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 IPR2024-01347
U.S. Patent 7,624,302

PETITION FOR *INTER PARTES* REVIEW

TABLE OF CONTENTS

TABLE OF AUTHORITIES

Page(s)

Cases

Statutes

Regulations

TABLE OF EXHIBITS

Exhibit	Description
Ex. 1001	U.S. Patent No. 7,624,302 B2 to Michaelis et al. ("the '302 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 '302 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

Exhibit	Description
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

Exhibit	Description
Ex. 1026	L. Spainhower and T.A. Gregg, *IBM S/390 Parallel Enterprise Server G5 fault tolerance: A historical perspective*, 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., *RAS strategy for IBM S/390 G5 and G6*, 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 *Ex Parte* Reexamination of the '302 Patent, Control No. 90/019,243
Ex. 1032	Prosecution History of *Ex Parte* 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. 7,627,781 B2 to Michaelis et al. ("the '781 Patent"), filed October 25, 2004, granted December 1, 2009
Ex. 1035	*Bayerische Motoren Werke AG v. Foras Techs. Ltd.*, Case No. 1:24-cv-363 (LMB/WEF), Dkt. 54 (E.D. Va. July 12, 2024) ("EDVA Stay Order")
Ex. 1036	*Foras Techs. Ltd. v. Nissan Motor Co., Ltd.*, 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 of claims 1-27 ("the Challenged Claims") of U.S. Patent No. 7,624,302 ("the '302 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, granted April 18, 1995	§ 102(b)
Ex. 1012	US 2003/0126498 A1 ("Bigbee"), filed January 2, 2002, published July 3, 2003	§ 102(b)

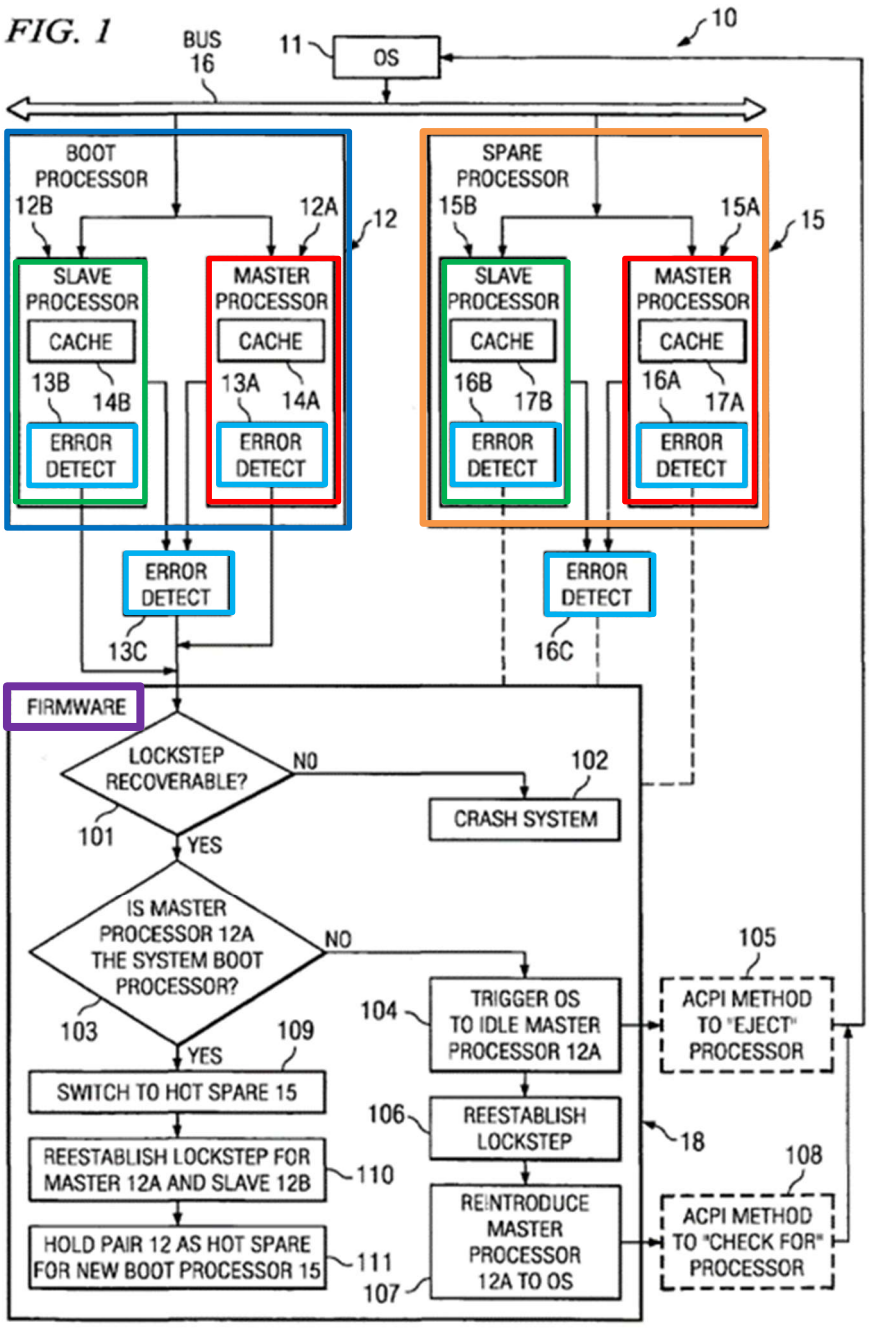
Claims 1-27 are unpatentable under the following grounds:

Ground	Claims	Reference(s)	Basis
1	1-25	Fox in view of Safford and Arai	§ 103
2	1-25	Fox in view of Safford, Landry, and Arai	§ 103
3	26, 27	Fox in view of Safford, Arai, and Bigbee	§ 103

4	26, 27	Fox in view of Safford, Landry, Arai, and Bigbee	§ 103

II. Overview of the '302 Patent

The '302 Patent claims priority to October 25, 2004. Ex. 1001, cover. The '302 Patent discloses a multi-processor system and method that switches a lockstep boot processor that lost lockstep to a spare processor. *Id.*, Abstract, 3:64-4:13, FIGS. 1-6. The multiprocessor system includes a boot processor, a spare processor, and processors other than the boot processor and spare processor. *Id.*, 4:14-27, 7:44-50, 12:22-35, 13:21-35. 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.*, 4:14-19, 6:42-58, 7:6-13, FIG. 1 (reproduced below). The multiprocessor system includes firmware that, responsive to LOL, switches the designation of the boot processor to the spare processor. *Id.*, 6:10-24, 9:32-45, 9:65-11:3, FIGS. 2-6.



Id., FIG. 1;[1] Brogioli, ¶¶56-58.

[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 '302 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.*, ¶¶40-55.

1. On or before October 25, 2004, using lockstep processors in a multiprocessor system was known. *Id.*, ¶¶40-44.

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.*, ¶43.

3. On or before October 25, 2004, it was known that a multiprocessor system includes a boot processor that performs system initialization. *Id.*, ¶¶45-48.

4. On or before October 25, 2004, it was known that a multiprocessor system configured with IA-32 architecture includes a boot processor. *Id.*, ¶¶46-47.

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.*, ¶48.

6. On or before October 25, 2004, it was known to use a spare processor to recover a failing processor in a multiprocessor. *Id.*, ¶¶49-52.

7. On or before October 25, 2004, using both lockstep processors and spare processors in a multiprocessor system was known. *Id.*, ¶52.

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.*, ¶¶53-55.

9. On or before October 25, 2004, it was known that the BIOS of a computer system is firmware typically embedded in a non-volatile memory. *Id.*, ¶53.

V. Claim Construction

The Board construes claims per *Phillips v. AWH Corp.*, 415 F.3d 1303 (Fed. Cir. 2005) (en banc), according to their ordinary and customary meaning as understood by a POSITA in the context of the specification. *Id.* at 1312-13. 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, ¶¶31-39.

A. "detecting loss of lockstep (LOL)" and "detect loss of lockstep (LOL)"

Terms "detecting loss of lockstep (LOL)" and "detect loss of lockstep (LOL)" 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. Ex. 1001, 2:40-3:6, 7:25-36; Brogioli, ¶¶34-36.

This is consistent with the undisputed construction 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. Means-Plus-Function Terms

During prosecution, the Examiner concluded, without dispute, that several limitations are means-plus-function limitations. Ex. 1031, 920-9952. To the extent 35 U.S.C. § 112, ¶6, applies to these limitations, Petitioner proposes the following constructions. Brogioli, ¶¶37-38.

Claim	Term	Function	Structure	Support
[26a]	"**system firmware means** for detecting loss of lockstep (LOL) for a processor in a multi-processor system"	detecting loss of lockstep (LOL) for a processor in a multi-processor system	firmware including a Processor Abstraction Layer (PAL), System Abstraction Layer (SAL), and Extended Firmware Interface (EFI), or equivalents thereof	Ex. 1001, 8:34-59, 11:4-60, 12:36-59, FIGS. 1-2
[26b]	"**system firmware means** for determining whether the processor for which said LOL is detected is assigned the role of boot processor"	determining whether the processor for which said LOL is detected is assigned the role of boot processor	firmware including a Processor Abstraction Layer (PAL), System Abstraction Layer (SAL), Extended Firmware Interface (EFI), and a device tree, or equivalents thereof	*Id.*, 5:34-48, 9:49-60, 11:4-60, 12:36-14:3, 18:50-58, FIGS. 1-2
[26c]	"**system firmware means** for switching the role of boot processor to a spare processor without shutting down the system's operating	switching the role of boot processor to a spare processor without shutting down	firmware including a Processor Abstraction Layer (PAL), System Abstraction	*Id.*, 11:4-60, 12:36-14:3, 17:12-18:43, FIGS. 1-2

	system if determined that the processor for which said LOL is detected is assigned the role of boot processor"	the system's operating system if determined that the processor for which said LOL is detected is assigned the role of boot processor	Layer (SAL), and Extended Firmware Interface (EFI), or equivalents thereof	

VI. The Cited Art Is Analogous Art to the '302 Patent

Like the '302 Patent, all cited art relates to fault-tolerant systems that operate despite a processor error. Like the '302 Patent, **Fox's** system detects a processor failure, in response to detecting the processor failure, and replaces the failing processor with a spare processor. Fox, Abstract, [0006], [0014], [0016], [0019], [0023]-[0024], [0036], [0044], [0047], FIG. 3. **Safford's** 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** 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** discloses a fault-tolerant power-up logic in finding a functional CPU to

operate as the logical CPU0 or boot processor. Landry, Abstract, 1:6-10, 2:9-41, 6:44-7:52, Figs. 1, 4. **Bigbee** discloses a system including 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. All art is in the same field of endeavor as the '302 Patent. Brogioli, ¶¶56-78.

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. *Id.*, ¶¶59-63, 82.

Fox's multiprocessor system does not expressly include lockstep processors or detecting loss of lockstep, but Safford discloses 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. *Id.*, ¶¶64-66, 83. A POSITA would have found it obvious to implement Fox's processors as lockstep processor pairs with error detection logics, taught in Safford, to improve reliability and availability. Fox, [0016], [0044], [0047] (providing high availability); 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, ¶83.

A POSITA would have understood that Fox's multiprocessor system would have a boot processor to initialize the system and load the operating system. Brogioli, ¶¶45-48, 84. 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 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.*, ¶84.

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. *Id.*, ¶¶67-69. A POSITA would have understood that Arai provides implementation details for switching the failing processor and the spare processors, as suggested by Fox. *Id.*, ¶85.

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.*, ¶¶86-108.

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.*, ¶¶40-44. And it was also known to use both lockstep processors and spare processors in a fault-tolerant multiprocessor system. *Id.*, ¶¶49-52. 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 are 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 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, ¶87.

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. Bigbee, [0002] (two or more processor cores incorporated into a single processor package); Nguyen, [0020] (an FRC-enabled processor including two execution cores operating in FRC model or lockstep mode). 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. Fox, [0028], [0034], [0040]; Safford, [0015], [0018], [0020]-[0021]; Ex. 1027, 5-6; Arai, [0019]. A POSITA would have known that processors of a multiprocessor system configured with IA-32 architecture, like Fox's, would operate in either independent mode or in lockstep mode with lockstep error detection. IA-32, 470-471; Suffin, 6:13-22; Elkhoury, 6:36-47; Brogioli, ¶¶88-89.

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, ¶¶40-44, 90.

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 the systems management interrupt ("SMI") in response to the detection of the loss of lockstep, 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, ¶¶91-92.

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 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. IA-32, 273-274, 294, 307; Brogioli, ¶93.

A POSITA would have understood that the 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. Brogioli, ¶¶45-48. 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* IA-32, 267, 269, 289-295; Brogioli, ¶94.

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 (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, ¶95.

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, loss of lockstep 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, ¶96.

2. Fox's multiprocessor system would have designated a processor as the 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; Brogioli, ¶¶45-48, 59-63. Fox expressly discloses system BIOS boot, which would be performed by a boot processor. Fox, [0006], [0034], Claims 4-6. 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, ¶97.

Fox suggests that its spare processor would be used for replacing the boot processor when failure in the boot processor is detected to improve system reliability. *Id.*, ¶¶98-102.

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. As Dr. Brogioli explains, once the operating system is up and running, the boot processor functions as an application processor. *Id.*, ¶47; MP Specification, 75. A POSITA would have understood that both the boot processor and application processors would be active processors in Fox, and Fox's spare processor would replace any failing processor. Fox, [0032]-[0033], [0035] (the system has a register or table for tracking active processors), [0045] (during initialization, the spare processor is inactive); Brogioli, ¶99.

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 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, ¶100.

Third, Fox's BIOS would have determined whether the failing processor is the boot 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 determine the failing processor is the boot processor to promptly replace it with the spare processor. A POSITA would have understood that such prompt replacement would improve the reliability 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, ¶101.

A POSITA would have expected success in determining whether the failing processor in Fox is the boot processor. Fox's SMI's identification of the failing processor to the BIOS and the operating system would have determined the failing processor is the assigned boot processor based on the unique identification of the boot processor (e.g., APIC ID or logical APIC ID). A POSITA would have understood that the BSP flag would be readily examined by the BIOS and operating system to determine whether the failing processor is the boot processor. IA-32, 238, 276. 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 reliability. Brogioli, ¶102.

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, ¶103.

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. Brogioli, ¶¶49-52; Slegel, 3:22-40 (spare processor assuming the logical identifier of a checkstopped processor is critical to make the processor sparing operation appear transparent to the operating system), 6:16-43 (spare processor to assume the role of the checkstopped processor loads the micro-architected state of the checkstopped

processor). Incorporating Arai's teaching into Fox-Safford simply involves the addition of a well-known and suitable implementation detail in Fox's processor-switching process. Brogioli, ¶104.

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 that during the SMI, the system state of each processor is completely saved to memory, and the BIOS switches the memory save state from the old processor (i.e., the failing processor) to the new processor (i.e., 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 supports 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 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, ¶105.

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, which is expressly taught as desirable by both Fox and Arai and known in the art. Fox, [0008], [0013], [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); Brogioli, ¶¶49-52. A POSITA would have understood that transparent switching would improve availability of Fox-Safford's multiprocessor system because the processes of the operating system would continue despite processor failure (e.g., loss of lockstep). *Id.*, ¶106.

A POSITA would have understood how to implement Arai's processor-switching details in Fox-Safford's system and expect success. Both Fox and Arai disclose or suggest (*id.*, ¶107):

- replacing a failing processor with a spare processor in a multiprocessor system (Fox, Abstract, [0014]-[0015]; Arai, Abstract, [0002], [0008]-[0009], [0019]);
- using registers or tables to store processor information available to the OS and BIOS and use the stored information to implement the switching (Fox, [0033], [0035], [0038], [0041], [0045]-[0046], Claims 2, 13; Arai, [0019]-[0022], [0025]-[0026], Claims 3, 9-11, 18-19, FIG. 1);
- using a BIOS interrupt handler for responding to processor error and implementing the processor switching or replacement (Fox, [0024], [0038]-[0039], [0041]-[0042]; Arai, [0022]-[0024]);
- using APIC identification for identifying processors (Fox, [0026]; IA-32, 244, 273-274; Arai, FIGS. 2, 4a-4b, 5a-5b (showing logical APIC ID));
- saving the state or context of the failing processor and transferring it to the spare processor (Fox, [0041]; Arai, [0022], [0025]); and
- using firmware to designate the processor roles during system initialization, identify the failing processor, and replace the failing

processor with the spare processor (Fox, [0040]-[0042]; IA-32, 244; MP Specification, 75, 79; Arai, [0020], [0022]-[0024], [0026]).

Fox 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 processor-switching details in Fox-Safford 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, ¶¶49-52, 108.

C. Independent Claim 1

1. [1preamble]: "A method comprising:"

Fox discloses "a method … for dynamically replacing a failing processor" with a spare processor. Fox, Abstract, [0002], [0014], [0023], [0046]; Brogioli, ¶110.

2. [1a]: "detecting loss of lockstep (LOL) for a processor in a multi-processor system;"

Fox discloses a multiprocessor computer system configured with IA-32 architecture, the system including a spare processor and a plurality of main processors. Fox, [0019], [0023]-[0025], FIG. 1. Fox does not expressly disclose that its processors are lockstep processors or detecting loss of lockstep (LOL). Safford does. Brogioli, ¶¶111-112.

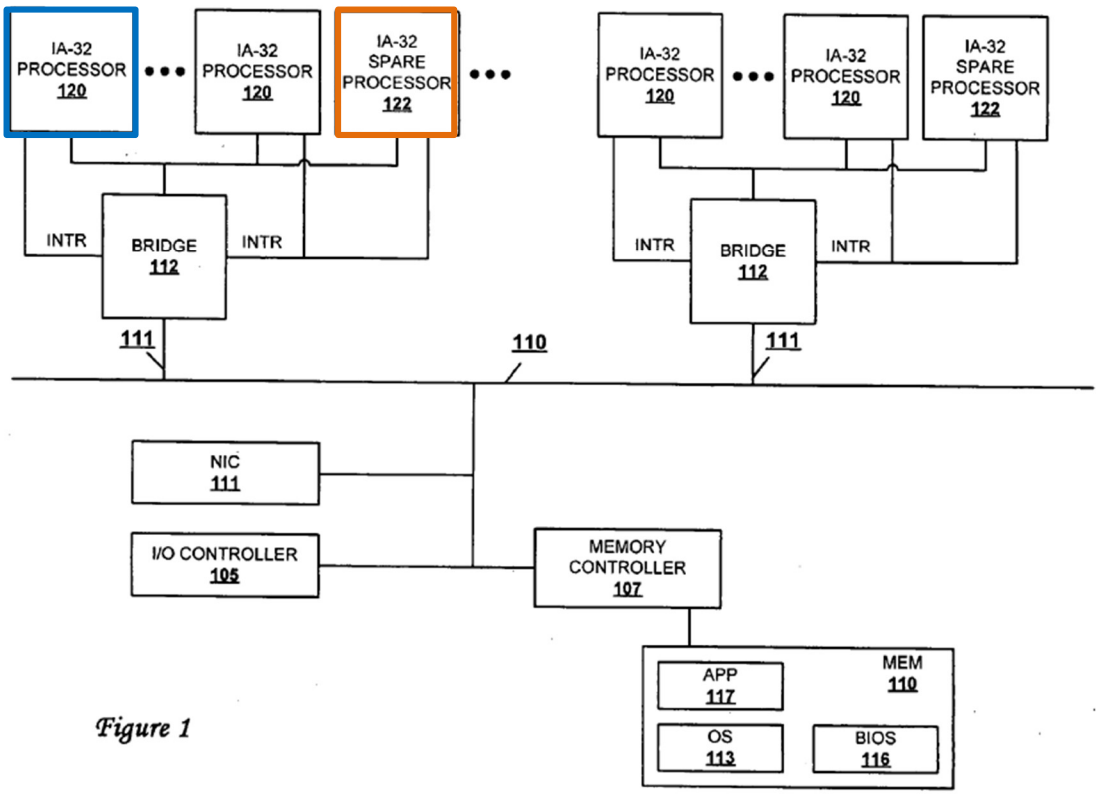


Figure 1

Fox, FIG. 1.

Safford discloses a multiprocessor computer system 100, and 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 include a logic circuit 115 to detect differences in the outputs of the pair of processors 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, ¶113.

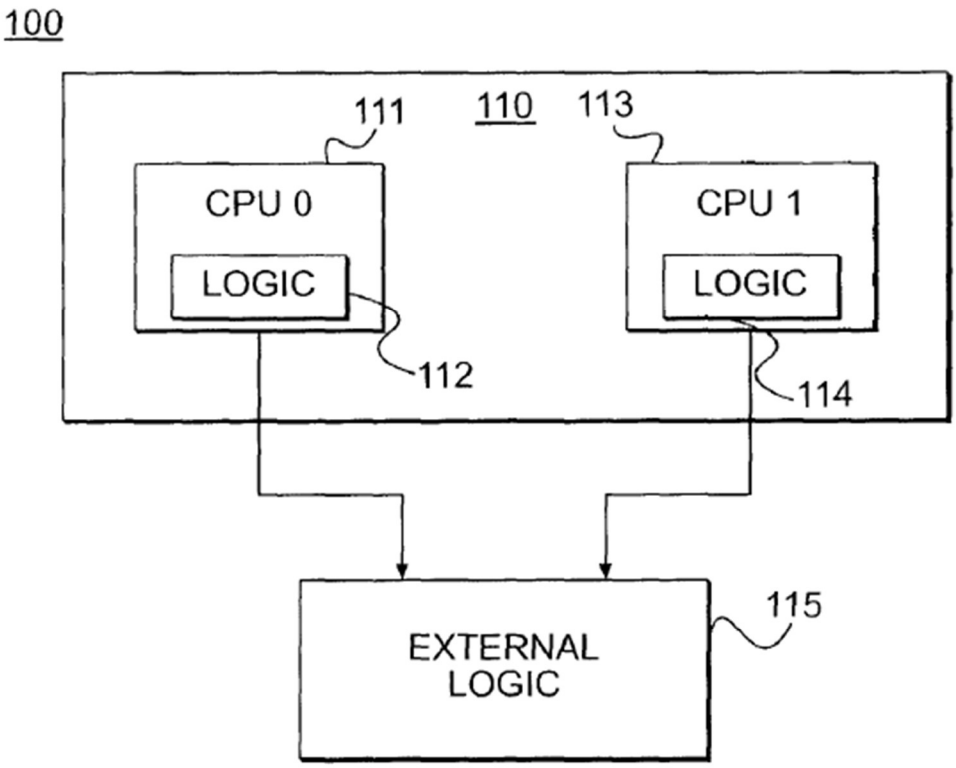


FIG. 2

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 (taught by Safford) to improve reliability and expected success. Safford, [0003]. A POSITA would have understood that when Fox's processors are implemented as lockstep processors with error detection logics (taught by Safford), detecting a

processor failure would have included "detecting loss of lockstep (LOL)." *See* Section VII.B.1; Brogioli, ¶114.

3. [1b]: "determining that the processor for which said LOL is detected is assigned a role of boot processor;"

As discussed in Section VII.B.2, a POSITA also would have understood that Fox's system would have designated a boot processor, and Fox's BIOS would have determined that the failing processor is the designated boot processor ("assigned a role of boot processor"). Fox, [0042]. This would have ensured continued performance of boot processor responsibilities during system runtime to improve system reliability. Brogioli, ¶¶115-116.

As explained for [1a] and Section VII.B.1, in Fox, as combined with Safford's lockstep processors and error detection logics, detecting a processor failure would have included "detecting loss of lockstep (LOL)." In Fox-Safford, Fox's determining that the failing processor is assigned a role of boot processor would be "determining that the processor for which said LOL is detected is assigned a role of boot processor." *Id.*, ¶117.

4. [1c]: "changing a system identifier, used by the system's operating system to recognize the boot processor, of said boot processor for which said LOL is detected, to a different value, which causes the system's operating system to not recognize the processor for which said LOL is detected as the boot processor; and"

As explained for [1a]-[1b], Fox-Safford renders obvious detecting a loss of lockstep and determining that the processor for which said LOL is detected is the boot processor. A POSITA would have understood that in Fox-Safford, Fox's spare processor would be a lockstep pair and used for replacing the boot processor for which said LOL is detected. *See* Sections VII.B.1-VII.B.2; Brogioli, ¶¶118-119.

Fox discloses saving processor identifications in 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. This was well known for IA-32 multiprocessor systems. IA-32, 273-274, 267, 269, 289-297, 709-727. It was known that all processors in an IA-32 multiprocessor system, like Fox, would have unique identifications assigned during the system initialization; the identifications are stored in the ACPI table and MP table created by the boot processor and accessible to the BIOS and operating system. *Id.*, 244, 256 (e.g., local APIC ID or logical APIC ID); MP Specification, 38, 69, 72-73, 76. A POSITA would have understood that Fox's processor identifications would be used

to implement the switching of the failing processor to the spare processor. Brogioli, ¶¶49-52. Arai discloses such implementation detail. *Id.*, ¶120.

Arai discloses replacing an active processor with an error with a reserved inactive processor. 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]; Brogioli, ¶121.

Arai discloses switching the logical APIC IDs of the active processor in error and a reserved processor. Arai, [0023]-[0024], FIGS. 4a-4b. Arai's logical APIC ID would be used by its operating system to recognize the processor in error, and constitutes "a system identifier." *Id.*, [0020]-[0022], [0025]-[0026]. A POSITA would have understood that such switching would have changed the logical APIC ID ("system identifier") of the processor in error to the logical APIC ID of the reserved processor ("a different value"). *Id.*; Brogioli, ¶121.

In Fox-Safford-Arai, a POSITA would have understood that identification switching would have switched the logical APIC ID ("system identifier") of the

lockstep processor pair of the boot processor detected to have LOL with that of the spare processor pair. Safford, [0021] (the spare processor becomes the new logical CPU0). A POSITA would have understood that such switching would have "cause[d] the system's operating system to not recognize the processor for which said LOL is detected as the boot processor" because the unique logical identification of the boot processor having LOL has been assigned to that of the spare processor. The operating system would have instead recognized the spare processor now assigned the unique logical identification of the boot processor as the boot processor. Brogioli, ¶122.

As discussed in Section VII.B.3, a POSITA would have been motivated to implement Arai's switching processor identifications ("changing a system identifier … to a different value") in Fox-Safford and expected success because it is the addition of a well-known and suitable implementation detail. *Id.*, ¶123.

5. [1d]: "changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to a spare processor without shutting down the system's operating system."

Arai discloses switching the identification numbers of the active processor in error and the reserved inactive processor. Arai, [0022]-[0023], [0025]. In Fox-Safford-Arai, a POSITA would have understood that such identification switching would have switched the logical identification of the boot processor lockstep pair

with that of the spare processor pair. A POSITA would have understood that such switching would have changed the logical identification, e.g., logical APIC ID, of the spare processor to that of the boot processor ("changing a system identifier of the spare processor to that of the boot processor"). Brogioli, ¶¶124-125.

Fox's BIOS switches the memory save state from the failing processor to the spare processor. Fox, [0041]. Similarly, Arai 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." Arai, [0022]. As discussed in Section VII.B.3, a POSITA would have been motivated to incorporate Arai's transferring the execution context in Fox's switching process to allow the processes to be continued by the spare processor without interrupting the operating system. Fox, [0008], [0013], [0036]; Arai, [0006]; Slegel, 3:6-20; Brogioli, ¶¶126, 49-52.

A POSITA would have understood that in Fox-Safford, in response to determining that the LOL is detected in the boot processor, implementing Arai's identification switching and context transfer would have reassigned the processor

logical identification and responsibilities of the failing boot processor pair to the spare processor pair without interrupting the operating system ("switching the role of boot processor to a spare processor without shutting down the system's operating system"). Fox, [0008], [0010], [0013]; Arai, [0006]. As discussed in Section VII.B.3, a POSITA would have understood that incorporating Arai's processor-switching implementation detail would improve the availability and reliability of Fox-Safford. A POSITA would have expected success because it involves using known techniques to improve a similar device in the same way as the prior art. Brogioli, ¶¶127-128.

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 claim 1. As explained 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. As explained in Ground 1, [1b], Fox-Safford's lockstep processors and error detection logics render obvious "determining that the processor for which said LOL is detected is assigned a role of boot processor." Landry likewise renders this obvious. *Id.*, ¶¶129-130.

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 as explained for Ground 1, Section VII.B, and the reasons below. *Id.*, ¶¶131-135.

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; Brogioli, ¶¶45-47. Landry's fault-tolerant boot 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; Brogioli, ¶¶70-72. A POSITA would have understood Landry's process as a suitable option for selecting a boot processor in Fox, consistent with the initialization protocol of the IA-32 system. Brogioli, ¶132.

A POSITA would have understood the benefits of implementing Landry's boot processor selection 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, logical CPU0 was a known and conventional identification of the boot processor. Landry, Abstract, 1:6-10, 2:9-41, 6:44-7:52, Figs. 1, 4; Cepulis, Abstract, 2:30-64; Jin, [0048]; Brogioli, ¶¶48, 133.

A POSITA would have expected success in implementing Landry's process in Fox because Fox's IA-32 architecture provides the necessary hardware, firmware, and data structure for implementing the processor selection. Identifying the boot processor as logical CPU0 would require ordinary computer programming skills to assign logical CPU0 as the unique identification of the boot processor in Fox's system and store such unique identification in Fox's table accessible to the BIOS and operating system. Fox, [0026], [0033], [0035], [0038], [0041], [0045]-[0046]; IA-32, 294 (e.g., logical APIC ID); MP Specification, 75, 79; Brogioli, ¶134.

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. Fox, [0015], [0026], [0033], [0035], [0038], [0040]-[0041], [0045]-[0046]; Arai, [0019]-[0021], [0025]-[0026]; IA-32, 294 (e.g., logical APIC ID); MP Specification, 75, 79; Brogioli, ¶135.

C. Independent Claim 1

1. [1preamble]

Fox discloses [1preamble] as explained for Ground 1, [1preamble]. Brogioli, ¶137.

2. [1a]

Fox in view of Safford renders obvious [1a] as explained for Ground 1, [1a]. *Id.*, ¶138.

3. [1b]

As explained in Sections VII.B.1 and VII.C.2 (Ground 1, [1a]), a POSITA would have found it obvious to implement Fox's processors as lockstep processors with error detection logics and detect loss of lockstep between a lockstep processor pair as a processor failure to improve system reliability. As explained in Ground 1,

[1b], Fox-Safford renders obvious detecting loss of lockstep (LOL) for a processor implemented as a pair of lockstep processors. Fox-Safford-Landry renders obvious determining that the processor for which said LOL is detected is assigned a role of boot processor. Brogioli, ¶¶140-141.

Safford discloses a multiprocessor system having a plurality of lockstep processors pairs, 111/113, 121/123, and 125/127, where each pair appears as a single logical processor to the system. Safford, [0015], [0018]. Safford discloses processor pair 111/113 is logical CPU0 and processor pair 125/127 is designated as a spare processor pair. *Id.*, [0015], [0020]-[0021]. When a loss of lockstep error is detected in processor pair 111/113 (logical CPU0), the spare processor pair 125/127 replaces processor pair 111/113 and becomes the logical CPU0. *Id.*, [0021], FIGS. 3-4. Safford discloses determining that the processor for which said LOL is detected is logical CPU0 such that the logical CPU0 would be switched to the spare processor pair. Brogioli, ¶142.

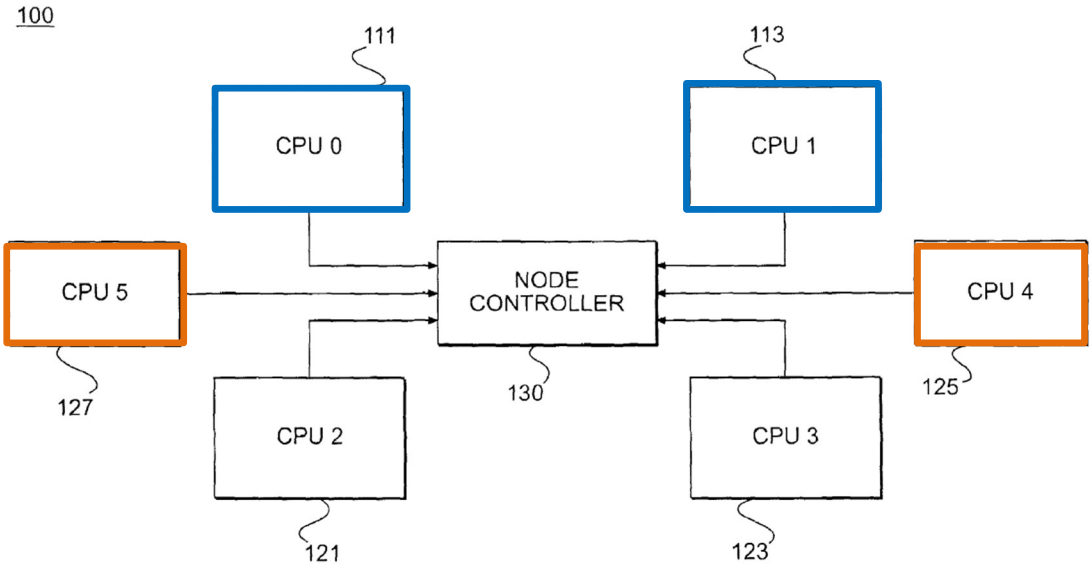


FIG. 3

Safford, FIG. 3.

As explained in Section VIII.B.1, a POSITA would have understood that Fox's system would have a boot processor, and would have implemented Landry's boot processor selection in Fox and identified logical CPU0 in Fox as the boot processor. Fox-Safford-Landry would have determined that the processor for which said LOL is detected is logical CPU0, assigned a role of boot processor. Brogioli, ¶143.

4. [1c]

Fox-Safford-Landry-Arai renders obvious [1c] as explained for Ground 1, [1c], and as explained for Ground 2, [1b]. *Id.*, ¶144.

5. [1d]

Fox-Safford-Landry-Arai renders obvious [1d] as explained for Ground 1, [1d], and as explained for Ground 2, [1b]. *Id.*, ¶145.

IX. Fox in View of Safford and Arai (Ground 1) or Fox in View of Safford, Landry, and Arai (Ground 2) Renders Obvious Claims 2-25

A. Independent Claim 13

1. [13preamble]: "A method comprising:"

Fox discloses [13preamble] as explained for Grounds 1-2, [1preamble]. *Id.*, ¶148.

2. [13a]: "system firmware determining that loss of lockstep (LOL) is detected for a lockstep pair of processors that are assigned the role of boot processor in a system;"

[13a] is similar to [1b] but recites "system firmware." As explained for Grounds 1-2, [1b], Fox-Safford or Fox-Safford-Landry renders obvious most of this limitation. Fox-Safford or Fox-Safford-Landry renders obvious that the BIOS ("system firmware") performs the determination. *Id.*, ¶¶149-150.

Fox's BIOS switches a failing processor to the spare processor. When a failure is detected in one of Fox's main processors, the processor failure response logic of the BIOS generates an SMI in response to detecting the processor failure to facilitate the processor replacement. Fox, [0015], [0032]. The SMI identifies the failing processor to the OS and the BIOS, and the BIOS, utilizing the SMI handler, replaces

the failing processor with the spare processor. *Id.*, [0041]-[0042], [0048]; Brogioli, ¶151.

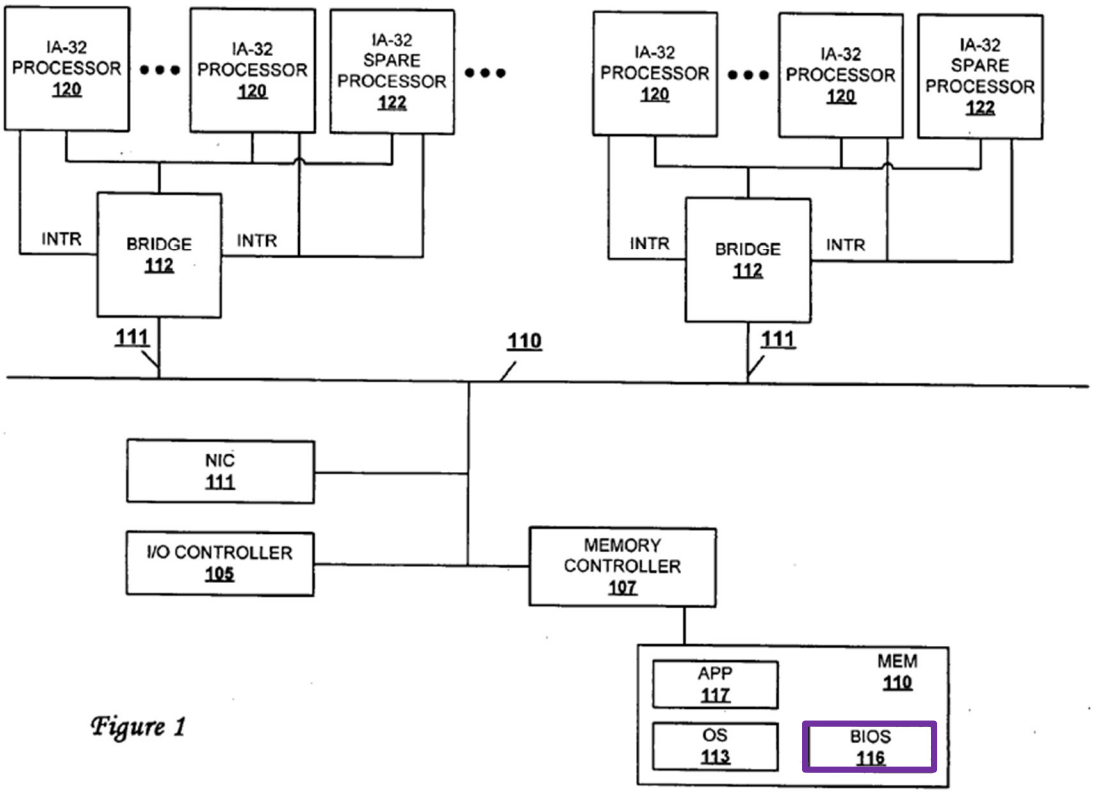


Figure 1

Fox, FIG. 1.

As explained for Ground 1, [1b] and Section VII.B.2, a POSITA would have understood that the identification of the failing processor by the SMI to the BIOS would have determined that the failing processor is the designated boot processor ("assigned the role of the processor") because such identification would be unique for the boot processor and known to the BIOS. Fox, [0042]; MP Specification, 75, 79. It would have been obvious to a POSITA that Fox's or Fox-Landry's BIOS

would have determined whether the failing processor is the boot processor to ensure performance of boot processor responsibilities during system runtime, improving system reliability. *See* Sections VII.B.2, VIII.B.2; Brogioli, ¶¶45-48. And in Fox-Safford or Fox-Safford-Landry, Fox's determining that the failing processor is assigned a role of boot processor would be "determining that the processor for which said LOL is detected is assigned a role of boot processor." *See* Sections VII.C.3, VIII.C.3. A POSITA would have understood that Fox's BIOS is "system firmware" that acts as an interface between the hardware and operating system. Brogioli, ¶¶53-55, 152.

3. [13b]: "determining one of said lockstep pair of processors that is not the cause of said LOL;"

As explained in Section VII.B.1 and Grounds 1-2, [1a], a POSITA would have found it obvious to implement the processors in Fox's multiprocessor systems as lockstep processors with error detection logics, with detecting a processor failure including "detecting loss of lockstep (LOL)" in a lockstep pair of processors. *Id.*, ¶154.

Safford's lockstep processor pair includes a logic circuit 115 to detect differences in the outputs of the pair of processors indicating a loss of lockstep and error detection, and signaling logics 112 and 114 to detect errors in the respective

processor cores to signal an impending loss of lockstep. Safford, [0016]-[0017]; Brogioli, ¶¶64-66, 155.

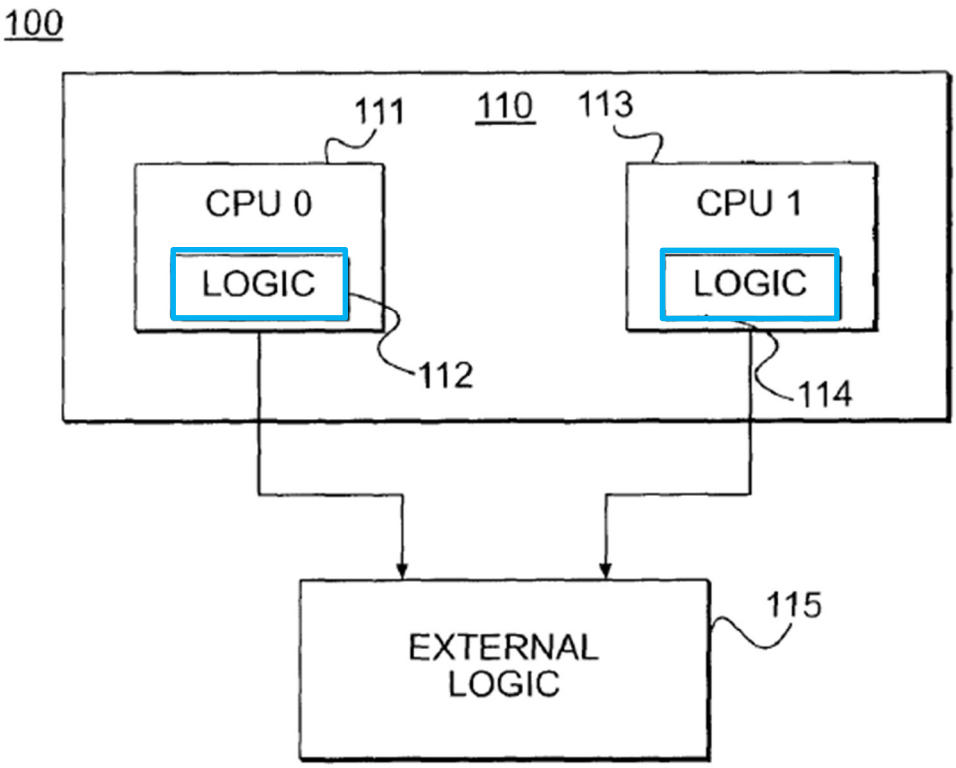


FIG. 2

Safford, FIG. 2.

The error detection and signaling logics 112 and 114, by detecting errors in the individual processors separately, signal to the logic circuit 115 which processor is in error and that the processor is a "bad" processor. *Id.*, [0017]. Logic circuit 115 turns off and processing continues using the other "good" processor. *Id.* Safford's error detection logics 112, 114, and 115 together determine the "bad" processor as

the processor that is experiencing an error and causing a loss of lockstep, and determine the "good" processor as the processor that is not the cause of the loss of lockstep. *Id.*; Brogioli, ¶¶34-36, 156.

Safford discloses saving the architected state of the "good" processor of the lockstep pair of processors and copying the architected state of the lockstep pair of processors that has lost lockstep to the hot standby processor pair. Safford, [0005], [0017], FIG. 4; *see also id.*, [0018], [0020]-[0021]. Fox discloses, during an SMI, saving the system state of each processor to memory, flushing the caches and pipelines, and switching the memory save state of the failing processor to the spare processor. Fox, [0041]. In Fox-Safford's lockstep processors and error detection logics, a POSITA would have been motivated to determine which processor is not the cause of loss of lockstep (e.g., Safford's "good" processor) such that the state of the processor without error would be switched to the spare processor and improve reliability in the processor recovery. Nguyen, [0046] (teaching localizing error to one of the execution cores to allow the machine state of the good core to be saved and used for recovery). A POSITA would have expected success in implementing Fox's processors as lockstep processors with error detection logics. *See* Section VII.B.1; Brogioli, ¶157.

4. [13c]: "copying the state of the determined one of said lockstep pair of processors that is not the cause of said LOL to a spare processor;"

As explained for [13b], Fox-Safford discloses the "good" processor "that is not the cause of said LOL." Brogioli, ¶¶158-159.

Fox discloses the BIOS switches the saved memory state of the failing processor to the spare processor during processor switching. Fox, [0041]; Brogioli, ¶160.

Safford discloses copying the state of the "good" processor to a spare processor. As explained for [13b], Safford discloses saving the architected state of the "good" processor and copying the architecture state of the lockstep processor pair 111/113 experiencing loss of lockstep to the hot standby processor pair 125/127 ("a spare processor"). Safford, [0005], [0017]-[0018], [0020]-[0021], FIG. 4; Brogioli, ¶161.

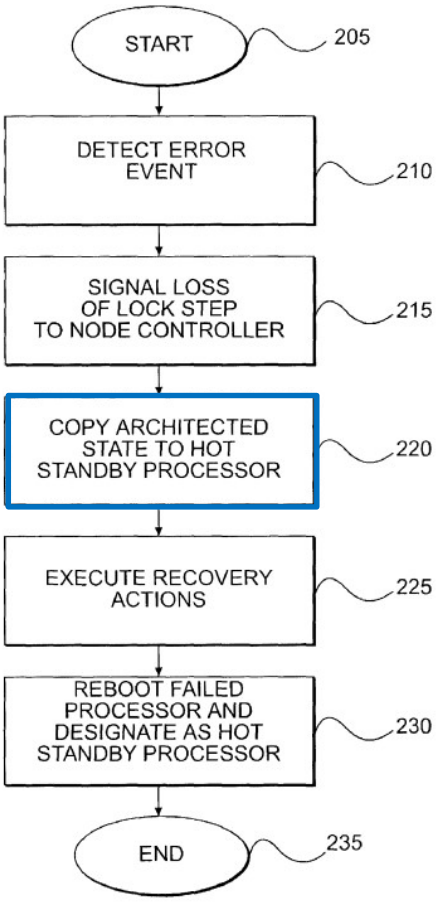


FIG. 4

Safford, FIG. 4.

As explained for [13b], in Fox-Safford's lockstep processors and error detection logics, a POSITA would have been motivated to determine which processor is not the cause of loss of lockstep (e.g., Safford's "good" processor) and copy the state of this processor to the spare processor as taught by Fox and Safford to improve reliability in the processor recovery. Nguyen, [0046] (saving the machine state of the good processor core of a lockstep processor pair to use the saved machine

state to recover the processor cores); Arai, [0005] ("it is very desirable not to lose the work being done" on the processor that encounters an error); Brogioli, ¶162.

5. [13d]: "changing a system identifier, used by the system's operating system to recognize the boot processor, of said lockstep pair of processors that are assigned the role of boot processor, to a different value, which causes the system's operating system to not recognize the lock step pair of processors for which said LOL is detected as the boot processor; and"

[13d] is similar to [1c]. Fox-Safford-Arai or Fox-Safford-Landry-Arai renders obvious [13d] as explained for Grounds 1-2, [1c]. Brogioli, ¶163.

6. [13e]: "changing a system identifier of the spare processor to that of the boot processor, thereby switching the role of boot processor to said spare processor."

[13e] is similar to [1d]. Fox-Safford-Arai or Fox-Safford-Landry-Arai renders obvious [13e] as explained for Grounds 1-2, [1d]. *Id.*, ¶164.

B. Independent Claim 21

1. [21preamble]: "A system comprising:"

Fox discloses "a system for dynamically replacing a failing processor" with a spare processor. Fox, Abstract, [0002], [0014], [0023]-[0024]; Brogioli, ¶166.

2. [21a]: "a plurality of processor modules that each include a master processor and a slave processor that operate in lockstep;"

Fox discloses a multiprocessor computer system configured with IA-32 architecture, the system including a spare processor and main processors ("a

plurality of processor modules"). Fox, [0019], [0023]-[0025]; Brogioli, ¶¶59-63, 168.

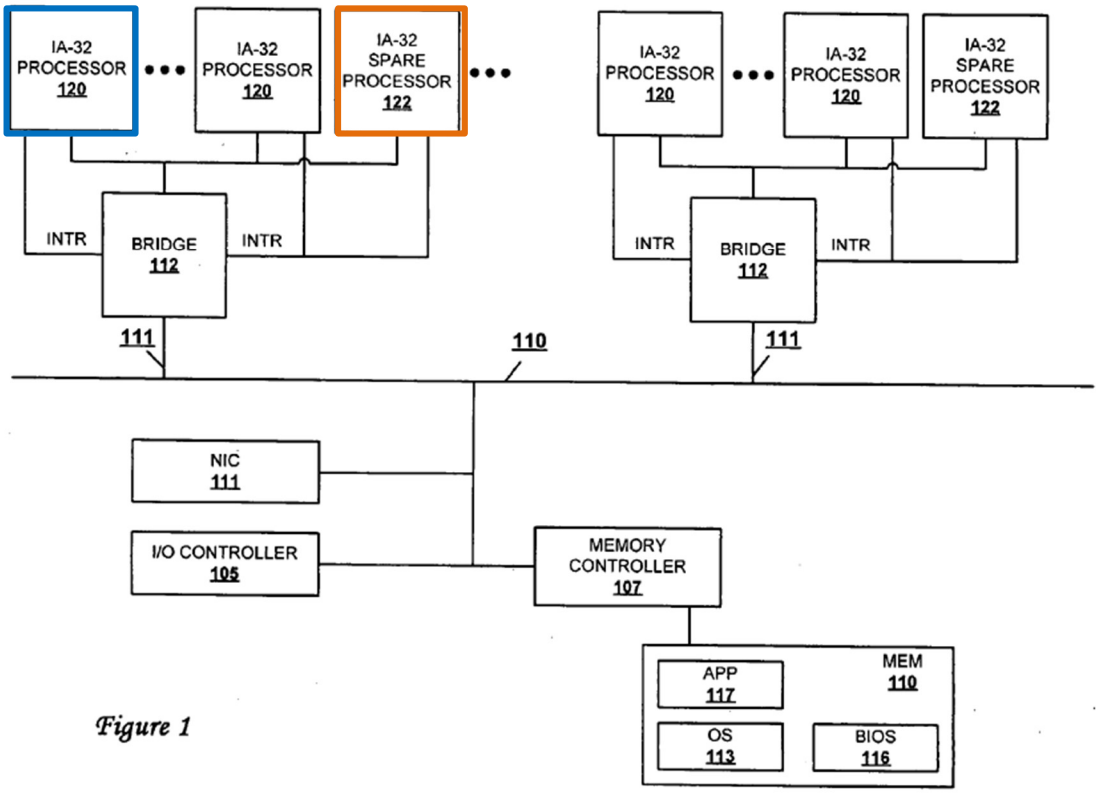


Figure 1

Fox, FIG. 1.

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. Safford discloses its lockstep processor pair "include[s] a master processor and a slave processor." Brogioli, ¶169.

Each of Safford's 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's lockstep processor pair includes a master/checker pair. *Id.*, [0015]; Brogioli, ¶170.

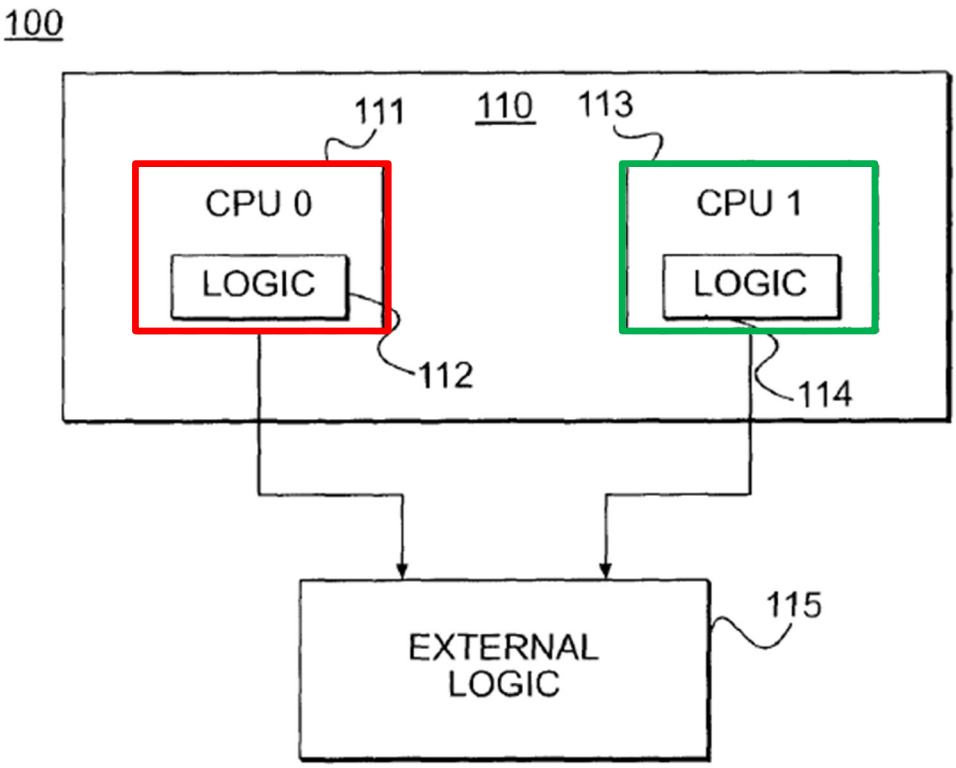


FIG. 2

Safford, FIG. 2.

Safford 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]; Brogioli, ¶171. Safford's lockstep processor's master/checker pair

constitutes "a master processor and a slave processor that operate in lockstep." Brogioli, ¶¶172-174.

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.*, ¶173 (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] (master and slave designations can be implemented using a status bit and the designations can be changed dynamically). Brogioli, ¶174.

As 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, including a master processor and a slave processor. Brogioli, ¶175.

3. [21b]: "an operating system;"

Fox's multiprocessor system includes a system memory 110, stored with software for controlling the system, the software including operating system 113, as

shown below. Fox, [0028]; FIG. 1. Fox's operating system is executed by the processors and controls work allocation to the processors. *Id.*, [0032], Claim 12. Fox's operating system continues to execute during switching of a failing processor to the spare processor. *Id.*, [0012]-[0016], [0023], [0032], [0044]; Brogioli, ¶¶176-177.

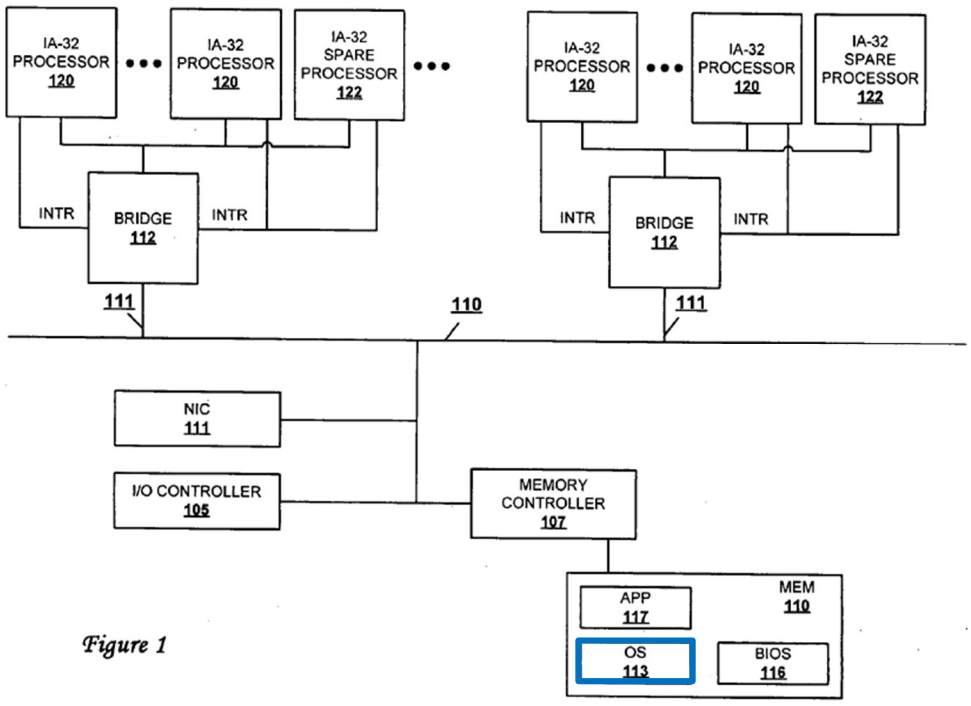


Figure 1

Fox, FIG. 1.

4. [21c]: "error detection logic operable to detect loss of lockstep (LOL) for at least one of said processor modules; and"

[21c] is similar to [1a] but recites "error detection logic" to detect loss of lockstep. Fox-Safford renders obvious [21c] as explained for Grounds 1-2, [1a]. Brogioli, ¶178.

As discussed in Section VII.B.1, a POSITA would have implemented Fox's processors as lockstep processors with error detection logics as taught in Safford. As explained for Grounds 1-2, [1a], Fox-Safford discloses detecting loss of lockstep (LOL) for a processor of the multiprocessor system ("for at least one of said processor modules"). In Fox-Safford, error detection logics of the lockstep processor pair detect loss of lockstep. Brogioli, ¶179.

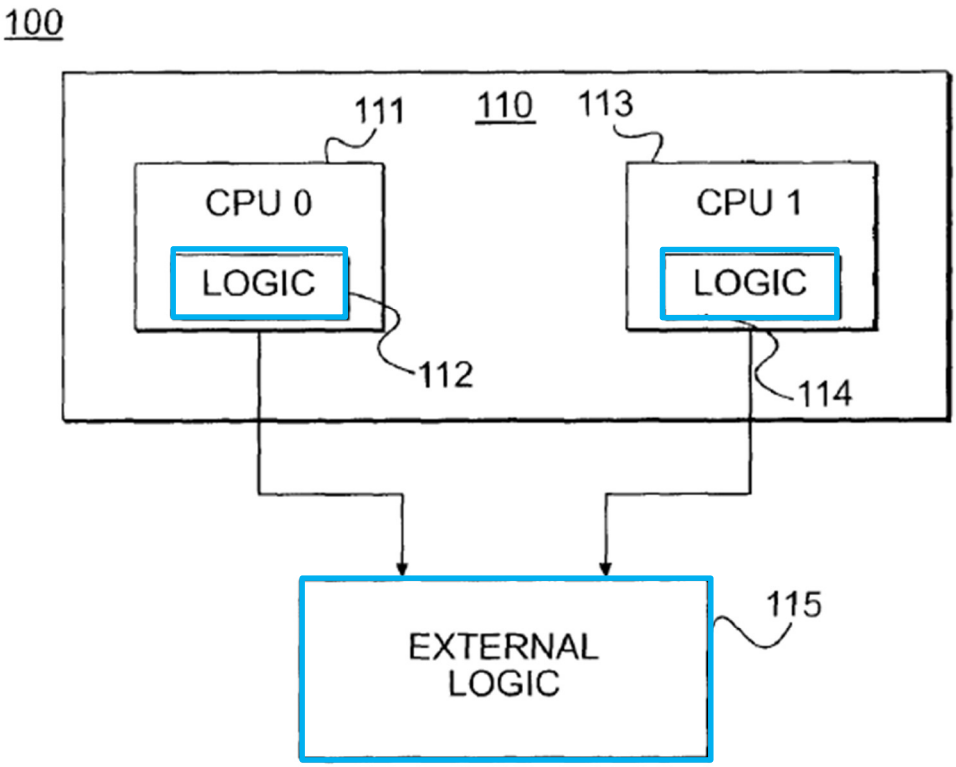


FIG. 2

Safford, FIG. 2; Brogioli, ¶¶64-66.

A POSITA would have known that such error detection logics would be implemented in hardware, firmware, or a combination of hardware and firmware. Bigbee, [0034] (FRC logic may be implemented in firmware); Safford-181, 15:60-64 (lockstep logic may be implemented in firmware); Brogioli, ¶180.

5. [21d]: "system firmware operable, responsive to the detection of LOL for a first of said processor modules, to determine whether said first processor module is assigned a role of system boot processor;"

[21d] is similar to [13a] but recites "responsive to the detection of LOL for a first of said processor modules." Fox-Safford or Fox-Safford-Landry renders obvious [21d] as explained for [13a] and reasons below. Brogioli, ¶¶181-184.

As explained for [13a], Fox-Safford or Fox-Safford-Landry renders obvious that Fox's system firmware performs the determination that LOL is detected for the boot processor. That determination is responsive to the detection of LOL for a processor. Brogioli, ¶182.

As explained for [13a], Fox discloses when a failure is detected in one of its main processors, the processor failure response logic of BIOS generates an SMI in response to detecting the processor failure to facilitate replacement of the failing processor with the spare processor by the BIOS. Fox, [0015], [0032], [0041]-[0042], [0048]; Brogioli, ¶183.

As explained for Grounds 1-2, [1a], a POSITA would have understood that in Fox-Safford, detecting a processor failure would have included "detecting loss of lockstep (LOL)." In Fox-Safford, the generation of SMI by the BIOS ("system firmware") to identify the failing processor and facilitate the processor switching by

the BIOS would have been "system firmware operable, responsive to the detection of LOL." Brogioli, ¶184.

6. [21e]: "wherein if determined that said first processor module is assigned the role of system boot processor, said system firmware is further operable to cause said operating system to recognize another processor module as system boot processor without shutting down said operating system."

Fox-Safford-Arai or Fox-Safford-Landry-Arai renders obvious [21e] as explained for Grounds 1-2, [1b]-[1d], [13a], and [21d], and reasons below. Brogioli, ¶¶185-190.

As explained for [13a], Fox-Safford or Fox-Safford-Landry renders obvious "system firmware" determining that loss of lockstep (LOL) is detected for a lockstep pair of processors ("first processor module") that are assigned the role of boot processor in a system. Brogioli, ¶186.

As explained for Grounds 1-2, [1c], Fox discloses saving processor identifications in a register or table (e.g., BIOS table, ACPI table, MP table) accessible to the BIOS and operating system. Fox, [0026], [0033], [0035], [0038], [0041], [0045]-[0046], Claims 2, 13; IA-32, 273-274, 267, 269, 289-297, 745 (e.g., local APIC ID or logical APIC ID). Fox's register or table indicates currently available processors to the operating system. Fox, [0035], Claim 13; Arai, [0025]

(ACPI table or MP table to identify location and status of each processor to the operating system); Brogioli, ¶187.

As explained for Grounds 1-2, [1c]-[1d], in Fox-Safford-Arai or Fox-Safford-Landry-Arai, the logical identifications (e.g., logical APIC IDs) of the lockstep processor pair of the boot processor detected to have LOL and the spare processor pair would have been switched. Safford, [0021] (teaching the spare processor becomes the new logical CPU0). This identification switching results in the unique logical identification of the boot processor (e.g., logical CPU0), as known to the operation system as the spare processor. A POSITA would have understood that such identification switching would have caused Fox's operating system to recognize the spare processor as the boot processor based on the unique logical identification of the boot processor. Brogioli, ¶188.

A POSITA would have understood that Fox's BIOS ("system firmware") would have performed the switching 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, ¶189.

A POSITA would have understood that the processor switching by the BIOS of Fox-Safford-Arai or Fox-Safford-Landry-Arai would be performed "without

shutting down said operating system" because it was well known that using firmware to switch processor logical identifications would allow the operating system to continue execution, improving system availability. Slegel, 3:7-46, 6:16-36; Armstrong, Abstract, [0019]-[0020]; Zorek, Abstract, 2:61-3:17; Fox, [0014], [0044], [0047]; Brogioli, ¶¶53-55, 190.

C. Claims 2, 14, and 23

1. Claims 2 and 14: "wherein the spare processor is a hot spare that is held in idle until switched the role of 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 idle"), and while in that mode, the OS is prevented from initially utilizing the hot-spare processor. Fox, Abstract, [0015], [0040], FIG. 1; Brogioli, ¶193.

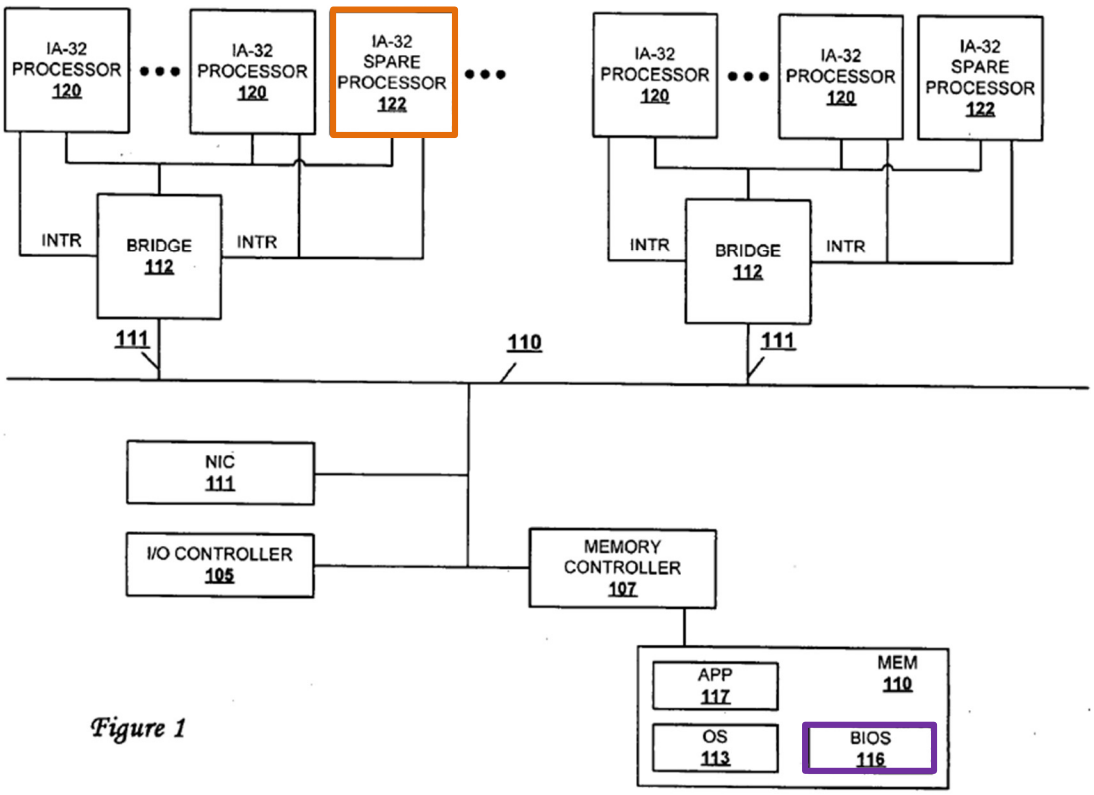


Figure 1

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 ("held in idle until switched"). Fox, [0040]-[0041]; Brogioli, ¶194.

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 idle"). Safford, [0020]. 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, ¶195.

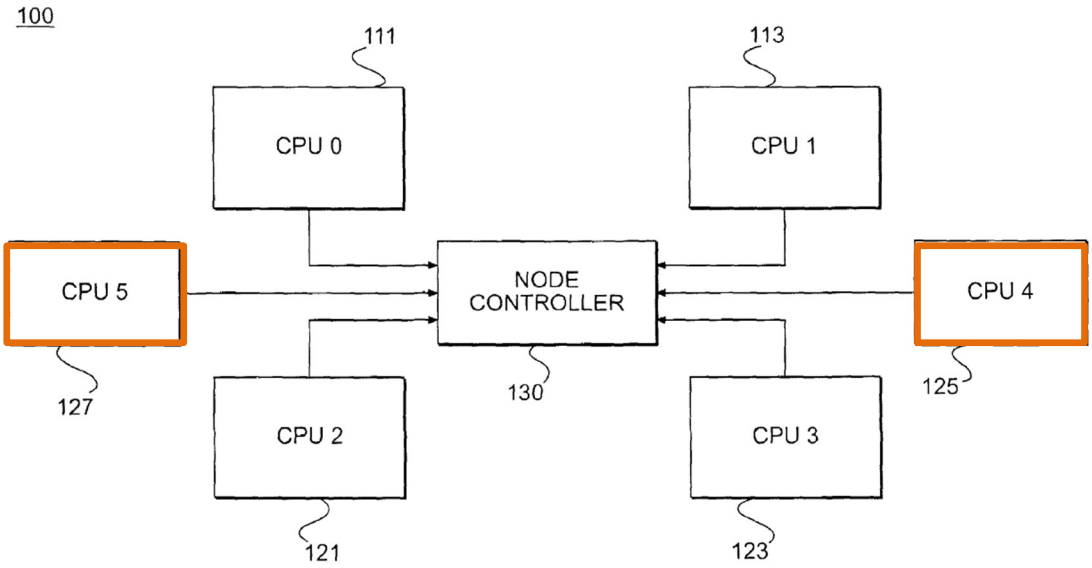


FIG. 3

Safford, FIG. 3.

A POSITA would have understood that in Fox-Safford, Fox's hot-spare processor would be a hot-spare processor pair "held in idle" in lockstep mode until being switched with the role of the boot processor, as explained for Grounds 1-2, [1b] and [1d]. Brogioli, ¶196.

2. Claim 23: "wherein the another processor module is held in idle until said system firmware causes said operating system to recognize said another processor module as said system boot processor."

As explained for [21e], Fox-Safford-Arai or Fox-Safford-Landry-Arai renders obvious "system firmware [that] causes said operating system to recognize said

another processor module as said system boot processor." As explained for claims 2 and 14, Fox-Safford renders obvious the hot-spare processor ("the another processor module") is held in idle until being switched to the role of boot processor. As explained for Grounds 1-2, [1d] and [21e], a POSITA would have understood that switching the role of the spare processor includes changing the identification of the spare processor ("the another processor module") to that of the boot processor, which causesthe operating system to recognize the spare processor as the boot processor. Brogioli, ¶198.

D. Claims 3 and 15: "wherein the spare processor is an active spare."

Fox discloses that, "in another embodiment[,] the system is made to enable the 'hot spare processor' feature on-the-fly by the OS with a driver-OS-SMI handshake. Tracking which processors are active may be completed by [a] software table of active processors, which is available to be read by the OS, when determining workload allocation to each of the active processors." Fox, [0035]. A POSITA would have understood that configuring a processor "on-the-fly" means that 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, and such active processor is "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, ¶200.

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 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 expect success. Brogioli, ¶201.

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. 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 data structures 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 and flags. IA-32, 276-277, 279, 304-305 (teaching software can disable or enable a local APIC by setting the software enable/disable flag); MP Specification, 44 (CPU flags can be set

to zero so that operating system does not access this processor); Brogioli, ¶¶201-202.

E. Claims 4 and 16: "causing the system's operating system to idle the active spare."

As explained for claims 3 and 15, a POSITA would have understood that Fox discloses or renders obvious "an active spare" to be idled when needed during system runtime, and that Fox's system allows using software to idle the active spare. A POSITA would have understood Fox's operating system to have been a suitable option to idle the active spare because the operating system would have access to the register of the processor, such as the APIC register, and set the processor flag indicating the state of the processor to the operating system. Fox, [0041] (providing a software register of active processors in the operating system), [0035] (using a software table to track active processors available to the operating system); Brogioli, ¶¶203-204.

F. Claim 5: "wherein the spare processor is a processor module that includes a master and slave processors that operate in lockstep."

As explained for [21a], Safford teaches a lockstep processor ("a processor module") that includes "a master and slave processors that operate in lockstep." As explained in Section VII.B.1, a POSITA would have found it obvious to implement Fox's processors as lockstep processors, each including a master processor and a slave processor.

G. Claims 6 and 25

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

As explained for Grounds 1-2, [1b], Fox-Safford or Fox-Safford-Landry renders obvious "determining that the processor for which said LOL is detected is assigned the role of boot processor." Fox-Safford-Arai or Fox-Safford-Landry-Arai renders obvious "accessing information stored to non-volatile data storage that identifies one of the processors in said multi-processor system that is assigned the role of boot processor." *Id.*, ¶¶207-208.

As explained for Grounds 1-2, [1c], Fox discloses saving processor identifications in a register or table (e.g., BIOS table, ACPI table, MP table) accessible to the BIOS and operating system. Fox, [0026], [0033], [0035], [0038], [0041], [0046]; IA-32, 273-274, 267, 269, 289-297, 745 (e.g., local APIC ID or logical APIC ID). A POSITA would have understood that each processor would have a unique identification saved in Fox's register or table, including the boot processor and its BSP flag ("information … that identifies one of the processors in said multi-processor system that is assigned the role of boot processor"). Brogioli, ¶¶46-47. Fox's BIOS would have accessed the register or table ("accessing

information") to determine the failing processor identification and BSP flag. Fox, [0042] (SMI identifies to the BIOS which one of the processors is failing). Grounds 1-2, [1b]; Brogioli, ¶209.

Fox discloses a system memory for storing the BIOS, but does not expressly disclose the processor register or table is stored in the system memory. Fox, [0028], FIG. 1. Arai discloses a table that identifies all processors in the system (e.g., ACPI table or MP table) is stored in a main memory ("data storage"). Arai, [0019]-[0021], [0025]-[0026]. Both Fox and Arai disclose using registers or tables to store processor information available to the OS and the BIOS, and use the stored information to implement switching. A POSITA would have found it suitable to store Fox's register or table in its system memory ("data storage"), like in Arai. Fox's system memory is "non-volatile" because it was common and well known that the BIOS would be embedded in a non-volatile memory (e.g., ROM). Fox, [0028], FIG. 1; Brogioli, ¶210.

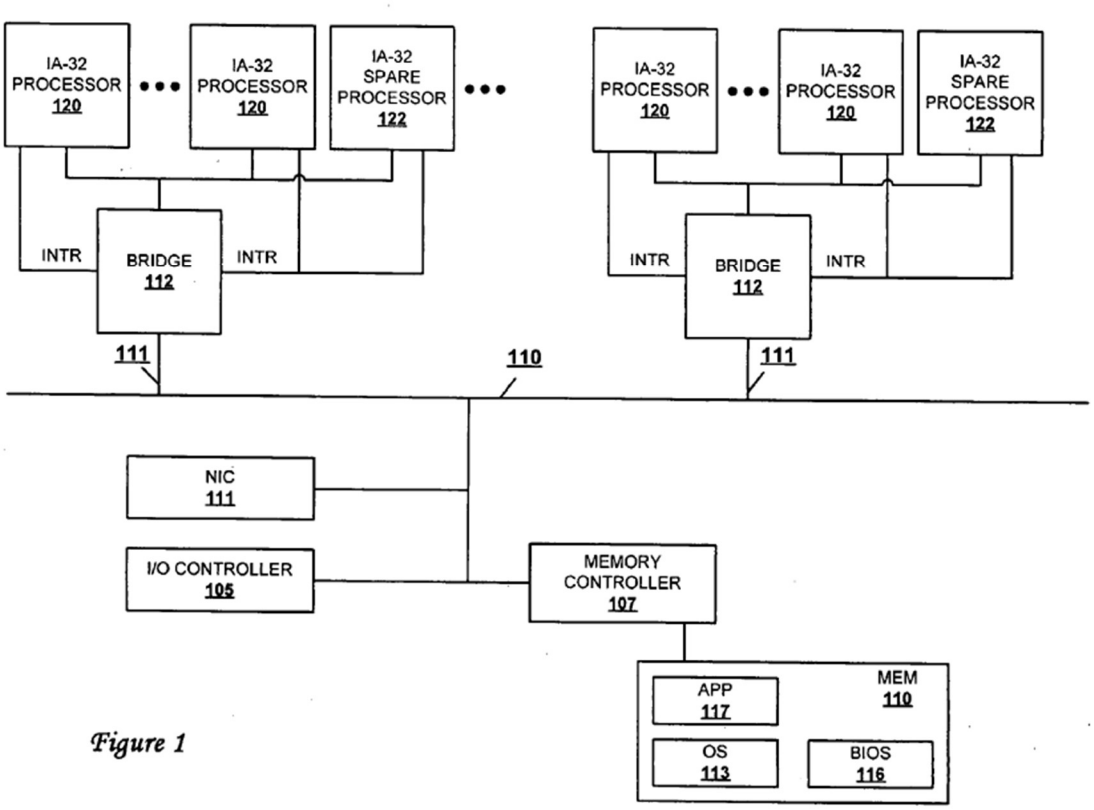


Figure 1

Fox, FIG. 1.

A POSITA would have found such modification routine because it was well known that the processor register or table should be stored in a memory and accessible to the BIOS. 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, ¶211.

The ’302 Patent similarly discloses a device tree or ACPI table that is stored in a RAM, accessible to the firmware. Ex. 1001, 9:4-15, 12:60-66, 13:21-44; Brogioli, ¶212.

2. Claim 25: "non-volatile data storage storing information that assigns the role of system boot processor to one of said plurality of processor modules."

As explained for claim 6, Fox-Safford-Arai or Fox-Safford-Landry-Arai renders obvious the system memory ("non-volatile data storage") storing a processor register or table containing the unique identifications of the processors, including the boot processor and its BSP flag ("information that assigns the role of system boot processor to one of said plurality of processor modules"); Brogioli, ¶¶213-214.

H. Claims 7, 8, and 22

1. Claim 7: "determining the processor that is assigned the role of spare processor."

Fox discloses the spare processor is reserved and identified with a software tag during system initialization ("assigned the role of spare processor"), and in response to detecting a processor failure, the BIOS switches the failing processor to the spare processor. Fox, [0032]-[0034], [0040]-[0042]; Brogioli, ¶¶59-63, 216.

Fox does not explicitly disclose determining the processor that is assigned the role of spare processor before switching. Arai does. Brogioli, ¶¶67-69. 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 available to replace the active processor ("determining the processor that is assigned the role of spare processor"). Arai, [0022], FIG. 3. Arai generates 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 examine the table and determine the processors that are in an inactive, reserved, or idle state. *Id.*, [0019]-[0021]; Brogioli, ¶¶216-217.

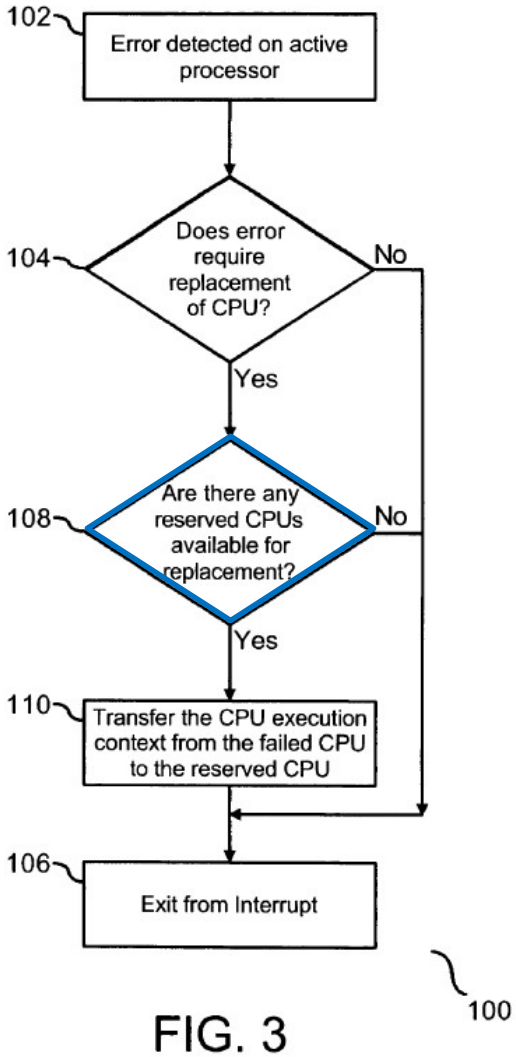


FIG. 3

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 before switching the failing processor to the spare processor in Fox. A POSITA would have expected success in adding Arai's determination to using routine computer programming skills. Both Arai and Fox use the BIOS interrupt handler to facilitate the processor switching. *Id.*, [0022]; Fox, [0041]. Like Arai, Fox also includes a register or table (e.g., BIOS table, ACPI table, MP table) accessible to the BIOS and operating system. Fox, [0026], [0033], [0035], [0038], [0041], [0046]; IA-32, 273-274, 267, 269, 289-297, 745 (e.g., local APIC ID or logical APIC ID). A POSITA would have understood that such table would include the information of the state of the processor in the form of a software tag or flag. Fox, [0034] (a software tag to identify the processor as a spare processor); IA-32, 305 (Figure 8-22 showing APIC software Enable/Disable Flag); Brogioli, ¶¶218-219.

The '302 Patent similarly discloses the system firmware examines the device tree to determine which processor is an active spare. Ex. 1001, 17:15-19, FIG. 5; Brogioli, ¶220.

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

As explained for claim 7, Fox-Arai renders obvious examining a register or table (e.g., ACPI table, MP table, or BIOS table) to determine which spare processor is available to replace the failing processor ("accessing information … that identifies one of the processors in said multi-processor system that is assigned the role of spare processor"). As explained for claim 6, a POSITA would have found it obvious to store such information in Fox's memory ("non-volatile data storage"); Brogioli, ¶¶221-222.

3. Claim 22: "wherein the another processor module is a processor module assigned a role of spare processor."

Fox discloses setting a software tag to identify a processor as a spare processor during system BIOS boot ("assigned a role of spare processor") and replacing the failing processor with the spare processor. Fox, [0023], [0034], [0046]. As explained for claim 7, Fox-Arai renders obvious a table that identifies each processor and the state of each processor (e.g., reserved, idle, or inactive), and examines the table to determine the processor that is assigned the role of spare processor. Arai, [0019]-[0021], [0025]-[0026]; Brogioli, ¶¶223-224.

I. Claims 9, 10, and 18-20

1. Claims 9 and 18: "wherein said determining step is performed by system firmware."

Fox discloses claims 9 and 18 as explained for [13a]. Brogioli, ¶225.

2. Claims 10 and 20: "wherein said switching step is performed by system firmware."

As explained for [13a], a POSITA would have understood Fox's BIOS as system firmware. As explained in Ground 1, [1d], Fox discloses that the BIOS performs the switching of the failing processor to the spare processor. Fox, [0041]; Brogioli, ¶¶59-63, 226.

3. Claim 19: "wherein said copying step is performed by system firmware."

As explained for [13a], a POSITA would have understood Fox's BIOS as system firmware, and Fox's BIOS switches the memory state of the failing processor to the spare processor during processor switching. Fox, [0041]; Brogioli, ¶¶227-228.

As explained for [13c], Fox-Safford renders obvious the "copying step." A POSITA would have understood that in Fox-Safford, the "copying step" would be performed by the BIOS. Brogioli, ¶229.

X. Fox in View of Safford, Arai, and Bigbee (Ground 3) or Fox in View of Safford, Landry, Arai, and Bigbee (Ground 4) Renders Obvious Claims 26-27

A. Overview of the Combinations

Fox-Safford-Arai (Ground 1) or Fox-Safford-Landry-Arai (Ground 2) renders obvious system firmware "detecting loss of lockstep (LOL) for a processor in a multi-processor system" (Grounds 1-2, [1a]), system firmware for "determining whether the processor for which said LOL is detected is assigned the role of boot processor" (Grounds 1-2, [13a]), and system firmware for "switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which said LOL is detected is assigned the role of boot processor" (Grounds 1-2, [1d], Claims 10, 20). A POSITA would have understood that Fox's BIOS is system firmware. *See* [13a]; Brogioli, ¶¶53-55, 152. To the extent that the Board finds 35 U.S.C. § 112, ¶6, applies to "system firmware means" (*see* Section V.B) and construes them to require additional firmware structures, they are rendered obvious in view of Bigbee. Brogioli, ¶¶230-254.

B. A POSITA Would Have Found It Obvious to Use Itanium Processor Family Processors and Firmware as Taught in Bigbee to Improve System Performance and Capacity

Fox's system is configured with IA-32 architecture and comprises IA-32 processors and a BIOS of the IA-32 architecture. Fox, Abstract, [0014]-[0015], [0024], [0028], [0033]. Bigbee discloses a multiprocessor system comprising

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's system's IA-32 processors and architecture to more advanced Itanium family processors to improve performance and processing capacity for high-end server systems. Brogioli, ¶232.

A POSITA would have known that 64-bit instruction set architecture is designed to operate on 64-bit data, enabling more advanced computational tasks than 32-bit. Systems configured with 64-bit architecture, like the Itanium family processors, would run more processes than those configured with 32-bit architecture because the processors have increased computing resources, buses, memories, and caches, allowing for increased throughput. Ex. 1008; Ex. 1030, 19; Brogioli, ¶233. A POSITA also would have understood that the firmware model of Itanium family processors is ACPI-compliant and provides numerous advantageous over legacy BIOS solutions. Ex. 1029, 15; Brogioli, ¶234.

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. A POSITA would have

known the Itanium-based firmware model and how the various firmware layers (PAL, SAL, EFI) work together. Ex. 1030, 267-276. Intel and other sources have extensively documented the Itanium architecture, including its firmware model (Ex. 1030), and Itanium family processors had been used in many fault-tolerant multiprocessor systems like that of Fox. Bigbee, [0033]; Nguyen, [0045]; Brogioli, ¶235.

C. Independent Claim 26

1. [26preamble]: "A system comprising:"

Fox discloses [26preamble] as explained for Grounds 1-2, [21preamble]. Brogioli, ¶237.

2. [26a]: "system firmware means for detecting loss of lockstep (LOL) for a processor in a multi-processor system;"

As explained in Section V.B, this limitation should be construed to have the claimed function of "detecting loss of lockstep (LOL) for a processor in a multi-processor system," and the corresponding structure in the specification is firmware including a Processor Abstraction Layer (PAL), System Abstraction Layer (SAL), and Extended Firmware Interface (EFI) or equivalents thereof. Ex. 1001, 8:34-59, 11:4-60, 12:36-59, FIG. 2. Bigbee discloses this structure, and Fox-Safford-Bigbee renders obvious using this structure to perform the claimed function. Brogioli, ¶¶38, 238-243.

As explained for Grounds 1-2, [1a], Safford discloses a pair of processors in lockstep and error detection logics for detection loss of lockstep (Safford, [0017]-[0018]), and Fox-Safford renders obvious detecting LOL. Brogioli, ¶240.

Like Safford, Bigbee discloses a logical processor comprising two processor cores operating in FRC mode (lockstep) and FRC logic for detecting loss of lockstep between the two processor cores, as shown below. Bigbee, [0019], [0032]-[0033], FIG. 3; Brogioli, ¶241.

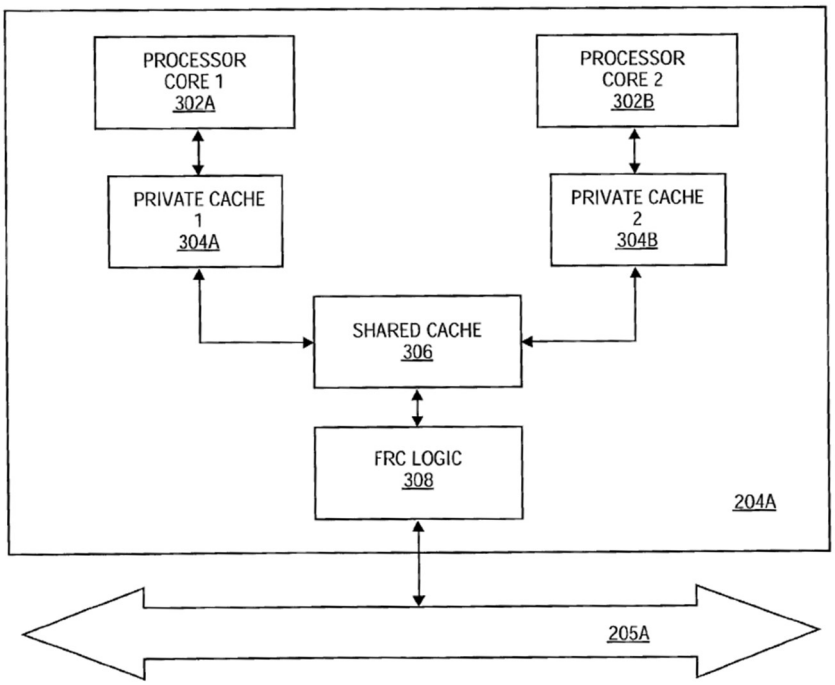


FIG. 3

Bigbee, FIG. 3.

Bigbee discloses implementing its FRC logic in executable firmware. *Id.*, [0034]. A POSITA would have implemented the error detection logics of Fox-

Safford in firmware as taught in Bigbee because firmware is one of three suitable options for detecting loss of lockstep—firmware, hardware, or a combination—all of which would have been obvious to try, and any of which would be a suitable alternative for the others. A POSITA also would have understood the benefits of using firmware because firmware is flexible. It can be modified after production, updated, and supported relatively quickly and inexpensively. Brogioli, ¶242.

Bigbee's processors are Itanium family processors and its firmware includes "an extensible firmware interface (EFI), a system abstraction level (SAL), and a processor abstraction level (PAL)." Bigbee, [0018], [0034]. As explained in Section X.B, a POSITA would have found it obvious to configure Fox's system with Itanium processors and its firmware model comprising EFI, SAL, and PAL as disclosed in Bigbee. Brogioli, ¶243.

3. [26b]: "system firmware means for determining whether the processor for which said LOL is detected is assigned the role of boot processor; and"

As explained in Section V.B, this limitation should be construed to have the claimed function of "determining whether the processor for which said LOL is detected is assigned the role of boot processor," and the corresponding structure in the specification is firmware including a Processor Abstraction Layer (PAL), System Abstraction Layer (SAL), Extended Firmware Interface (EFI), and a device tree, or

equivalents thereof. Ex. 1001, 5:34-48, 9:49-60, 11:4-60, 12:36-14:3, 18:50-58, FIG. 2. Bigbee discloses this structure, and Fox-Safford-Bigbee or Fox-Safford-Landry-Bigbee renders obvious using this structure to perform the claimed function. Brogioli, ¶¶38, 245.

As explained for [26a], Bigbee's Itanium processors include "an extensible firmware interface (EFI), a system abstraction level (SAL), and a processor abstraction level (PAL)." Bigbee, [0018], [0034]; Brogioli, ¶246.

Bigbee discloses a device tree or its equivalent as described in the '302 Patent. Bigbee's system comprises "an advanced configuration and power interface (ACPI)-compliant processing system," and the operating system includes an ACPI device driver and firmware that includes "ACPI machine language (AML) program code describing what hardware devices are present within data processing system 100, as well as their configuration and interfaces, via ACPI control methods, objects, registers, and tables." Bigbee, [0016]. This is like the '302 Patent disclosing that the firmware builds a device tree, converts this information to ACPI tables, and presents it to the operating system. Ex. 1001, 12:60-13:2; Brogioli, ¶247.

A POSITA also would have understood that Fox's register or table (e.g., BIOS table, ACPI table, MP table) is created by the BIOS (firmware) during system

initialization as a device tree or its equivalent. Fox, [0032]-[0033], [0035]; Brogioli, ¶248.

4. [26c]: "system firmware means for switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which said LOL is detected is assigned the role of boot processor."

As explained in Section V.B, this limitation should be construed to have the claimed function of "switching the role of boot processor to a spare processor without shutting down the system's operating system if determined that the processor for which said LOL is detected is assigned the role of boot processor," and the corresponding structure in the specification is firmware including a Processor Abstraction Layer (PAL), System Abstraction Layer (SAL), and Extended Firmware Interface (EFI) or equivalents thereof. Ex. 1001, 11:4-60, 12:36-14:3, 17:12-18:43, FIG. 2. Bigbee discloses this structure, and Fox-Safford-Arai-Bigbee or Fox-Safford-Landry-Arai-Bigbee renders obvious using this structure to perform the claimed function. Brogioli, ¶¶38, 249-250.

As explained for Grounds 1-2, [1d], claims 10 and 20, Fox-Safford-Arai (Ground 1) or Fox-Safford-Landry-Arai (Ground 2) renders obvious system firmware (e.g., BIOS) for switching the role of boot processor to a spare processor without shutting down the system's operating system. *Id.*, ¶251.

As explained for [26a], Bigbee's firmware includes "an extensible firmware interface (EFI), a system abstraction level (SAL), and a processor abstraction level (PAL)." Bigbee, [0018]. As explained in Section X.B, a POSITA would have found it obvious to configure Fox's system with Bigbee's Itanium architecture, including its firmware model comprising EFI, SAL, and PAL. Brogioli, ¶252.

D. Claim 27: "wherein said processor in said multi-processor system is an Itanium Processor Family (IPF) processor."

Bigbee discloses using Itanium family processors in a multiprocessor system. Bigbee, [0033]. For reasons discussed in Section X.B, a POSITA would have been motivated to upgrade Fox's system to use Itanium family processors and architecture and expect success. Brogioli, ¶¶253-254.

XI. Discretionary Denial Under *Fintiv* 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).

XII. 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 and Landry were not before the Examiner. None of the grounds presented in this Petition were addressed during prosecution of the '302 Patent.

Safford was cited by the Examiner in a non-final office action that rejected all pending claims as anticipated or obvious. Ex. 1004, 242-261. Applicant argued that Safford does not disclose "the boot processor elements." *Id.*, 232-234. The Examiner subsequently maintained the Safford anticipation rejections. *Id.*, 204-209. Applicant maintained its argument that Safford does not disclose a boot processor. *Id.*, 196-197. The Examiner never applied Safford in a combination again.

Fox was cited by the Examiner in a non-final office action rejection of claims 1-23 and 29-30 as obvious over Quach (U.S. Patent No. 6,625,749) in combination with Fox. *Id.*, 209-216. The Examiner relied on Fox for "system response to processor failure in a multi-processor data processing system" and "dynamically activate spare processor when processor failure is detected in a multi-processor data processing system," but did not rely on Fox for disclosure of a boot processor. *Id.* 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.*, 160, 197-198.

Bigbee was cited on an IDS submitted by Applicant. *Id.*, 145. The Examiner did not analyze Bigbee in an office action (*see* Ex. 1004) or Notice of Allowance (*id.*, 30-33, 160-163).

Denial is at least 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. None of Safford, Fox, or Bigbee are relied on for the allowable subject matter identified during prosecution, "changing an identifier" and "switching the role." *Id.*, 177-179. Instead, Arai, which was not before the Examiner, is relied on here.

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

First, the Examiner did not reapply Safford in a combination. *See* Ex. 1004; *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 system would include a boot processor. As discussed above for Grounds 1-2, Fox repeatedly references IA-32, which shows that one of Fox's processors would be designated a boot processor. The Examiner also overlooked that Fox's spare processor would be used for replacing the boot processor in which a failure is detected. *See* Section VII.B.2.

Third, Grounds 3-4 show the Office made a material error by not using Bigbee. The Examiner did not discuss or rely on Bigbee.

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

Discretionary denial in view of Unified's *ex parte* reexamination ("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. 1031, 775-853.

Ground	Claims	Basis	Reference(s)
1	1, 5-8, 13, 17, 21, 22, 25-26	§ 103	Kalyanasundharam in view of Kennedy
2	1-27	§ 103	Kennedy in view of Nguyen and Safford
3	1-4	§ 103	Safford in light of Cepulis

On October 17, 2023, reexamination was ordered for all claims. *Id.*, 876-906. On February 23, 2024, the Examiner issued a non-final office action rejecting a subset of the claims based on the ground below. *Id.*, 914-980. On August 27, 2024, the Office issued a reexamination certificate confirming patentability of claims 1, 5-8, 13, 17, 21-22, and 25-26. *Id.*, 1131-1132.

Ground	Claims	Basis	Reference(s)
1	1, 5-8, 13, 17	§ 103	Kalyanasundharam in view of Kennedy

Unified's EPR and the Examiner's subsequent office action present materially different grounds from those presented in this Petition because they do not rely on Fox, Landry, or Arai—critical references disclosing the "changing an identifier" and "switch[ing] the role." Because this Petition proves that the allowable claim features are obvious, the grounds here are new and not cumulative. *See* Sections VII-X. While Safford, a secondary reference here, is also in Unified's reexamination, Safford-Cepulis does not disclose an operating system or firmware. Kennedy in view of Nguyen and Safford is deficient because Kennedy's disclosed boot processor replacement does not occur during system runtime, unlike Fox and Arai, and modifying Kennedy to have a spare processor during system runtime is unsupported.

See Ex. 1032, 91-93. Finally, the ground applied in the office action during reexamination do not include any references present in this Petition. Ex. 1031, 918.

XIII. 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 '302 Patent is involved in the following:

Name	Number	Court	Filed
Foras Technologies Ltd. v. Bayerische Motoren Werke AG	1:24-cv-00363	E.D. Va.	Mar. 6, 2024
Foras Technologies Ltd. v. Toyota Motor North America, Inc.	2:23-cv-00321	E.D. Tex.	July 12, 2023
Foras Technologies Limited v. Nissan Motor Company, Ltd. et al.	1:23-cv-00640	W.D. Tex.	June 6, 2023
Foras Technologies Limited v. Toyota Motor North America, Inc. et al.	2:23-cv-00150	E.D. Tex.	Apr. 5, 2023
Ex Parte Reexamination filed by Unified Patents	Control No. 90/019,243	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.

XIV. Standing

The ’302 Patent is eligible for *inter partes* review and Petitioner is not barred or estopped.

XV. Conclusion

Petitioner requests the Board institute review and cancel all 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 13,591 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 Attorney 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. 7,624,302.

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