would be used for switching processors transparent to the operating system, as desired by Fox. Fox, [0035], [0041], [0045]; Arai, [0023], [0025], FIGS. 4a-4b (switching logical APIC IDs of the failing processor and the reserved processor); Slegel, 3:22-40 (the spare processor assuming the logical identifier of a checkstopped processor is critical to make the processor sparing operation appear transparent to the operating system); Ex. 1027, 7-8 (“Once the replacement for the clock-stopped [processing unit] is identified, the physical/logical ID fields in the configuration area are updated.”); Brogioli, ¶98.

A POSITA would have found it obvious to implement Fox’s processors as lockstep processors with error detection logics, taught by Safford, according to their known functions. And a POSITA would have understood that in Fox-Safford’s system, a loss of lockstep for a lockstep processor pair would have been detected as a processor failure triggering processor switching. Such modification merely involves using a known technique to improve similar devices in the same way taught, within the technical grasp of a POSITA. Brogioli, ¶99.

2. Fox's multiprocessor system would have one designated boot processor and replaced the failing boot processor with the spare processor.

Fox discloses a multiprocessor system configured with IA-32 architecture. Fox, Abstract, [0014]-[0015]. Therefore, Fox would have designated one of its processors as the boot processor responsible for initializing the multiprocessor system and booting the operating system. IA-32, 238-239; *see* Brogioli, ¶¶44-47, 61-62. Fox expressly discloses system BIOS boot, which would be performed by a boot processor. Fox, [0006], [0034], Claims 4-6. Thus, a POSITA would have understood that Fox's system would have designated, or it would have been obvious to designate, one of Fox's processors as the boot processor. Brogioli, ¶100.

Fox suggests and a POSITA would have understood that Fox's spare processor would be used for replacing the boot processor when failure in the boot processor is detected to improve system reliability. *Id.*, ¶¶101-105.

First, a POSITA would have understood that Fox's spare processor would be used for any of the other main processors, including the designated boot processor and application processors. Once the operating system is up and running, the boot processor functions as an application processor. MP Specification, 75; Brogioli, ¶46. Thus, a POSITA would have understood that both the boot processors and application processors would be active processors in Fox, and Fox's spare processor

would replace any failing processor. *See* Fox, [0032]-[0033], [0035], [0045]; Brogioli, ¶102.

Second, Fox's BIOS would have determined that the failing processor is the designated boot processor or assigned the role of the processor. Fox, [0042]. Fox's SMI's identification of the failing processor to the BIOS would have included a unique physical or logical identification of the processor. *Id.*, [0030], [0042]; IA-32, 277-278, 255, 294; MP Specification, 33, 43. Fox's BIOS would know the unique identification of each processor stored in the BIOS register or table (MP Specification, 75, 79; IA-32, 277; Fox, [0035]), and would have determined whether the failing processor is the boot processor through the boot processor's unique identification in the SMI. Brogioli, ¶103.

Third, Fox's BIOS would have replaced a failing boot processor with the spare processor to ensure performance of boot processor responsibilities during system runtime to improve system reliability. It was known that the boot processor controls various system functions after the initial boot process, and is kept as the last running processor during system shutdown. MP Specification, 75 ("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."), 79. A POSITA would have been motivated to promptly replace the failing processor that

is assigned the role of boot processor with the spare processor. This would allow the responsibilities of the boot processor to continue to be performed during system runtime. A POSITA would have understood that such prompt replacement would improve the reliability of the system by maintaining the uptime of proper functions of the boot processor, reduce system interruption during system runtime, and improve system reliability during system runtime, shutdown, reset, and restart. Brogioli, ¶¶104-105.

A POSITA would have expected success in replacing the failing boot processor with the spare processor in Fox. Fox's SMI's identification of the failing processor to the BIOS and the operating system would have identified the assigned boot processor based on the unique identification of the boot processor (e.g., APIC ID or logical APIC ID). Also, a POSITA would have understood that the BSP flag would be readily examined by the BIOS and operating system to identify the failing processor. IA-32, 238, 276. Indeed, a POSITA would have been motivated to determine whether the failing processor is the designated boot processor so the system would know whether to set the BSP flag of the spare processor to 1 during the processor switching. Such determination and BSP flag setting would ensure a processor performs the responsibilities of the boot processor during system runtime to improve system stability. Brogioli, ¶106.

3. A POSITA would have found it obvious to incorporate Arai's processor-switching implementation details to minimize system interruption.

Fox's processor identifications are saved in a register or table (e.g., BIOS table, ACPI table, MP table), and all interrupt transactions include processor identifications. Fox, [0032]-[0033], [0035], [0038]. Fox does not explicitly disclose how the processor identifications would be used in the switching process. Arai discloses such implementation detail. Brogioli, ¶107.

Arai discloses switching a failing processor with a spare processor by switching the logical identifications of the failing processor and the spare processor and transferring the context of the failing processor to the spare processor. Arai [0020]-[0022]. This process was common and well known. *See* Brogioli, ¶¶48-51; Slegel, 3:22-40, 6:16-43. Incorporating Arai's teaching into Fox-Safford's system simply involves the addition of a well-known and suitable implementation detail in Fox's processor-switching process. Brogioli, ¶108.

A POSITA would have been motivated to incorporate Arai's teaching of transferring processor context from the failing processor to the spare processor not to interrupt the operating system and its running applications as desired by Fox. Fox teaches during the SMI, saving the system state of each processor to memory, flushing the caches and pipelines, and switching the memory save state from the

failing processor to the spare processor. Fox, [0041]. Similarly, Arai teaches transferring the execution context from the active processor with the error to the spare processor. Arai, [0022], [0025]. The execution context includes information in the processor registers and other data that support communication between the processor and the operating system, such as processor identification and instructions. *Id.*, [0022]; Slegel, 6:21-43. A POSITA would have understood that both the failing processor's identification and context/state would be transferred to the spare processor so the instructions and processes previously performed in the failing processor can be continued without system interruption. This would allow the role of the failing processor to be switched to the spare processor. Brogioli, ¶109.

A POSITA would have understood that incorporating Arai's processor-switching implementation details into Fox-Safford would provide the benefit of switching the role of the failing processor to the spare processor without affecting the operating system and running applications, as expressly taught as desirable by Fox and Arai and known in the art. Fox, [0008], [0013]-[0014], [0036] (allowing the operating system to continue despite a failing processor during dynamic replacement); Arai, [0005] ("it is very desirable not to lose the work being done" on the processor that encounters an error), [0006] ("There is therefore a need to provide a method and system for transparently reassigning responsibilities of a failed

processor to a reserved processor without a modification to the operating system.”); Slegel, 3:7-20 (moving state of failed processor to spare processor in the system transparently and dynamically); *see* Brogioli, ¶¶52-54. A POSITA would have understood that transparent switching would improve availability of Fox-Safford’s system because the processes of the operating system would continue despite processor failure (e.g., loss of lockstep). Brogioli, ¶110.

A POSITA would have understood how to implement Arai’s processor-switching details in Fox-Safford’s system and would have expected success. Fox’s multiprocessor system is equipped for such modification; it includes the hardware (e.g., APIC structure) and firmware (e.g., BIOS, BIOS interrupt functions and handlers, BIOS table, ACPI table, or MP table) used by Arai’s switching process. Implementing Arai’s switching detail in Fox-Safford’s system would have been applying a known technique to a similar device in the same way to achieve predictable results of processor switching with minimal disruption to the system. Fox, [0008], [0010], [0013], [0036]; Brogioli, ¶¶48-54, 111-112.

C. Independent Claim 1

1. [1preamble]: “A system comprising:”

If the preamble is limiting, Fox discloses a “system for dynamically replacing a failing processor” with a spare processor. Fox, Abstract, [0002], [0014], [0023]-[0024]; Brogioli, ¶114.

2. [1a]: “a plurality of processors;”

Fox discloses a multiprocessor computer system configured with IA-32 architecture, the system including a spare processor and main processors (“a plurality of processors”), as shown below. Fox, [0019], [0023]-[0025]; Brogioli, ¶¶59-61, 115-116.

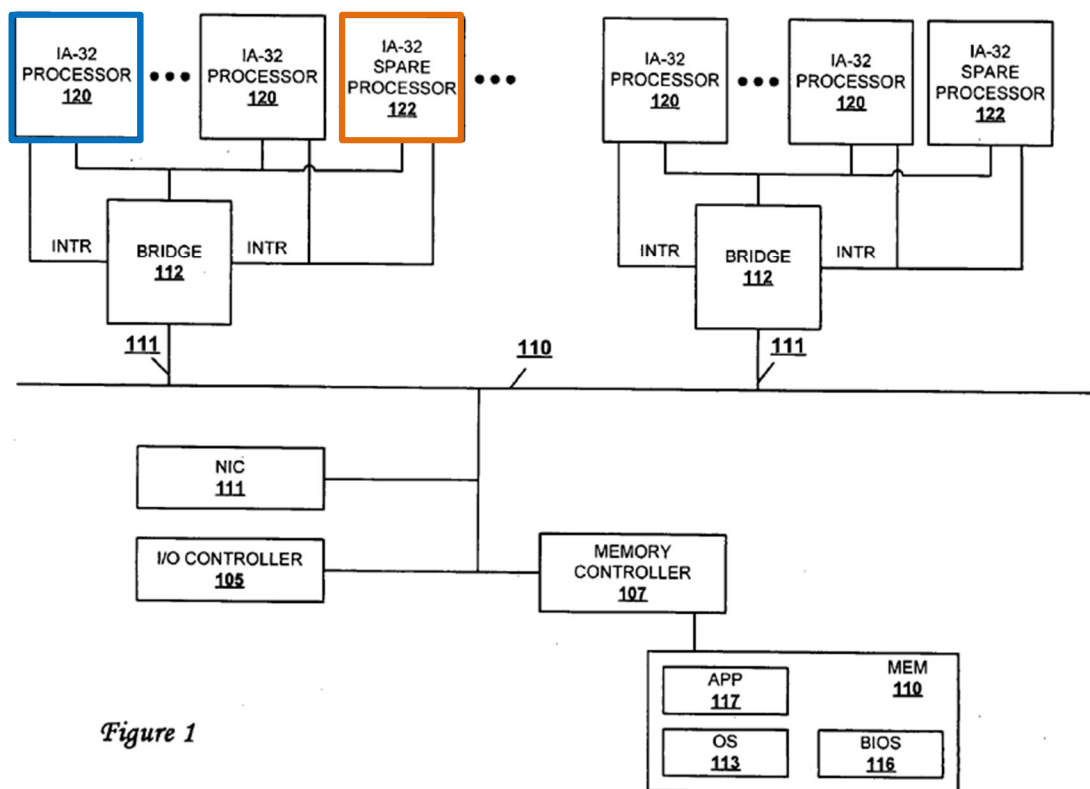


Figure 1

Fox, FIG. 1.

3. [1b]: “data storage storing information that assigns a role of boot processor to one of said plurality of processors and assigns a role of spare processor to another of said plurality of processors; and”

As discussed in Section VII.B.2, a POSITA would have understood that Fox’s system would have designated one of its processors as a boot processor. Fox discloses system BIOS boot, which a POSITA would have understood to be performed by a boot processor. Fox, [0006], [0034], Claims 4-6. And a POSITA would have understood that Fox’s IA-32 system would have designated one of its processors as the boot processor responsible for initializing the multiprocessor system and booting the operating system. IA-32, 238-239. A POSITA would have understood that Fox’s IA-32 system would have set the BSP flag of the designated boot processor to 1 (“assigns a role of boot processor”). *See* Brogioli, ¶¶44-46, 118.

Fox also discloses reserving, during system initialization, at least one of the processors of the multiprocessor system as a spare processor and setting a software tag to identify a reserved processor as a spare processor (“assigns a role of spare processor”). Fox, [0015], [0034]; *see* Brogioli, ¶¶59-61, 119.

Fox saves processor identifications in using a register or table (e.g., BIOS table, ACPI table, MP table), and including processor identifications in all interrupt transactions. Fox, [0026], [0033], [0035], [0038], [0041], [0045]-[0046], Claims 2, 13. A POSITA would have understood that in an IA-32 multiprocessor system, like

that in Fox, the identifications and software flags of processors, including those of the boot processor and the spare processor (“information”), would be stored in Fox’s register or table created by the boot processor and accessible to the BIOS and operating system. IA-32, 244, 256, 709-727; MP Specification, 38, 69, 72-73, 76; Brogioli, ¶120.

While Fox discloses a system memory for storing the BIOS, Fox does not expressly disclose the processor register or table is stored in the system memory. Fox, [0028], FIG. 1. Arai discloses this. Brogioli, ¶¶67, 121-122.

Arai discloses that during system initialization, the BIOS creates a table (e.g., ACPI table or MP table) that identifies each processor in the system and its designated operational state, i.e., active or disabled. Arai, [0019]-[0022], [0025]-[0026]. Arai discloses the table is stored in a main memory (“data storage”). *Id.* Both Fox and Arai disclose using registers or tables to store processor information available to the OS and BIOS and use the stored information to implement the switching. A POSITA would have found it suitable to store Fox’s register or table that stores information assigning the roles of boot and spare processors in its system memory (“data storage”), like in Arai. Brogioli, ¶122.

A POSITA would have found such modification routine and ordinary because it was well known that the processor register or table should be stored and accessible

to the BIOS. *See* IA-32, 242 (saving ACPI and MP tables in RAM); MP Specification, 38 (MP configuration table must be stored either in a non-reported system RAM or within the BIOS read-only memory space); Brogioli, ¶123.

The '781 Patent similarly discloses a device tree or ACPI table that is stored in a RAM, accessible to the firmware, and designates which processor is the boot processors. Ex. 1001, 8:43-54, 12:59-65, 13:20-44; Brogioli, ¶124.

4. [1c]: “logic operable, responsive to detecting loss of lockstep for said boot processor, for transferring, during system runtime, the role of boot processor to said spare processor,”

First, Fox in view of Safford renders obvious “detecting loss of lockstep for said boot processor.” Brogioli, ¶¶126-129.

As discussed in Section VII.B.2 and [1b], a POSITA would have understood that Fox’s multiprocessor system would have designated a boot processor, and that Fox would have replaced the failing boot processor with the spare processor. Fox does not expressly disclose that its processors are lockstep processors or detecting loss of lockstep (“LOL”). Safford does. *Id.*, ¶¶64-65, 127-129.

Safford discloses a multiprocessor computer system 100, wherein each of the processors includes a pair of processor cores 111 and 113 operating in lockstep. Safford, Abstract, [0015], FIGS. 2-3. Safford’s lockstep processors each include a logic circuit 115 to detect differences in the outputs of the pair of processor cores

indicating a loss of lockstep and error detection, and signaling logic 112 and 114 to detect errors in the respective processor cores to signal an impending loss of lockstep. *Id.*, [0015]-[0017]. Safford also discloses a fault recovery process that replaces the lockstep pair of processors 111 and 113 with a spare processor pair in response to detection of a loss of lockstep error by the error detection logics. *Id.*, [0007], [0018], [0021], FIGS. 3-4; Brogioli, ¶128.

100

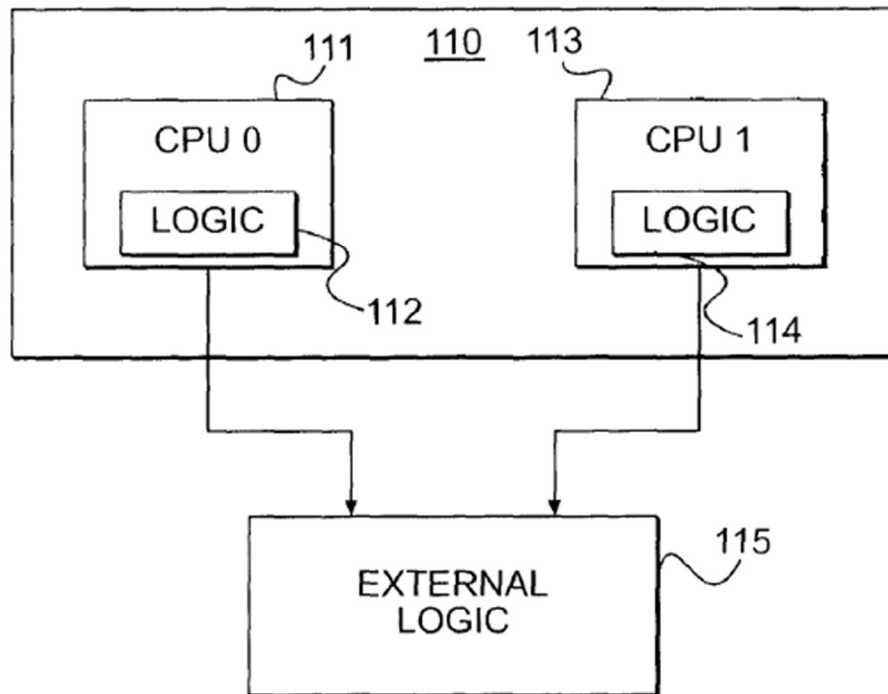


FIG. 2

Safford, FIG. 2.

As explained in Section VII.B.1, a POSITA would have been motivated to implement Fox's processors as lockstep processors with error detection logics as taught by Safford to improve the reliability of Fox's server system and expected success. Safford, [0003]. A POSITA would have further understood that in Fox-Safford's system, Fox's spare processor would be implemented as a lockstep pair to replace the boot processor pair for which said LOL is detected. *See* Sections VII.B.1-VII.B.2; Brogioli, ¶129.

Second, Fox in view of Safford renders obvious "logic operable, responsive to detecting loss of lockstep for said boot processor" Brogioli, ¶¶130-132.

As discussed in Section V.B, the term "logic" in this limitation should be construed to include "firmware logic." Fox discloses using the BIOS of its IA-32 multiprocessor system to implement switching of a failing processor to the spare processor. Fox discloses when a failure is detected in one of its main processors, the BIOS's processor failure response logic generates an SMI in response to detecting the processor failure to facilitate the processor replacement. Fox, [0015], [0032]. The SMI identifies to the OS and the BIOS the processor that is failing, and the BIOS, utilizing the SMI handler, takes control of the system and replaces the failing processor with the spare processor. *Id.*, [0041]-[0042], [0048]. A POSITA would have understood Fox's BIOS implementing processor switching as "[firmware]

logic” because a POSITA would have understood that the BIOS is firmware that acts as an interface between the system’s hardware and operating system and contains program logic. Brogioli, ¶131.

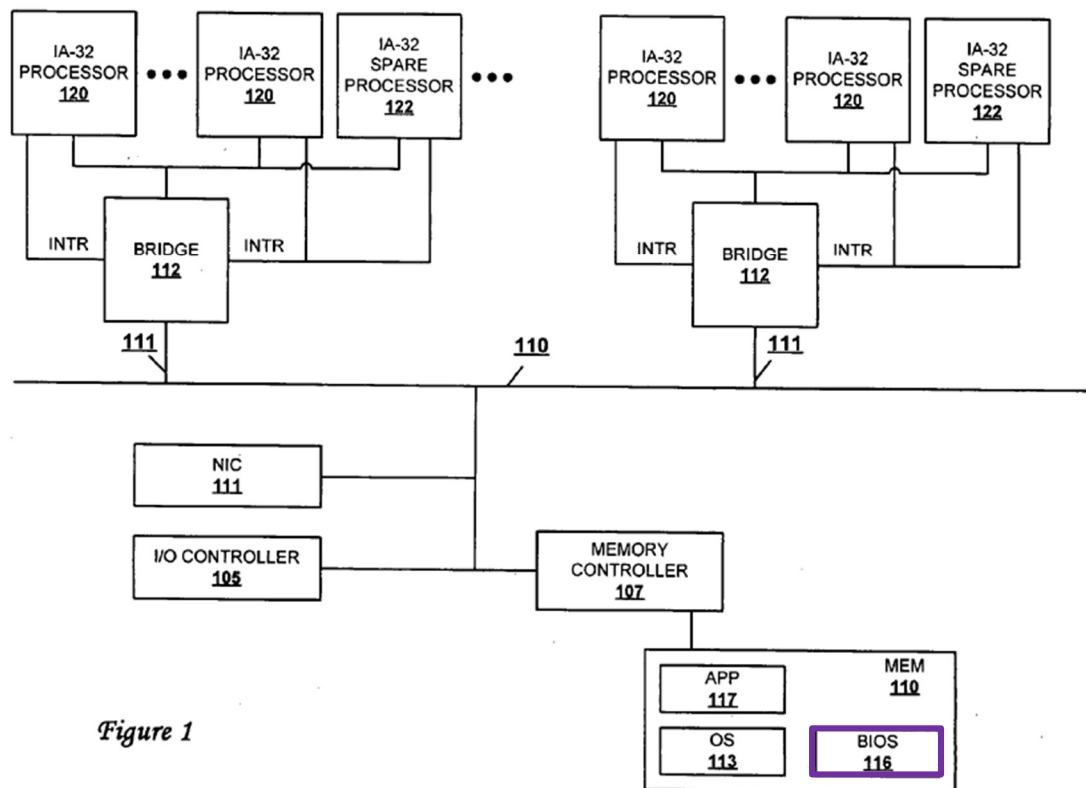


Figure 1

Fox, FIG. 1.

A POSITA would have understood that in Fox-Safford, detecting a processor failure in the boot processor would have included detecting loss of lockstep for the boot processor, and Fox’s BIOS-implemented processor switching would be “responsive to detecting loss of lockstep” for the boot processor. *See* Sections VII.B.1-VII.B.2; Brogioli, ¶132.

Third, Fox in view of Safford and Arai renders obvious “transferring, during system runtime, the role of boot processor to said spare processor.” Brogioli, ¶¶133-138.

Fox’s processor-switching process dynamically replaces the failing processor with the spare processor while allowing the system to continue executing operating system and application processes (“during system runtime”). Fox, [0014]. Fox discloses that, during the processor switching, the executing threads/processes are transferred to the replacement processor, and the memory save state is switched from the failing processor to the spare processor. *Id.*, [0014], [0041]; Brogioli, ¶133.

Arai discloses replacing an active processor detected with an error with a reserved inactive processor in a multiprocessor system. Arai, Abstract, [0002], [0008]-[0009], [0019]-[0024]. Arai discloses that during system initialization, the BIOS creates a table that identifies each processor in the system and its designated operational state, i.e., active or disabled. *Id.*, [0019]-[0022]. If the BIOS determines the error requires replacement of the processor, and if there is a reserved inactive processor, the identification numbers of the active processor in error and the reserved inactive processor are switched. *Id.*, [0022]-[0023], [0025].

Arai discloses switching the logical APIC IDs of the active processor in error with the reserved processor. *Id.*, [0023]-[0024], FIGS. 4a-4b. A POSITA would have

understood that Arai's logical APIC IDs would be used by Arai's operating system to recognize the processor in error. *Id.*, [0020]-[0022], [0025]-[0026]. A POSITA would have further understood that such switching would have changed the logical APIC ID of the processor in error to the logical APIC ID of the reserved processor. *Id.*; Brogioli, ¶134.

Arai further discloses "transfer[ring] of the execution context from the active processor that is experiencing the error to at least one of the inactive processors that has been designated to replace the active processor." Arai, [0022], [0025], FIG. 4b. Arai discloses that "[t]he execution context may include information in the processor registers and other data that supports communication between the processor and the operating system." *Id.*, [0022]; Brogioli, ¶135.

As discussed in Section VII.B.3, a POSITA would have been motivated to incorporate Arai's processor identification switching and transferring the execution context in Fox-Safford's switching process to allow the processes previously performed by the failing processor to be continued by the spare processor without interrupting the operating system. Fox, [0008], [0013], [0036]; Arai, [0006]; Slegel, 3:7-20; Brogioli, ¶¶48-51, 136.

A POSITA would have understood that in Fox-Safford-Arai, the processor identification switching would have switched the logical identification (e.g., logical

APIC ID) of the lockstep processor pair of the boot processor detected to have LOL with that of the spare processor pair. *See* Safford, [0021] (the spare processor becomes the new logical CPU0); *see* Section VII.B.1. A POSITA also would have understood that in Fox-Safford, implementing Arai's identification switching and context transfer in the switching process would have reassigned the processor logical identification and responsibilities of the failing boot processor pair to the spare processor pair ("transferring ... the role of boot processor to said spare processor") in response to detecting loss of lockstep for the boot processor while the operating system processes continue to execute ("during system runtime"). Fox, [0008], [0010], [0013]-[0014]; Arai, [0006]; Brogioli, ¶137.

A POSITA would have understood that incorporating Arai's processor-switching implementation detail would improve availability and reliability of the multiprocessor system in Fox-Safford and expected success because it uses known techniques to improve a similar device in the same way taught. *See* Section VII.B.3; Brogioli, ¶138.

5. **[1d]: "wherein said another of said plurality of processors that is assigned said role of spare processor is held in reserve as a hot spare processor until said loss of lockstep is detected for said boot processor."**

Fox discloses that 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

(“held in reserve as a hot spare processor”), and while in that mode, the OS is prevented from initially utilizing the hot-spare processor. Fox, Abstract, [0015], [0040], FIG. 1; Brogioli, ¶140.

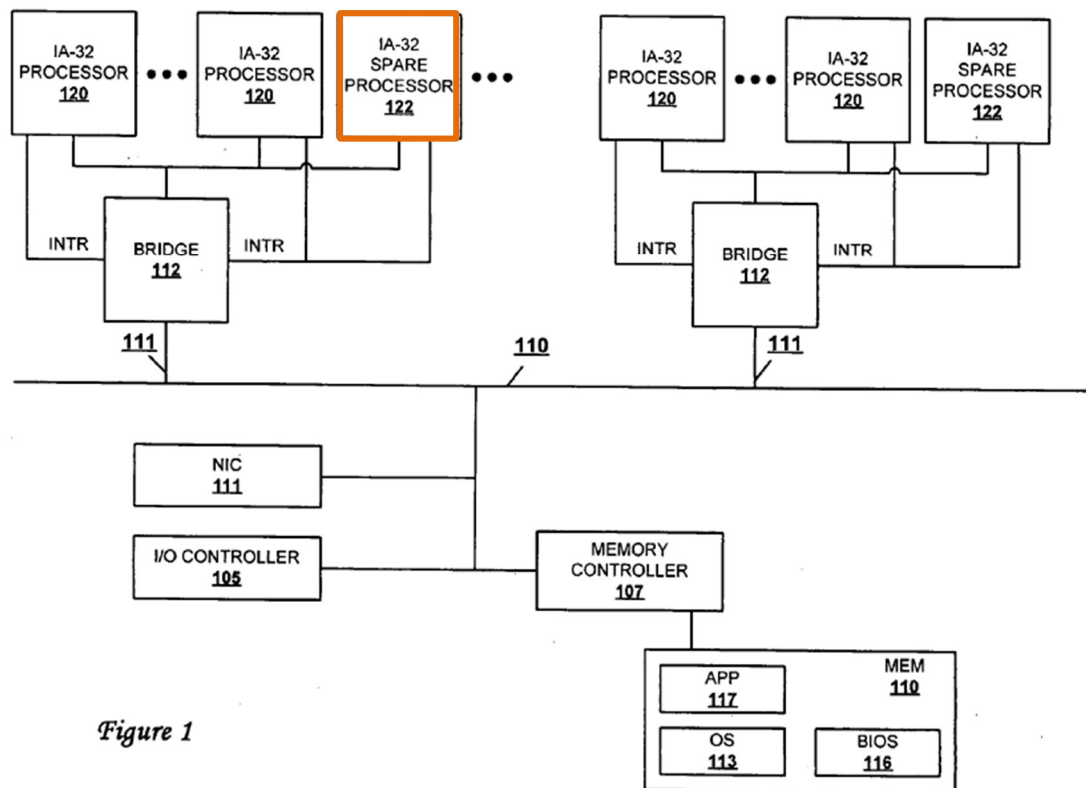


Figure 1

Fox, FIG. 1.

Fox discloses when a processor failure is detected, SMI activates and completes an initialization of the hot-spare processor and the BIOS switches the failing processor to the active hot-spare processor. *Id.*, [0040]-[0041]; Brogioli, ¶141.

Likewise, Safford discloses that the spare processor pair 125/127 is designated as “a hot standby” (“hot spare”) and “sitting idle in lock step mode” (“held in reserve”). Safford, [0020]. Safford discloses when loss of lockstep is detected in the processor pair 111/113, the architecture state of processor pair 111/113 is copied to the hot standby processor pair 125/127, which becomes an active processor. *Id.*, [0018], [0020]-[0021]; Brogioli, ¶142.

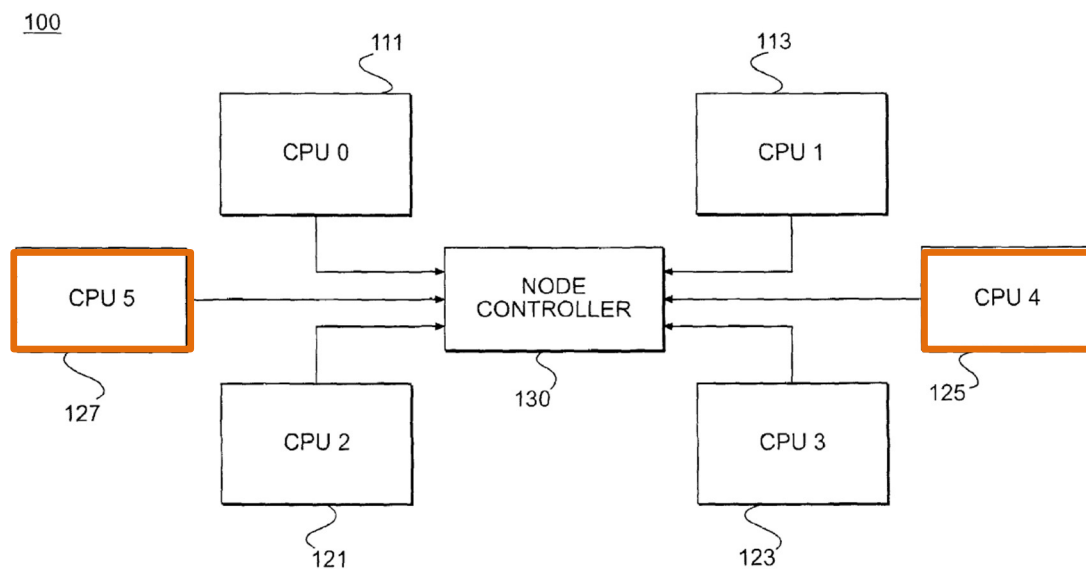


FIG. 3

Safford, FIG. 3.

A POSITA would have understood that in Fox-Safford, the hot-spare processor would be a hot-spare processor pair “held in reserve” in lockstep mode until loss of lockstep is detected for the boot processor pair, as taught in Fox and Safford, and as discussed in [1b] and [1c]. Brogioli, ¶143.

VIII. Fox in View of Safford, Landry, and Arai Renders Obvious Claim 1 (Ground 2)

A. Overview of the Combination

The combination of Fox, Safford, and Arai renders obvious every element of claim 1. For reasons discussed in Section VII.B.2, a POSITA would have understood that Fox's system would have designated a boot processor and used the spare processor to replace the boot processor in response to detecting failure in the boot processor. And as discussed in Ground 1, [1b]-[1c], Fox-Safford-Arai renders obvious "data storage storing information that assigns a role of boot processor" and "logic operable, responsive to detecting loss of lockstep for said boot processor, for transferring, during system runtime, the role of boot processor to said spare processor." These are rendered obvious further in view of Landry. Brogioli, ¶145.

B. A POSITA Would Have Found It Obvious to Combine Fox, Safford, Landry, and Arai

A POSITA would have found it obvious to combine Fox, Safford, Landry, and Arai for the reasons discussed in Ground 1, Section VII.B and the additional reasons below. *Id.*, ¶¶146-150.

1. A POSITA would have found it obvious to identify Fox's boot processor as logical CPU0.

A POSITA would have understood that Fox's system would have designated a boot processor. *See* Section VII.B.2. Fox's multiprocessor system configured with

IA-32 architecture follows the MP initialization protocol that dynamically designates one of its processors as the boot processor. Fox, [0014]; IA-32, 238; *see* Brogioli, ¶¶45-46. Landry's fault-tolerant process dynamically designates a functional CPU as logical CPU0 that performs the unique functions of a boot processor. Landry, Abstract, 1:6-10, 2:9-41, 6:44-7:42, Figs. 1, 4; *see* Brogioli, ¶¶70-71. A POSITA would have understood Landry's process as a suitable option for selecting a boot processor in Fox's system consistent with the initialization protocol of an IA-32 multiprocessor system. Brogioli, ¶147.

A POSITA would have understood the benefits of implementing Landry's process in Fox. **First**, it would improve system availability by allowing the system to power up despite initially starting on a non-functional processor. Landry, 1:47-62, 2:9-41. **Second**, using the logical CPU0 as the identification for the boot processor allows for transparent replacement of the physical processor experiencing a failure with another processor without interrupting the operating system. Slegel, 3:7-46. Indeed, the logical CPU0 was a known and conventional identification of the boot processor. *See* Landry, Abstract, 1:6-10, 2:9-41, 6:44-7:52, Figs. 1, 4; Cepulis, Abstract, 2:30-64; Jin, [0048]; Brogioli, ¶¶47, 148.

A POSITA would have expected success in implementing Landry's process in Fox because Fox's IA-32 system provides the necessary hardware, firmware, and

data structures for implementing the boot processor selection. Identifying the boot processor as logical CPU0 would only require ordinary computer programming skills to assign logical CPU0 as the unique identification of the boot processor and store such identification in Fox's register or table. *See* Fox, [0026], [0033], [0035], [0038], [0041], [0045]-[0046]; IA-32, 294; MP Specification, 75, 79; Brogioli, ¶149.

2. A POSITA would have found it obvious to switch logical CPU0 (boot processor) to the spare processor in response to detecting loss of lockstep in the logical CPU0.

Safford's lockstep pair of processors appear to the system as a single logical CPU0, and in response to detecting loss of lockstep in the lockstep pair, the logical CPU0 is switched to a spare pair of processors. Safford, [0015], [0018]-[0021]. A POSITA would have found it obvious to switch Fox-Landry's logical CPU0 (boot processor) to the spare processor to improve system availability and reliability. *See* Section VII.B.2. A POSITA would have expected success in implementing the switching of logical CPU0 to the spare because such switching only requires ordinary computer programming skills to assign unique logical identifications to processors in Fox's system and update Fox's processor register or table storing the processor's identifications, as known in the art. *See* Fox, [0015], [0026], [0033], [0035], [0038], [0040]-[0041], [0045]-[0046]; Arai, [0019]-[0021], [0025]-[0026]; IA-32, 294; MP Specification, 75, 79; Brogioli, ¶150.

C. Independent Claim 1

1. [1preamble]

Fox discloses [1preamble]. *See* Section VII.C.1; Brogioli, ¶152.

2. [1a]

Fox discloses [1a]. *See* Section VII.C.2; Brogioli, ¶153.

3. [1b]

As explained in Ground 1, Section VII.B.2 and [1b], a POSITA would have understood that Fox’s system would have designated one of its processors as a boot processor (“assigns a role of boot processor”). Fox in view of Landry further renders obvious assigning a role of boot processor—logical CPU0—to one of said plurality of processors. As explained in Section VIII.B.1, a POSITA would have implemented Landry’s boot processor selection process in Fox and identified logical CPU0 in Fox as the boot processor (“assigns a role of boot processor”). Brogioli, ¶155.

For reasons explained in Ground 1, [1b], a POSITA would have understood that the identifications and software tags of the processors, including those of the boot processor and spare processor (“information”), would be stored in Fox’s register or table. A POSITA also would have found it suitable, routine, and ordinary to store Fox’s register or table that stores information assigning the roles of boot and spare processors in its system memory (“data storage”), like in Arai. *Id.*, ¶156.

4. [1c]

Fox-Landry-Safford-Arai renders obvious [1c] for the reasons explained in Ground 1, [1c] and the additional reasons discussed in Ground 2, [1b]. A POSITA would have understood that in Fox-Landry-Safford-Arai, detecting loss of lockstep for the boot processor would be detecting loss of lockstep for logical CPU0—Fox's boot processor—and that transferring the role of the boot processor to the spare processor would include switching the logical CPU0 of the boot processor to the spare processor, as taught in Safford and Arai. *See* Safford, [0018]-[0021] (detecting loss of lockstep in logical CPU0 and switching logical CPU0 to the spare processor), FIGS. 3-4; Arai, [0023] (assigning the spare processor from logical CPU₆ to logical CPU₀); Brogioli, ¶¶64-65, 67-68, 157.

5. [1d]

Fox-Landry-Safford renders obvious [1d] for the reasons explained in Ground 1, [1d], and the additional reasons discussed in Ground 2, [1b]. A POSITA would have understood that in Fox-Landry-Safford, the spare processor would be a lockstep pair of processors held in reserve until loss of lockstep is detected for logical CPU0—the boot processor. Safford, [0018]-[0021]; Brogioli, ¶158.

IX. Fox in View of Safford and Arai (Ground 1) or Fox in View of Safford, Landry, and Arai (Ground 2) Renders Obvious Claims 2, 5-7, 9-18, 22-23, 25-26, and 29-30

A. Independent Claim 9

1. [9preamble]: “A method comprising:”

If the preamble is limiting, Fox discloses “a method ... for dynamically replacing a failing processor” with a spare processor. Fox, Abstract, [0002], [0014], [0023], [0046]; Brogioli, ¶160.

2. [9a]: “assigning a role of boot processor to one of a plurality of processors in a system;”

Fox or Fox-Landry discloses or renders obvious [9a] for the reasons explained in Grounds 1-2, [1b]. Brogioli, ¶161.

3. [9b]: “assigning a role of spare processor to another of said plurality of processors;”

Fox discloses [9b] for the reasons explained in Grounds 1-2, [1b]. Brogioli, ¶162.

4. [9c]: “responsive to detecting loss of lockstep for said boot processor, changing, during system runtime, the assignment of the role of boot processor to said spare processor; and”

Fox-Safford-Arai or Fox-Safford-Landry-Arai renders obvious [9c] for the reasons explained in Grounds 1-2, [1c]. Brogioli, ¶163.

5. [9d]: “responsive to changing the assignment of the role of boot processor to said spare processor, an operating system of the system recognizing said spare processor as the boot processor without requiring shutdown of said operating system.”

Fox-Safford-Arai or Fox-Safford-Landry-Arai renders obvious [9d]. Brogioli, ¶¶164-166.

For reasons discussed in Ground 1-2, [1c] and [9c], Fox-Safford-Arai or Fox-Safford-Landry-Arai renders obvious changing, during system runtime, the assignment role of the boot processor to the spare processor, including switching the logical identification of the boot processor (e.g., logical CPU0) to the spare processor. A POSITA would have understood that in response to such logical identification switching, Fox’s operating system would recognize the spare processor as the boot processor because the unique logical identification of the boot processor (e.g., logical CPU0), as it was known to the BIOS and operating system, has been assigned to the spare processor. Slegel, 3:7-46 (the operating system’s view of the system is only through the logical identification for a processor, and the capability of the spare processor to assume the logical identification of a failing processor is critical to making the processor sparing operation appear transparent to the operating system); Brogioli, ¶¶44-47, 165.

A POSITA would have further understood that the switching of the logical identification of the boot processor to the spare processor allows the operating system to continue execution (“without requiring shutdown of said operating system”) because: (1) Fox’s BIOS (firmware) would perform the processor switching; (2) both Fox and Arai disclose that the BIOS performs the processor switching without interrupting the operating system; and (3) it was well known that using firmware to switch processor logical identifications would allow the operating system to continue execution to improve system availability despite processor failure. Fox, [0014], [0035], [0038], [0041]-[0044], [0047]; Arai, [0004]-[0006], [0023], [0025]; Slegel, 3:7-46, 6:16-36; Armstrong, Abstract, [0019]-[0020]; Zorek, Abstract, 2:61-3:17; Brogioli, ¶¶48-56, 166.

B. Independent Claim 16

1. [16preamble]: “A system comprising:”

Fox discloses [16preamble]. *See* Section VII.C.1; Brogioli, ¶167.

2. [16a]: “a plurality of lockstep processor modules that each includes a master processor and a slave processor that is in lockstep with said master processor;”

Fox in view of Safford renders obvious [16a]. Brogioli, ¶¶168-176.

Fox discloses a multiprocessor computer system configured with IA-32 architecture, the system including a spare processor and main processors (“a plurality of lockstep processor modules”). Fox, [0019], [0023]-[0025], FIG. 1.

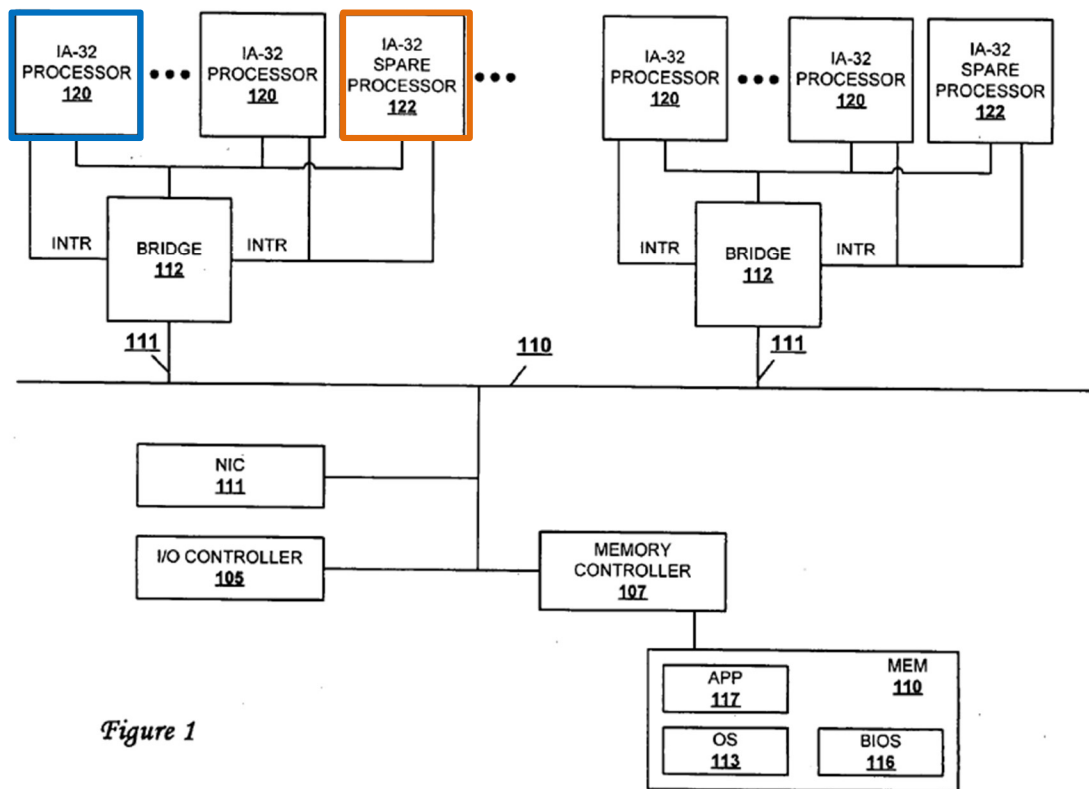


Figure 1

Fox, FIG. 1; Brogioli, ¶169.

As explained in Section VII.B.1, a POSITA would have found it obvious to implement Fox’s processors as lockstep processors with error detection logics as taught in Safford to improve system reliability and availability. Safford further discloses its lockstep processor pair “includes a master processor and a slave processor.” Brogioli, ¶¶170-175.

Safford discloses each of the processors 110 of a multiprocessor computer system 100 includes a pair of processor cores 111 and 113 operating in lockstep.

Safford, Abstract, [0015], [0018], FIGS. 2-3. Safford discloses the lockstep processor pair includes a master/checker pair. *Id.*, [0015]; Brogioli, ¶171.

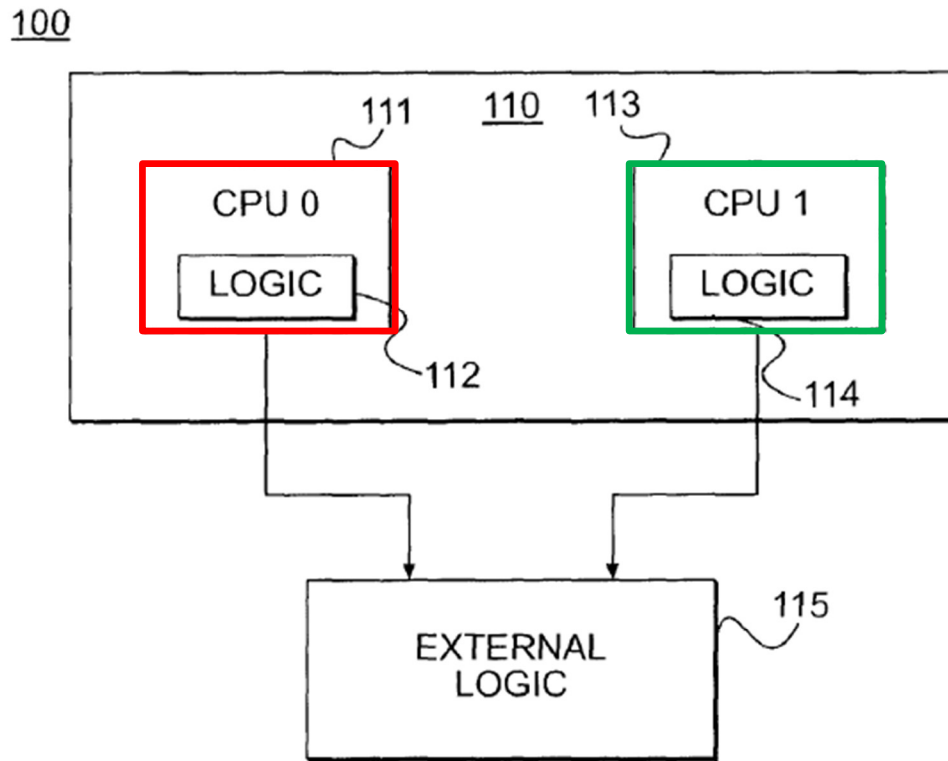


FIG. 2

Safford, FIG. 2.

Safford further discloses that “[e]ach of [the] two processors in the pair processes the same code sequences, and the resulting outputs of the processors are compared by a logic circuit,” and that difference in the processor outputs indicates loss of lockstep. *Id.*, [0015]. Safford’s lockstep processor’s master/checker pair

constitutes “a master processor and a slave processor that is in lockstep with said master processor.” Brogioli, ¶¶172-173.

As Dr. Brogioli explains, it was well known that a lockstep pair of processors operates as a single processor, where one of the processors (“master processor”) writes data to the memory and outputs to the system and the other processor (“slave processor”) performs redundant processing to check whether there is mismatch between the two processors indicating a processor error. *Id.*, ¶174 (citing Kondo, [0094]; Marshall, 1:11-20, 1:25-32; Nguyen, [0047]). A POSITA would have understood Safford’s master/checker pair operating in lockstep are a “master processor” and “slave processor,” but regardless, such designations would be readily implemented. Nguyen, [0027], [0054]-[0055]; Brogioli, ¶¶174-175.

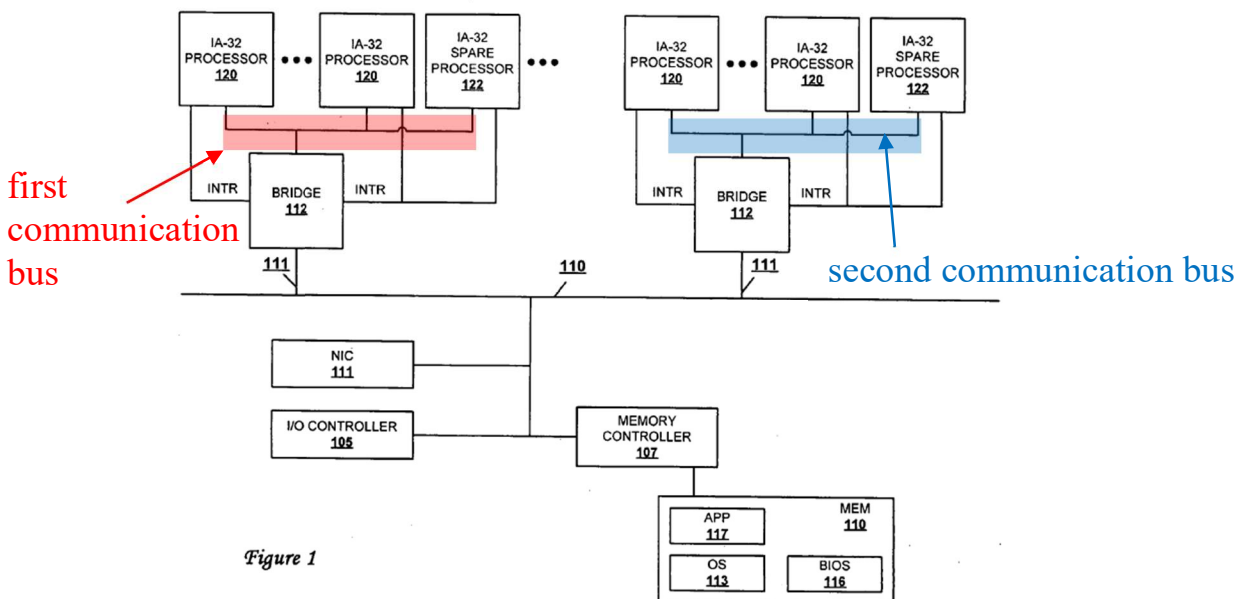
For reasons explained in Section VII.B.1, a POSITA would have found it obvious to implement the processors in Fox’s multiprocessor systems as lockstep processors, each including a master processor and a slave processor. Brogioli, ¶176.

3. [16b]: “a first communication bus with which at least a first two of said plurality of lockstep processor modules are communicatively coupled;”

[16c]: “a second communication bus with which at least a second two of said plurality of lockstep processor modules are communicatively coupled; and”

Fox discloses a multiprocessor system including two clusters of processors, each including at least one main IA-32 processor 120 and a spare processor 122.

Fox, [0024]-[0025], [0033], FIG. 1. Fox discloses processors 120 and 122 of each cluster share access to a common bus. *Id.* A POSITA would have understood that the common bus of the processors would handle communications (e.g., interrupts) from the system bus to the processors. *Id.*, [0030]-[0031]. A POSITA would have understood that processors 120 and 122 of the first cluster (“at least a first two of said plurality of lockstep processor modules”) are communicatively coupled to a first common bus (“a first communication bus”), and processors 120 and 122 of the second cluster (“at least a second two of said plurality of lockstep processor modules”) are commonly coupled to a second common bus (“a second communication bus”). Brogioli, ¶178.



Fox, FIG. 1.

As explained in Section VII.B.1 and Grounds 1-2, [1c], a POSITA would have found it obvious to implement Fox's processors as lockstep processors with error detection logics as taught in Safford to improve system reliability and availability. A POSITA would have understood that in Fox-Safford, each lockstep processor pair would operate as a single logical processor and be coupled to the common bus, for example, via the external logic circuit or a shared resource. Safford, [0015]; Bigbee, [0027], FIGS. 2-3; *see* Section VII.B.1; Brogioli, ¶179.

4. **[16d]: “data storage storing information that assigns a role of boot processor to one of said plurality of processors and assigns a role of spare processor to another of said plurality of processors,”**

Fox-Arai or Fox-Landry-Arai renders obvious [16d] for the reasons explained in Grounds 1-2, [1b]. Brogioli, ¶180.

5. **[16e]: “wherein said spare processor is communicatively coupled to the same one of said first and second communication buses as said boot processor.”**

Fox or Fox-Landry renders obvious [16e]. *Id.*, ¶¶181-183.

As discussed in [16b]-[16c], each of Fox's processor clusters includes a spare processor 122 communicatively coupled to the common bus (“first and second communication buses”) shared with the other processors 120. For reasons discussed in Grounds 1-2, [1b], one of processors 120 of one of the clusters would have been designated as the boot processor. Thus, Fox's spare processor 122 of the cluster that

includes the designated boot processor would have been communicatively coupled to the same communication bus as the boot processor. *Id.*, ¶182.

A POSITA would have used the spare processor that is communicatively coupled to the same communication bus as the boot processor to replace a failing boot processor. This would facilitate processor switching to minimize interruption to the system. Fox, [0041]; Brogioli, ¶183.

C. Independent Claim 25

1. [25preamble]: “A method comprising:”

Fox discloses [25preamble]. *See* Section IX.A.1; Brogioli, ¶184.

2. [25a]: “selecting a first one of a plurality of lockstep processor modules in a system as a boot processor,”

Fox-Safford or Fox-Safford-Landry renders obvious [25a]. Brogioli, ¶¶185-187.

As discussed in Section VII.B.1, a POSITA would have found it obvious to implement Fox’s processors as lockstep processor pairs with error detect logics as taught in Safford to improve the reliability and availability of Fox’s system. *Id.*, ¶186.

For reasons discussed in Sections VII.B.2 and VIII.B.1, a POSITA would have understood that Fox-Safford’s system would have selected one of its lockstep

processors as a boot processor, and Fox-Landry-Safford renders obvious selecting one of the lockstep processors as logical CPU0—the boot processor. *Id.*, ¶187.

3. **[25b]: “wherein each of said plurality of lockstep processor modules includes a master processor and a slave processor that is in lockstep with said master processor, and”**

Fox-Safford renders obvious [25b] for the reasons explained in [16a]. *Id.*, ¶188.

4. **[25c]: “wherein said boot processor is communicatively coupled to a first of a plurality of communication buses;”**

Fox or Fox-Landry renders obvious [25c]. *Id.*, ¶¶189-190.

As explained in [16b]-[16c], Fox discloses two processor clusters, wherein processors of each cluster are communicatively coupled to a shared communication bus of that cluster. And as discussed in [16e], Fox or Fox-Landry renders obvious the boot processor being communicatively coupled to the same communication bus as the spare processor, as shown below. *Id.*, ¶190.

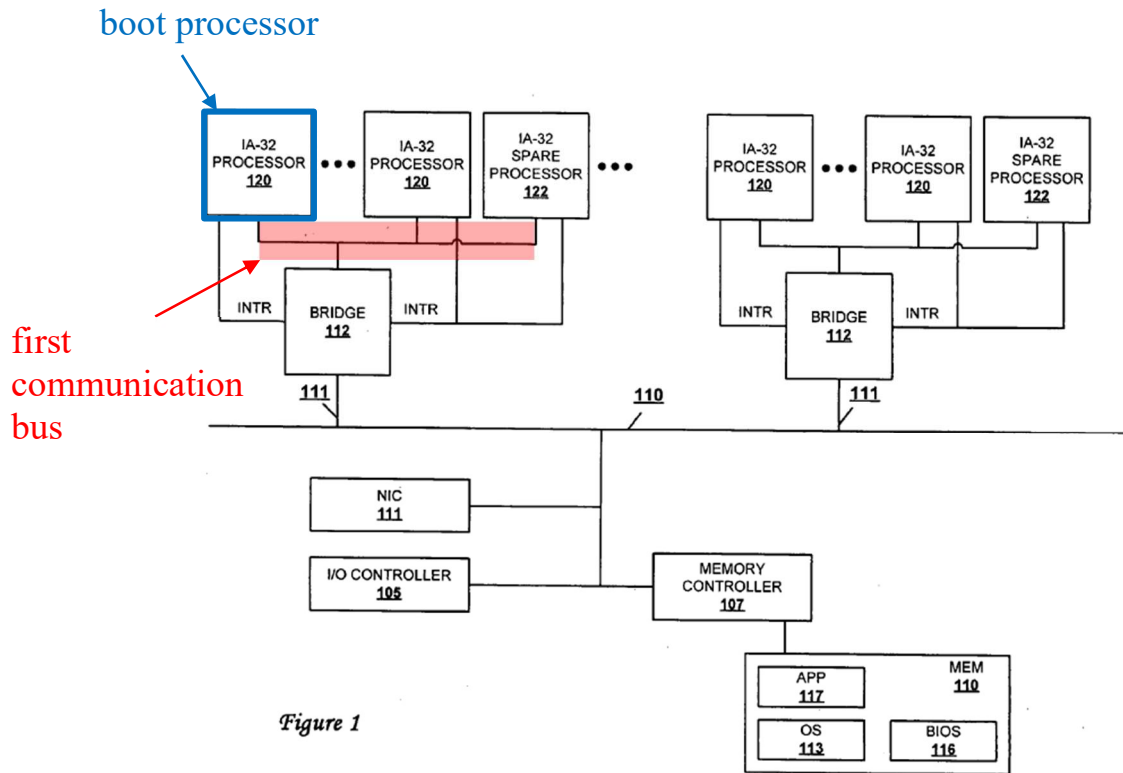


Figure 1

Fox., FIG. 1.

5. [25d]: “selecting a second one of said plurality of lockstep processor modules as a spare processor,”

Fox-Safford renders obvious [25d]. Brogioli, ¶¶191-193.

As discussed in Grounds 1-2, [1b], Fox discloses reserving, during system initialization, at least one of the processors of the multiprocessor system as a spare processor and setting a software tag to identify a reserved processor as a spare processor (“selecting ... a spare processor”). Fox, [0015], [0034]; Brogioli, ¶¶58-62, 192.

And as discussed in Section VII.B.1, a POSITA would have implemented Fox's processors as lockstep processors with error detection logics as taught by Safford, and would have understood that the spare processor would be a lockstep pair of processors. Safford, [0018]-[0021]; Brogioli, ¶193.

6. **[25e]: “wherein said spare processor is selected as one that is communicatively coupled to the same communication bus as said boot processor; and”**

Fox or Fox-Landry renders obvious [25e] for reasons explained in [25d] and [16e]. Brogioli, ¶194.

7. **[25f]: “responsive to detecting loss of lockstep for said boot processor, said system causing said spare processor to dynamically assume the role of boot processor.”**

Fox-Safford-Arai or Fox-Safford-Landry-Arai renders obvious [25f]. *Id.*, ¶¶195-196.

As discussed in Grounds 1-2, [1c], Fox-Safford-Arai or Fox-Safford-Landry-Arai renders obvious, responsive to detecting loss of lockstep for said boot processor, transferring, during system runtime, the role of boot processor to the spare processor. Fox discloses its processor-switching process dynamically replaces the failing processor with the spare processor while allowing the system to continue executing operating system and application processes. Fox, [0013]-[0014], [0021], [0041]. A POSITA would have understood that, in Fox-Safford-Arai or Fox-Safford-Landry-Arai, transferring the role of the boot processor to the spare processor during

system runtime would have caused the “spare processor to dynamically assume the role of boot processor.” Brogioli, ¶196.

D. Dependent Claim 2: “wherein said data storage is non-volatile data storage.”

As discussed in Grounds 1-2, [1b], Fox-Arai renders obvious storing Fox’s register or table, which stores information assigning the roles of boot and spare processors, in its system memory (“data storage”). A POSITA would have understood Fox’s system memory would be “non-volatile” because it was common and well known that the BIOS would be embedded in a non-volatile memory (e.g., ROM). *See* Fox, [0028], FIG. 1; MP Specification, 38 (storing MP configuration table within the BIOS read-only memory space); Brogioli, ¶¶197-198.

E. Dependent Claim 5: “wherein each of said plurality of processors is a processor module that includes a master processor and a slave processor.”

Fox-Safford renders obvious this limitation for the reasons explained in [16a].
Brogioli, ¶199.

F. Dependent Claim 6: “wherein said another of said plurality of processors that is assigned said role of spare processor is an active spare before being transferred said role of boot processor.”

Fox discloses “enabl[ing] the ‘hot spare processor’ feature on-the-fly by the OS with a driver-OS-SMI handshake” and “[t]racking which processors are active.” Fox, [0035]. A POSITA would have understood Fox’s configuring a processor “on-

the-fly” to mean the configuration is performed dynamically as needed while the operating system is running. A POSITA would have understood that Fox discloses or renders obvious configuring one of the active processors to be the spare processor when a spare processor is needed for recovering a failing processor. A POSITA would also have understood such active processor to be “an active spare” because it would have been initialized as an active processor with a software tag indicating it is a spare processor. *Id.*, [0034]; Brogioli, ¶201.

A POSITA would have been motivated to use “an active spare” because allowing the spare processor to be available to the operating system as an active spare improves the processing capacity of the system when no processor failure occurs. A POSITA would have understood how to configure an active processor to a spare processor using ordinary computer programming skills and expected success.

Fox discloses the processor initialization/identification code includes a software tag that can be set to identify the processor as a spare processor during system BIOS boot. Fox, [0034]. Instead of reserving the spare processor in an idle state during initialization, a POSITA would have initialized the active spare processor like other active processors in the system. When a processor failure is detected, the state of the active spare would have been set to idle to be switched with the failing processor, as taught in Fox. A POSITA would have known that various

software tools and the data structure of Fox's system would be readily available to allow for dynamic idling of the active spare for replacing the failing processor, including the SMI and BIOS, the operating system, and IA-32 processor register, table, and flags. *See* IA-32, 276-277, 279, 304-305; MP Specification, 44; Brogioli, ¶202.

A POSITA would have understood that in Fox-Safford, the active spare processor would have included a lockstep pair of processors as taught in Safford to replace the lockstep pair of processors detected to have loss of lockstep. Safford, [0020]-[0021], FIG. 4; Brogioli, ¶203.

G. Dependent Claim 7: “an operating system, wherein, responsive to the logic transferring the role of boot processor to the spare processor, the operating system recognizes the spare processor as the boot processor.”

Fox-Safford-Arai or Fox-Safford-Landry-Arai renders obvious claim 7 for the reasons discussed in Grounds 1-2, [1c] and [9d]. Brogioli, ¶204.

H. Dependent Claims 10-11

Claim 10 recites “wherein said assigning said role of boot processor comprises storing information to non-volatile data storage that identifies said one of said plurality of processors as said boot processor.” Claim 11 is similar to claim 10 except reciting “spare processor” instead of “boot processor” and reciting “another of” instead of “one of.” Fox-Arai or Fox-Landry-Arai renders obvious the limitations of

these claims for the reasons explained in Grounds 1-2, [1b] and claim 2. *Id.*, ¶¶205-206.

I. Dependent Claim 12

- 1. [12a]: “determining at least one processor that is coupled to a common communication bus with said one of said plurality of processors assigned the role of boot processor; and”**

Fox or Fox-Landry renders obvious [12a] for the reasons explained in [16b]-[16c] and [16e] and the additional reasons below. *Id.*, ¶¶208-213.

As discussed in [16b]-[16c] and [16e] and shown below, a POSITA would have understood that Fox discloses at least one main IA-32 processor and the **spare processor** (“at least one processor”) are coupled to the **same communication bus** (“a common communication bus”) and the main IA-32 processors would have included a designated **boot processor** (“one of said plurality of processors assigned the role of boot processor”). *Id.*, ¶209.

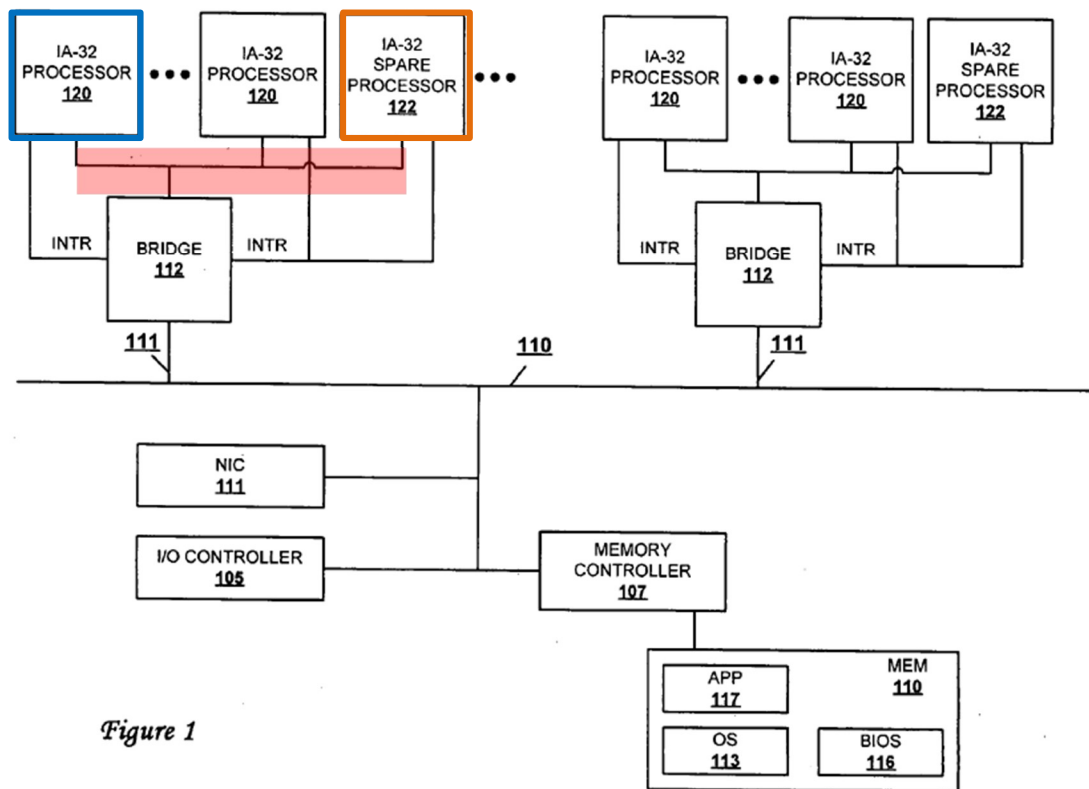


Figure 1

Fox, FIG. 1.

A POSITA would have found it obvious to determine and select the processor to be assigned the role of spare processor during system initialization in Fox. Brogioli, ¶¶210-213.

Fox's system includes more than one processor cluster, the processors of each cluster are coupled to the same communication bus, and each cluster includes a spare processor. *See* Fox, FIG. 1. Fox also discloses the processors of a cluster, including the spare processor, are interconnected with each other via processor-processor buses. *Id.*, [0025], [0033], [0038], FIG. 2B. Fox further discloses switching the

memory state of the failing processor to the spare processor. *Id.*, [0044]; Brogioli, ¶211.

Fox's system would have determined the processors that are coupled to the same communication bus with the boot processor ("determining at least one processor that is coupled to a common communication bus with ... [the] boot processor") because to replace the failing boot processor, the spare processor would need to be in the same cluster with and communicate with the failing boot processor. Fox's system would have further selected at least one of those determined processors as the spare processor because Fox teaches either the administrator or the BIOS processor initialization/identification code would identify the spare processor to the OS. Fox, [0034]; Brogioli, ¶¶212-213.

2. [12b]: "selecting said another of said plurality of processors to be assigned the role of spare processor from said determined at least one processor."

For the reasons discussed in [12a], Fox or Fox-Landry renders obvious [12b]. Brogioli, ¶¶212-214.

J. Dependent Claim 13: "during system boot-up, system firmware identifying the spare processor and holding said spare processor in reserve as a hot spare."

Fox discloses that during system initialization, the BIOS recognizes the spare processor, identifies the spare processor with a software tag ("during system boot-up, system firmware identifying the spare processor"), and reserves the processor as

a hot-spare processor that remains in an idle, off, or low-power mode not utilized by the operating system (“holding said spare processor in reserve as a hot spare”). Fox, Abstract, [0015], [0034], [0040], FIG. 1; Brogioli, ¶¶215-216.

K. Dependent Claim 14: “during system boot-up, system firmware booting the spare processor as an active spare.”

Fox discloses during system BIOS boot (“during system boot-up”), the BIOS (“system firmware”) reserving the spare processor in an idle state and identifies the spare processor by setting a software tag. Fox, [0015], [0034]. As discussed in claim 6, a POSITA would have configured Fox’s spare processor as “an active spare” to improve system processing capacity. A POSITA would have understood that, instead of reserving the spare in an idle state during initialization, the BIOS would have initialized and identified the active spare processor to the operating system (“boot[.]”) like other active processors in the system. *Id.*, [0034]; Brogioli, ¶¶217-218.

L. Dependent Claim 15: “system firmware performs said changing operation.”

Fox-Safford-Arai or Fox-Safford-Landry-Arai renders obvious this limitation for the reasons explained in Grounds 1-2, [1c], [9c], and the additional reasons below. Brogioli, ¶¶219-220.

A POSITA would have understood that Fox’s BIOS (“system firmware”) would have performed the changing operation because: (1) the BIOS would assign

processor configurations; and (2) both Fox and Arai disclose that the BIOS performs the processor switching without interrupting the operating system. Fox, [0035], [0038], [0041]-[0043]; Arai, [0004]-[0006], [0023], [0025]; Brogioli, ¶220.

M. Dependent Claim 17: “system firmware operable, responsive to detecting loss of lockstep for said boot processor, changing the assignment of the role of boot processor to said spare processor.”

Fox-Safford-Arai or Fox-Safford-Landry-Arai renders obvious this limitation for the reasons explained in Grounds 1-2, [1c], [9c], and claim 15. Brogioli, ¶221.

N. Dependent Claim 18: “wherein said system firmware performs said changing during system runtime without requiring an operating system of said system to be shutdown.”

Fox-Safford-Arai or Fox-Safford-Landry-Arai renders obvious this limitation for the reasons explained in Grounds 1-2, [1c], [9c]-[9d], and claim 17. *Id.*, ¶222.

O. Dependent Claim 22: “system firmware that, during system boot-up, identifies the spare processor and holds said spare processor in reserve as a hot spare.”

Fox discloses or renders obvious this limitation for the reasons explained in claim 13. *Id.*, ¶223.

P. Dependent Claim 23: “system firmware that, during system boot-up, boots the spare processor as an active spare.”

Fox discloses or renders obvious this limitation for the reasons explained in claim 14. *Id.*, ¶224.

Q. Dependent Claim 26: “wherein said system causing said spare processor to dynamically assume the role of boot processor comprises: said system causing said spare processor to dynamically assume the role of boot processor without shutting down an operating system of said system.”

Fox-Safford-Arai or Fox-Safford-Landry-Arai renders obvious this limitation for the reasons explained in [25f] and [9d]. *Id.*, ¶225.

R. Dependent Claim 29: “storing information identifying said selected first one of said plurality of lockstep processor modules as said boot processor and identifying said selected second one of said plurality of lockstep processor modules as said spare processor.”

Fox-Safford-Arai or Fox-Safford-Landry-Arai renders obvious this limitation for the reasons explained in Grounds 1-2, [1b], [16d], and claims 10-11. *Id.*, ¶226.

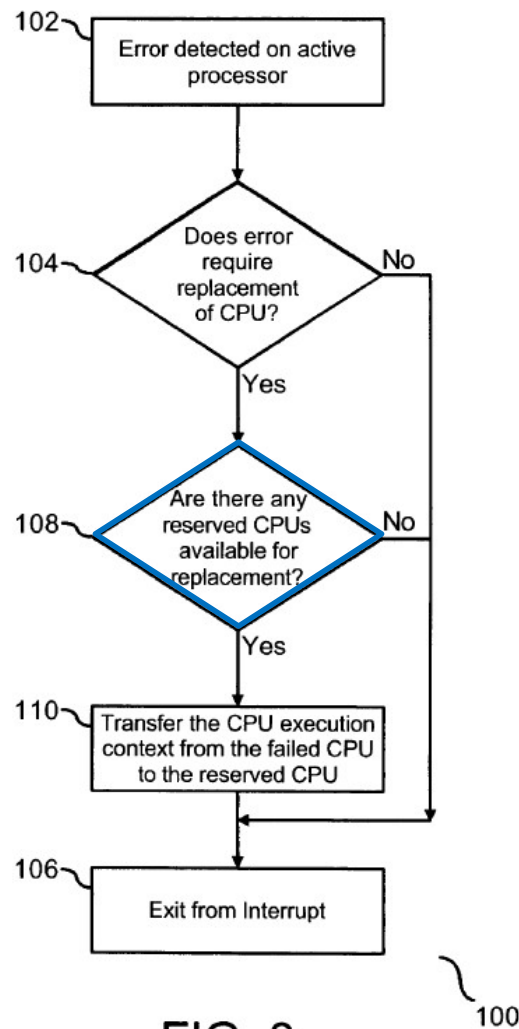
S. Dependent Claim 30

1. [30a]: “system firmware determining said spare processor from said stored information; and”

As discussed in Grounds 1-2, [1b] and claim 29, Fox-Arai or Fox-Landry-Arai renders obvious storing information that identifies the boot processor and the spare processor, including the identifications and software flags of the boot processor and the spare processor, in Fox’s register or table accessible to the BIOS and operating system. Fox in view of Arai further renders obvious determining the spare processor from the stored information. *Id.*, ¶¶228-232.

Arai discloses that in response to detecting an error on an active processor, the BIOS interrupt handler performs a [test 108](#) to determine if there are any

processors reserved during system initialization available to replace the active processor (“determining said spare processor”). Arai, [0022], FIG. 3. Arai further discloses generating a table (e.g., ACPI table, MP table, or an alternative table) that identifies each processor and the state of each processor. *Id.*, [0019]-[0021], [0025]-[0026]. A POSITA would have understood that Arai’s determination would include having the BIOS interrupt handler (“system firmware”) examine Arai’s table (“stored information”) and determine the processors that are in an inactive, reserved, or idle state—the spare processor. *Id.*, [0019]-[0021]; Brogioli, ¶229.



Arai, FIG. 3.

A POSITA would have been motivated to include Arai's determination step in response to detecting a processor failure to ensure a spare processor is available or selected before switching the failing processor to the spare processor. Fox, [0015], [0024]. A POSITA would have expected success in adding Arai's determination in Fox's processor-switching process using ordinary and routine computer

programming skills. *Id.*, [0026], [0033]-[0035], [0038], [0041], [0046]; Arai, [0022]; IA-32, 273-274, 267, 269, 289-297, 745; Brogioli, ¶¶230-231.

The '781 Patent similarly discloses the system firmware accesses the device tree to determine the spare processor. Ex. 1001, 17:41-46, FIG. 5; Brogioli, ¶232.

2. [30b]: “said system firmware causing an operating system to recognize said spare processor as said boot processor.”

Fox-Arai or Fox-Landry-Arai renders obvious this limitation for reasons explained in [9d] and claims 7 and 15. Brogioli, ¶233.

X. Fox in View of Safford, Arai, and Jin (Ground 3) or Fox in View of Safford, Landry, Arai, and Jin (Ground 4) Renders Obvious Claims 3-4, 19-21, and 27-28

A. Overview of the Combinations

As discussed above, Fox-Safford-Arai (Ground 1) or Fox-Safford-Landry-Arai (Ground 2) renders obvious a multiprocessor system having a plurality of processors (*see* Grounds 1-2, [1a]) and implementing those processors as lockstep processors (*see* Grounds 1-2, [1c], [9c], [16a], and [25a]). And as discussed in Grounds 1-2, [1b], [25a], and [25d], Fox-Safford-Arai or Fox-Safford-Landry-Arai renders obvious selecting one of the lockstep processors as a boot processor and selecting another of the lockstep processors as a spare processor. A POSITA would have found it obvious to modify Fox-Safford-Landry's or Fox-Safford-Landry-

Arai's system to include a user interface for receiving input specifying these selections in view of Jin. Brogioli, ¶¶235-239.

B. A POSITA Would Have Found It Obvious to Include a User Interface for Receiving Input Specifying the Selection of the Boot Processor and Spare Processor to Improve System Customizability and Availability

Fox discloses the initial set-up of the spare processor in its multiprocessor computer system may require administrative input. Fox, [0034]. Fox also discloses its multiprocessor computer system includes I/O controllers and buses for connecting with standard computer peripherals, such as a keyboard. *Id.*, [0027]. Fox thus suggests that the system would include standard and ordinary computer input devices for receiving input information for setting up the processors. Brogioli, ¶236.

Jin discloses a multiprocessor computer system that selects the boot processor and includes input devices, such as a touchscreen, keyboard, keypad, and mouse, for communicating direction information and command selections to the processors. Jin, [0014]-[0015], [0019], [0025], [0049], [0061], FIG. 7; Brogioli, ¶¶77-78, 237.

A POSITA would have been motivated to modify Fox-Safford-Arai's or Fox-Landry-Safford-Arai's system to use the input devices in Jin to receive input information to allow the administrator to customize the selection of the boot and spare processors. An administrator would have selected the boot processor and spare processor sharing common resources to facilitate the reassignment of the

responsibilities of the failing boot processor to the spare processor, minimize interruption to the system, and improve system availability. Brogioli, ¶238.

A POSITA would have expected success in the modification because Fox provides the necessary hardware, e.g., I/O controller and buses, for connecting to standard input devices. Such modification merely involves using a known technique to improve similar devices in the same way taught. *Id.*, ¶239.

C. Dependent Claims 3-4, 19-20, and 27-28

Claim 3 recites “a user interface operable to receive input specifying said one of said plurality of processors to be assigned said role of boot processor.” Claim 19 is substantially similar to claim 3. Claim 27 is substantially similar to claim 3 except omitting “a user interface.” *Id.*, ¶240.

Claim 4 recites “a user interface operable to receive input specifying said another of said plurality of processors to be assigned said role of spare processor.” Claim 20 is substantially similar to claim 4. Claim 28 is substantially similar to claim 4 except omitting “a user interface.” *Id.*, ¶241.

Jin’s multiprocessor computer system selects the boot processor and includes input devices (“user interface”), such as a touchscreen, for communicating direction information and command selections to the processors (“to receive input”). Jin, [0014]-[0015], [0019], [0025], [0049], [0061], FIG. 7. As discussed in Section X.B,

a POSITA would have found it obvious to include a user interface for receiving input specifying the selections of the boot processor and the spare processor to improve customizability and availability of Fox's system. Brogioli, ¶243.

D. Dependent Claim 21: “wherein said user interface presents for selection as said spare processor at least one processor that is communicatively to the same one of said first and second communication buses as said boot processor.”

For reasons explained in [16e] and [25e], a POSITA would have selected the spare processor as one of the processors communicatively coupled to the same communication bus as the boot processor to facilitate processor switching and minimize interruption to the system. Jin discloses the input device can be a touchscreen that displays information to the user. Jin, [0061]. A POSITA would have been motivated to modify Fox's system to include a touchscreen (“user interface”) that presents the processors communicatively coupled to the same communication bus as the boot processor and receives input selecting the spare processor among the presented processors. A POSITA would have understood such modification would reduce the time and effort for an administrator or a user to determine the processors suitable to be selected as the spare processor. Brogioli, ¶245.

A POSITA would have expected success in making the modification because using a user interface of a computer system to present options for selection was

routine and ordinary, and Fox is readily equipped for such modification. Fox, [0026], [0033], [0035], [0038], [0041], [0045]-[0046], Claims 2, 13; Brogioli, ¶246.

XI. Fox in View of Safford, Arai, and Bigbee (Ground 5) or Fox in View of Safford, Landry, Arai, and Bigbee (Ground 6) Renders Obvious Claims 26 and 27

A. Overview of the Combinations

As discussed above, Fox-Safford-Arai (Ground 1) or Fox-Safford-Landry-Arai (Ground 2) renders obvious a multiprocessor system having a plurality of processors (*see* Grounds 1-2, [1a]) and implementing those processors as lockstep processors (*see* Grounds 1-2, [1c], [9c], and [16a]). A POSITA would have found it obvious to upgrade Fox-Safford-Arai's or Fox-Safford-Landry-Arai's multiprocessor system with Itanium Processor Family ("IPF") processors, as taught in Bigbee. Bigbee, [0019], [0032]-[0033], FIG. 3; Brogioli, ¶¶247-251.

B. A POSITA Would Have Found It Obvious to Configure Fox-Safford-Arai's or Fox-Safford-Landry-Arai's System with Itanium Family Processors as Taught in Bigbee to Improve System Performance and Capacity

Fox's multiprocessor server system is configured with IA-32 architecture and comprises IA-32 processors and BIOS of the IA-32 architecture. Fox, Abstract, [0014]-[0015], [0024], [0028], [0033]. Bigbee's multiprocessor system comprises Itanium family processors and architecture. Bigbee, [0033]; Ex. 1030, 267-268. A POSITA would have known that Intel's IA-32 architecture is a 32-bit instruction set

architecture and Itanium family processors' architecture is a 64-bit instruction set architecture. Ex. 1008; Ex. 1030, 19. A POSITA would have been motivated to upgrade Fox-Safford-Arai's or Fox-Safford-Landry-Arai's system's IA-32 processors and architectures to more advanced Itanium family processors and architectures to improve performance and processing capacity. Ex. 1030; Brogioli, ¶¶249-250.

A POSITA would have expected success in upgrading Fox's system with Itanium family processors and architecture using ordinary computer programming skills and common knowledge of Itanium family processors and architecture. *See* Ex. 1030, 267-276; Bigbee, [0033]; Nguyen, [0045]; Brogioli, ¶251.

C. Dependent Claims 8 and 24

Claim 8 recites "wherein said plurality of processors are Itanium Processor Family (IPF) processors." Claim 24 is substantially similar to claim 8.

Bigbee discloses using Itanium family processors in a multiprocessor system, and further discloses enabling pairs of these processors to operate in FRC mode or lockstep mode. Bigbee, [0019], [0033]-[0035]; Brogioli, ¶¶73-76. For reasons discussed in Section XI.B, a POSITA would have been motivated to use Itanium family processors and architecture to improve system performance and capacity, and expected success in such modification. Brogioli, ¶254.

XII. *Fintiv* Discretionary Denial Is Inappropriate

At this time, no factors articulated in *Apple Inc. v. Fintiv, Inc.*, IPR2020-00019, Paper 11, 5-6 (PTAB Mar. 20, 2020), weigh in favor of non-institution under 35 U.S.C. § 314(a) because the parallel litigations are administratively stayed (Exs. 1035, 1036).

XIII. Discretionary Denial Is Inappropriate Under 35 U.S.C. § 325(d)

A. The Grounds of This Petition Were Not Considered During Prosecution

This Petition presents new prior art, grounds, and arguments not previously considered by the Office. *Advanced Bionics, LLC v. MED-EL Elektromedizinische Geräte GmbH*, IPR2019-01469, Paper 6 (PTAB Feb. 13, 2020); *Becton, Dickinson & Co. v. B. Braun Melsungen AG*, IPR2017-01586, Paper 8 (PTAB Dec. 15, 2017).

Under the *Advanced Bionics* step 1, Arai, Landry, and Jin were not before the Examiner. None of the grounds presented in this Petition were addressed during prosecution of the '781 Patent.

Safford was cited by the Examiner in a non-final office action that rejected claims 1, 3-4, 6-8, 10, and 13-17 as anticipated and rejected claims 2, 5, 9, 11-12, and 18-32 as obvious over Safford in view of Marshall. Ex. 1004, 58-75. The Examiner subsequently maintained the Safford anticipation rejections. *Id.*, 101-105. Applicant maintained that Safford does not disclose a boot processor. *Id.*, 127-129.

Applicant and Examiner discussed Safford during a phone interview and Applicant subsequently amended the claims. *Id.*, 134-140, 155. The Examiner never applied Safford in a combination again.

Fox was cited by the Examiner in a non-final office action that rejected claims 1-17 as obvious over Fox in view of Quach (U.S. Patent No. 6,625,749). *Id.*, 106-112. The Examiner relied on Fox for “data storage storing [abstract] information that assigns a role of boot processor to one of said plurality of processors and assigns a role of spare processor to another of said plurality of processors,” but did not rely on Fox for disclosing a boot processor. *Id.*, 106. Applicant argued that Quach-Fox did not disclose a boot processor or switching the role of the boot processor, and the Examiner allowed the application. *Id.*, 129, 133-134, 155.

Bigbee was cited on an IDS submitted by Applicant and signed by the Examiner. *Id.*, 80, 225, 301. The Examiner did not analyze Bigbee in an office action (*see* Ex. 1004) or Notice of Allowance. *Id.*, 151-154.

Even under a broad reading of *Becton, Dickinson, Advanced Bionics* at 10, denial would be improper because the art relied on in this Petition is not substantially the same as, or cumulative of, the prior art considered during prosecution. None of the grounds in this Petition were analyzed during prosecution and several references were not analyzed at all.

Under the *Advanced Bionics* step 2, Grounds 1-2 above show that the Examiner made four material errors.

First, the Examiner did not reapply Safford in a combination after Applicant erroneously argued that it was not valid prior art under § 103, despite Applicant walking back that statement. *See* Ex. 1004, 127-129. The Examiner's oversight on Safford's applicability in a combination reflects error and oversight, specifically because the Examiner did not combine Safford with Fox. *See Billerud Aktiebolag (publ) v. WestRock MWV, LLC*, IPR2023-01206, Paper 14, 16-17 (PTAB Mar. 13, 2024) (finding examiner error where examiner did not consider combination of references).

Second, the Examiner erred by overlooking that Fox's spare processor would be used for replacing the boot processor in which a failure is detected. *See* Section VII.B.2. The Examiner's failure to recognize this disclosure reflects error and oversight.

Third, the Examiner failed to apply Fox and Safford to the claims that were amended after the phone interview. *See Billerud*, IPR2023-01206, Paper 14, 16-17 (finding examiner error where prior art was not considered after amendment); *Peloton Interactive, Inc. v. Mad Dogg Athletics, Inc.*, IPR2022-00016, Paper 9, 15 (PTAB Apr. 8, 2022) (finding examiner error for not applying art to amended

claims). As discussed in Grounds 1-2, [1d], the amended limitation is disclosed by Fox and Safford.

Fourth, Grounds 5-6 show the Office made a material error by not using Bigbee in a rejection of any claims. Although the Examiner signed the IDS listing Bigbee, the Examiner did not discuss or rely on Bigbee. The Examiner's silence with respect to Bigbee reflects error and oversight.

B. This Petition Is Not Cumulative of Unified Patents' *Ex Parte* Reexamination

Discretionary denial of this Petition in view of Unified's EPR is improper. Unified has not been accused of infringement, but real party-in-interest BMW AG has. Defendants accused of infringement should be entitled to challenge the claims asserted against them.

Unified filed its Request for *Ex Parte* Reexamination on September 1, 2023, based on the grounds below. Ex. 1032, 233, 803-918.

Ground	Claims	Basis	Reference(s)
1	1-6, 9-15	§ 103	Kalyanasundharam in view of Kennedy
2	1-6, 9-15	§ 103	Safford in view of Kennedy
3	1-30	§ 103	Kennedy in view of Nguyen and Fox
4	1, 6	§ 103	Safford in light of Cepulis

On October 10, 2023, reexamination was ordered for all claims. *Id.*, 199-218.

On February 20, 2024, the Examiner issued a non-final office action rejecting all claims based on the grounds below. *Id.*, 137-194.

Rejection	Claims	Basis	Reference(s)
1	1-6, 9-15	§ 103	Safford in view of Kennedy
2	1-30	§ 103	Kennedy in view of Nguyen and Fox

Patent Owner responded to the non-final office action on May 20, 2023, and on July 26, 2024, the Examiner issued a final office action rejecting a subset of the claims on the grounds listed below. *Id.*, 1-73, 82-102.

Rejection	Claims	Basis	Reference(s)
1	1-2, 5-6, 9-15	§ 103	Safford in view of Kennedy
2	1-2, 5-18, 21-30	§ 103	Kennedy in view of Nguyen and Fox

Unified's reexamination and the Examiner's subsequent office actions present materially different grounds that are not the same or substantially the same as those presented in this Petition because they do not rely on Arai, Landry, Jin, or Bigbee—critical references for this Petition. Additionally, in this Petition, Safford is used as a secondary reference rather than a primary reference like in the reexamination.

Likewise, Fox is used as the primary reference in this Petition, not a secondary reference, and the reexamination does not involve any grounds that combine Fox and Safford.

Accordingly, discretionary denial of this Petition would be improper.

XIV. Mandatory Notices

A. Real Parties-in-Interest

The real parties-in-interest for this Petition are BMW of North America, LLC and Bayerische Motoren Werke AG.

B. Related Matters

To Petitioner's knowledge, the '781 Patent is involved in the following:

Name	Number	Court	Filed
<i>Foras Technologies Ltd. v. Bayerische Motoren Werke AG</i>	1:24-cv-00363	E.D. Va.	Mar. 6, 2024
<i>Foras Technologies Ltd. v. Toyota Motor North America, Inc.</i>	2:23-cv-00321	E.D. Tex.	July 12, 2023
<i>Foras Technologies Limited v. Nissan Motor Company, Ltd. et al.</i>	1:23-cv-00640	W.D. Tex.	June 6, 2023
<i>Foras Technologies Limited v. Toyota Motor North America, Inc. et al.</i>	2:23-cv-00150	E.D. Tex.	Apr. 5, 2023
<i>Ex Parte Reexamination</i> filed by Unified Patents	Control No. 90/019,244	USPTO	Sept. 1, 2023

C. Lead and Back-Up Counsel

Lead Counsel	Back-Up Counsel
Lionel M. Lavenue (Reg. No. 46,859) lionel.lavenue@finnegan.com Finnegan, Henderson, Farabow, Garrett & Dunner, LLP 1875 Explorer Street, Suite 800 Reston, VA 20190-6023 Tel: 571-203-2700 Fax: 202-408-4400	Kara A. Specht (Reg. No. 69,560) kara.specht@finnegan.com Finnegan, Henderson, Farabow, Garrett & Dunner, LLP 271 17th Street, NW, Suite 1400 Atlanta, GA 30363-6209 Tel: 404-653-6400 Fax: 404-653-6444 Cory C. Bell (Reg. No. 75,096) cory.bell@finnegan.com Xirui Zhang (Reg. No. 78,809) xirui.zhang@finnegan.com Finnegan, Henderson, Farabow, Garrett & Dunner, LLP 2 Seaport Lane, 6th Floor Boston, MA 02210-2001 Tel: 617-646-1641 Fax: 202-408-4400 Alissa E. Green (Reg. No. 78,501) alissa.green@finnegan.com Finnegan, Henderson, Farabow, Garrett & Dunner, LLP 901 New York Avenue, NW Washington, DC 20001-4413 Tel: 202-408-4000 Fax: 202-408-4400 Deanna Smiley (Reg. No. 79,284) deanna.smiley@finnegan.com Finnegan, Henderson, Farabow, Garrett & Dunner, LLP 1875 Explorer Street, Suite 800 Reston, VA 20190-6023

Lead Counsel	Back-Up Counsel
	Tel: 571-203-2700 Fax: 202-408-4400

D. Service Information

Please address correspondence to counsel at the addresses above and BMW-Foras-IPR@finnegan.com. Petitioner consents to service by e-mail.

XV. Standing

The '781 Patent is eligible for *inter partes* review and Petitioner is not barred or estopped from requesting review.

XVI. Conclusion

Petitioner requests the Board institute review and cancel the Challenged Claims.

Dated: August 30, 2024

Respectfully submitted,

/Lionel M. Lavenue/

Lionel M. Lavenue, Lead Counsel

Reg. No. 46,859

CERTIFICATION UNDER 37 C.F.R. § 42.24(d)

Pursuant to 37 C.F.R. § 42.24(a)(1)(i), the undersigned hereby certifies that the foregoing Petition for *Inter Partes* Review contains 12,923 words, excluding the parts of this Petition that are exempted under 37 C.F.R. § 42.24(a), as measured by the word-processing system used to prepare this paper.

/Lionel M. Lavenue/

Lionel M. Lavenue, Lead Counsel
Reg. No. 46,859

CERTIFICATE OF SERVICE

Pursuant to 37 C.F.R. §§ 42.6(e) and 42.105(a), the undersigned certifies that on August 30, 2024, a copy of the foregoing **Petition for *Inter Partes* Review, Petitioner's Power of Attorneys and Designation of Lead and Back-up Counsel, and Exhibits 1001-1036** were served by FedEx Priority Overnight on the correspondence address of record indicated in the Patent Office's Patent Center system for U.S. Patent No. 7,627,781.

Antonio Papageorgiou
Lombard Geliebter LLP
1325 Avenue of the Americas
28th Floor
New York, NY 10019

Date: August 30, 2024

By: /Daniel E. Doku/
Daniel E. Doku
Litigation Legal Assistant
FINNEGAN, HENDERSON, FARABOW,
GARRETT & DUNNER, LLP