

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORPORATION,
Petitioner,

v.

BiTMICRO LLC,
Patent Owner.

IPR2023-00785
Patent 9,858,084 B2

Before KEN B. BARRETT, TERRENCE W. McMILLIN, and
KEVIN C. TROCK, *Administrative Patent Judges*.

McMILLIN, *Administrative Patent Judge*.

DECISION
Granting Institution of *Inter Partes* Review
35 U.S.C. § 314

I. INTRODUCTION

A. Background and Summary

Intel Corporation (“Petitioner”)¹ filed a Petition to institute an *inter partes* review of claims 1, 5, 8, 9, 19, and 20 (the “challenged claims”) of U.S. Patent 9,858,084 B2 (Ex. 1001,² the “’084 patent”) pursuant to 35 U.S.C. § 311 *et seq.* Paper 1 (“Petition” or “Pet.”). BiTMICRO, LLC (“Patent Owner”)³ filed a Preliminary Response. Paper 6 (“Preliminary Response” or “Prelim. Resp.”).

We have authority under 35 U.S.C. § 314, which provides that an *inter partes* review may not be instituted unless the information presented in the Petition and the Preliminary Response shows that “there is a reasonable likelihood that the petitioner would prevail with respect to at least 1 of the claims challenged in the petition.” 35 U.S.C. § 314(a) (2018). For the reasons discussed below, we institute *inter partes* review on the grounds presented in the Petition.

B. Related Proceedings

The parties identify the following related proceedings: *BiTMICRO LLC v. Intel Corp.*, 5:23-cv-00625 (N.D. Cal.). Pet. 61; Paper 4, 2.

¹ Petitioner identifies Intel Corporation as the real party-in-interest. Pet. 61.

² Exhibit 1001 is incomplete. *See* Prelim. Resp. 2 n.2 (“[T]he FIG. 2 included in Ex. 1001 is not the complete drawing, as certain components including CPUs 201, the optional DMA controllers 203, and reset controller 206 in Ex. 1001’s FIG. 2 appear to be annotated out of the drawing.”). A complete Figure 2 of the ’084 patent is shown below in Section I.C. A complete copy of the ’084 patent has been placed into the record of this proceeding by the Board as Exhibit 3001.

³ Patent Owner identifies BitMicro, LLC as the real party-in-interest. Paper 4, 2.

Petitioner further identifies IPR2023-00787 challenging U.S. Patent No. 10,120,694, which is related to the '084 patent. Pet. 61.

C. The '084 patent

The '084 patent is titled “Copying of Power-On Reset Sequencer Descriptor from Nonvolatile Memory to Random Access Memory.” Ex. 1001, code (54). The '084 patent relates to a “mechanism of booting up a system directly from a storage device and a means of initializing an embedded system prior to activating a CPU,” with a “power-on reset (POR) sequencer” that “controls the overall boot process of the embedded system” using “descriptors (POR Sequence descriptors) which are used to update the configuration registers of the system and to enable CPU-independent data transfers with the use of DMA controllers.” *Id.*, code (57).

The '084 patent “makes use of a minimum amount of non-volatile memory in order to boot an embedded system.” *Id.* at 2:12–14. The '084 patent also provides a “[r]eduction in size of the nonvolatile memory [that] brings about an advantage in terms of cost and complexity,” and “also allows the system to verify the integrity of data and code used in the boot process.” *Id.* at 2:14–17.

Figure 2, reproduced below, illustrates “an embedded system with a power-on reset sequencer that manages the initialization and boot up of the system” in accordance to the '084 patent. *Id.* at 4:22–24.

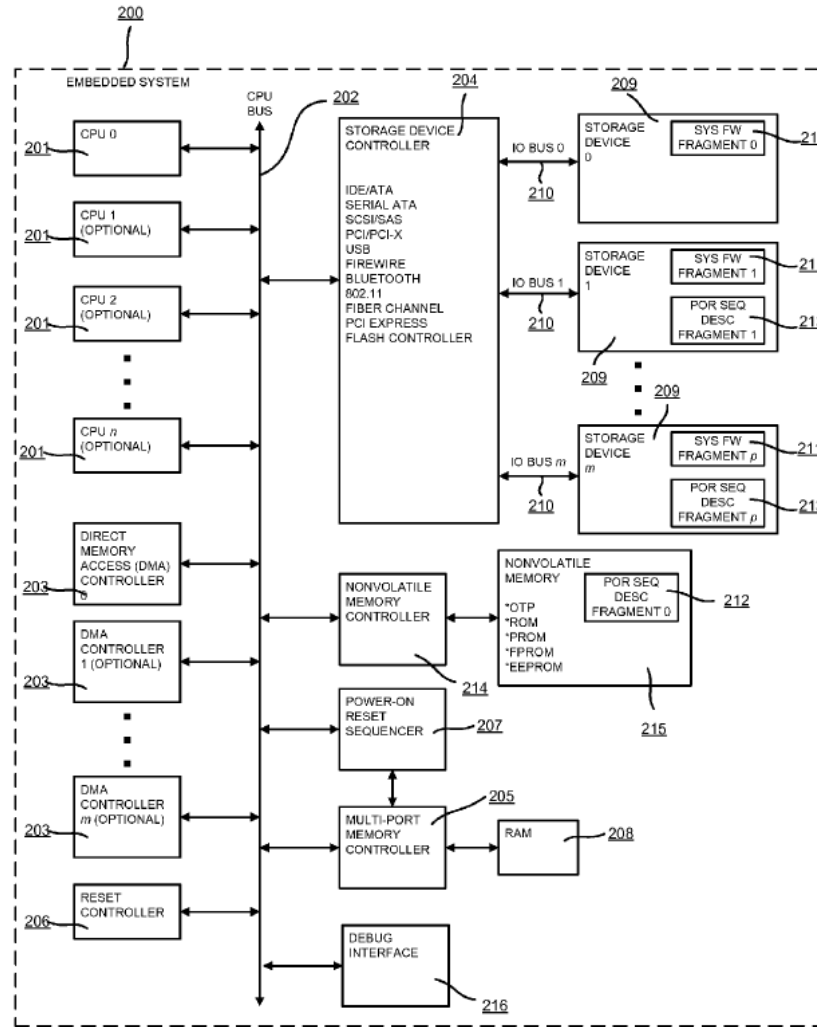


FIG. 2

Figure 2 depicts “an embedded system 200” with one or more CPUs 201, one or more DMA controllers 203, RAM controller 205, and a power-on reset (POR) sequencer 207. *Id.* at 4:49–60. “The POR Sequencer makes use of a POR Sequencer Descriptor which is a preassembled descriptor that is stored in a nonvolatile memory of the system” and which “contains register information and direct memory access (DMA) controller descriptors” that are “read and used by the POR Sequencer to update the configuration registers of the system.” *Id.* at 2:48–56. A POR sequencer descriptor can be “divided into fragments 212, 213 and distributed across the nonvolatile memory 215 and storage devices 209.” *Id.* at 5:8–11; *see id.* at 6:34–37.

Figure 5, reproduced below, illustrates “a linked-list data structure of the power-on reset sequencer descriptor.” *Id.* at 4:30–31.

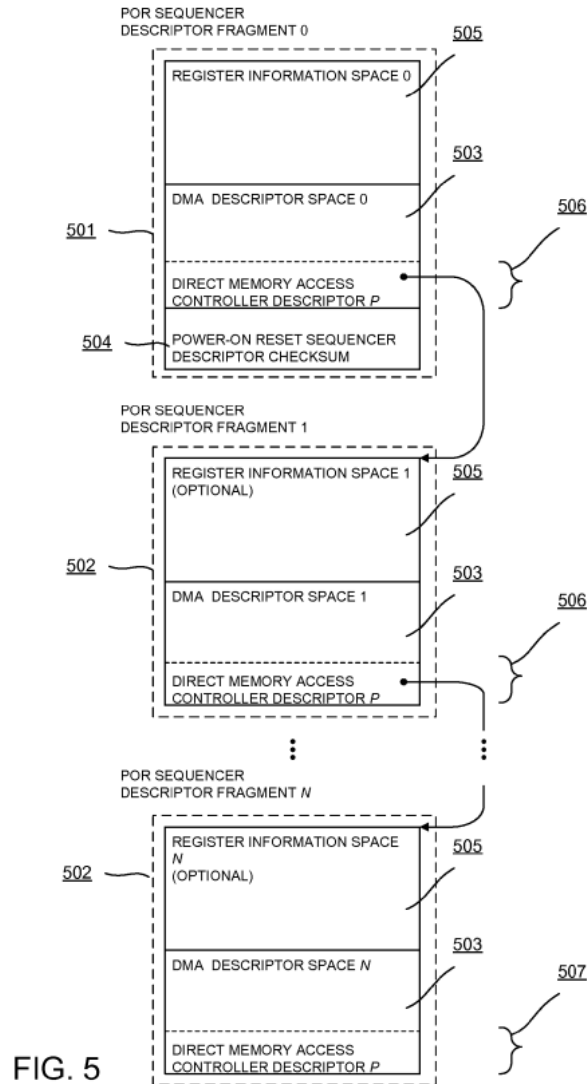


Figure 5 depicts “a fragmented POR sequencer descriptor.” *Id.* at 6:37–39.

To “transfer[] the POR sequencer descriptors 212, 213 from nonvolatile memory 215 and storage device 209 to the RAM 208,” “[a]fter the system is powdered-up and the input power supply stabilizes,” then the POR sequencer 207 is “brought out of reset.” *Id.* at 6:56–63. “The POR sequencer 207 then detects the presence of POR sequence descriptor 212 in

the nonvolatile memory” and the POR sequencer descriptor 212 is copied from the nonvolatile memory 215 to the RAM 208, including checking for integrity and handling errors. *Id.* at 6:64–7:9. The POR sequencer then processes the POR sequencer descriptor fragment by “process[ing] the register information and then updat[ing] the corresponding configuration register based on the contents of the register information.” *Id.* at 7:23–26.

D. Challenged Claims

Petitioner challenges claims 1, 5, 8, 9, 19, and 20 of the ’084 patent. Pet. 1. Of the challenged claims, claims 1 and 19 are independent claims. Ex. 1001, 9:14–33, 10:49–65. Claim 1 recites:

1. A computer-implemented method, comprising:
releasing components of an embedded system from reset;
detecting a power-on reset (POR) sequencer descriptor in a nonvolatile memory;
wherein ***the POR sequencer descriptor comprises*** information to initialize configuration registers of the embedded system, ***Direct Memory Access (DMA) descriptors used to fetch other POR sequencer descriptor fragments***, and a system firmware;
copying the POR sequencer descriptor from the nonvolatile memory to a random access memory (RAM);
verifying an integrity of the POR sequencer descriptor; and
detecting any error in the POR sequencer descriptor; and
wherein the computer-implemented method allows the embedded system to verify an integrity of user data and program code used in a boot process in order to provide a resilient boot up sequence and wherein the computer-implemented method reduces an amount of the nonvolatile memory used in the boot process.

Id. at 9:14–33 (emphasis added to disputed limitations).

E. The Asserted Ground

Petitioner challenges claims 1, 5, 8, 9, 19, and 20 of the '084 patent based on the ground set forth in the table below.

Claims Challenged	35 U.S.C. §	Reference(s)/Basis
1, 5, 8, 9, 19, 20	103	Post ⁴ , Wilson ⁵ , Kao ⁶

Pet. 6. Petitioner supports its showing of unpatentability of the challenged claims of the '084 patent with Exhibits 1001–1021 including the Declaration of Donald Alpert, Ph.D. (Ex. 1003). Patent Owner did not file any exhibits in support of the Preliminary Response. As such, there is no testimony in the record to contradict that provided in the Declaration of Dr. Alpert.

II. ANALYSIS

A. Legal Standards

A patent claim is unpatentable as obvious if the differences between the claimed subject matter and the prior art are such that the subject matter, as a whole, would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 406 (2007). The question of obviousness is resolved on the basis of underlying factual determinations including: (1) the scope and content of the prior art; (2) any differences between the claimed subject matter and the prior art; (3) the level of ordinary skill in the art; and (4) objective evidence of non-obviousness. *Graham v. John Deere Co.*, 383 U.S. 1, 17–18 (1966).

“In an [*inter partes* review], the petitioner has the burden from the onset to show with particularity why the patent it challenges is

⁴ US 8,799,555 B2, issued Aug. 5, 2014 (Ex. 1005).

⁵ US 7,861,073 B2, issued Dec. 28, 2010 (Ex. 1013).

⁶ US 7,490,177 B2, issued Feb. 10, 2009 (Ex. 1004).

unpatentable.” *Harmonic Inc. v. Avid Technology, Inc.*, 815 F.3d 1358, 1363 (Fed. Cir. 2016) (citing 35 U.S.C. § 312(a)(3) (requiring *inter partes* review petitions to identify “with particularity . . . the evidence that supports the grounds for the challenge to each claim”)). Petitioner cannot satisfy its burden of proving obviousness by employing “mere conclusory statements.” *In re Magnum Oil Tools Int’l, Ltd.*, 829 F.3d 1364, 1380 (Fed. Cir. 2016).

B. Level of Ordinary Skill in the Art

With regard to the level of ordinary skill in the art, Petitioner contends a person of ordinary skill in the art “would possess a Bachelor of Science degree in electrical engineering or computer science, and at least two years of experience in the field of embedded system design, including booting of such systems, or an equivalent combination of education, work and/or experience.” Pet. 17 (citing Ex. 1003 ¶ 20). Patent Owner does not address the level of ordinary skill in the art. *See generally* Prelim. Resp.

For purposes of this Decision, we adopt Petitioner’s definition of the level of ordinary skill in the art because it is consistent with the asserted prior art.⁷ *See Okajima v. Bourdeau*, 261 F.3d 1350, 1355 (Fed. Cir. 2001) (recognizing that the prior art itself may reflect an appropriate level of skill in the art).

C. Claim Construction

We apply the same claim construction standard used in district court actions under 35 U.S.C. § 282(b) as articulated in *Phillips v. AWH Corp.*, 415 F.3d 1303 (Fed. Cir. 2005) (en banc). *See* 37 C.F.R. § 42.100(b).

⁷ No matter how designated in this Decision, any determination (except our decision to institute trial) is preliminary and non-binding. We wish to have the full record as developed during trial before rendering any binding determination, finding, or conclusion.

In applying that standard, claim terms generally are given their ordinary and customary meaning as would have been understood by a person of ordinary skill in the art at the time of the invention and in the context of the entire patent disclosure. *Phillips*, 415 F.3d at 1312–13. “In determining the meaning of the disputed claim limitation, we look principally to the intrinsic evidence of record, examining the claim language itself, the written description, and the prosecution history, if in evidence.” *DePuy Spine, Inc. v. Medtronic Sofamor Danek, Inc.*, 469 F.3d 1005, 1014 (Fed. Cir. 2006) (citing *Phillips*, 415 F.3d at 1312–17). There is a “heavy presumption,” however, that a claim term carries its ordinary and customary meaning. *CCS Fitness, Inc. v. Brunswick Corp.*, 288 F.3d 1359, 1366 (Fed. Cir. 2002) (citation omitted). We shall construe only terms that are in controversy and then only to the extent necessary to resolve the controversy. *Vivid Techs., Inc. v. Am. Sci. & Eng’g, Inc.*, 200 F.3d 795, 803 (Fed. Cir. 1999).

Petitioner submits that none of the “term[s] in the challenged claims requires an express construction.” Pet. 17. Petitioner states that, “[t]he Petition thus applies the plain and ordinary meaning to the language of the challenged claims.” *Id.*

Patent Owner argues that “Petitioner’s proposed characterization of the claimed ‘Direct Memory Access (DMA) descriptors’ as satisfied by nothing more than a pointer in Post (which does not disclose a DMA controller or a DMA descriptor) is incorrect and does not comport with the plain and ordinary meaning of this claim term.” Prelim. Resp. 11. Patent Owner argues that Petitioner’s “incorrect characterization fails to consider the ’084 Patent’s specification disclosure that the ‘Direct Memory Access (DMA) descriptors’ allow for activation of associated DMA controllers, and

is inconsistent with Petitioner’s own evidence.” *Id.* Patent Owner argues the Petitioner’s approach does not consider the claim language, that the “POR sequencer descriptor comprises . . . Direct Memory Access (DMA) descriptors used to fetch other POR sequencer descriptor fragments.” *Id.* at 11–12.

Patent Owner further “submits that the Board need not engage in a formal construction of ‘Direct Memory Access (DMA) descriptors’ in this pre-institution phase.” *Id.* at 12. Patent Owner argues that “[a]ll evidence of record confirms that a descriptor is more than just Post’s pointer, and Petitioner’s decision to disregard the plain meaning of this claim term is fatal to the Petition.” *Id.*

At bottom, neither party argues that any express claim construction is necessary in order to determine whether to institute trial. *See* Pet. 17, Prelim. Resp. 11–12. We agree. To the extent necessary, we will consider Patent Owner’s argument regarding “Direct Memory Access (DMA) descriptors” as recited in challenged independent claim 1 (Ex. 1001, 9:20–21) in our analysis of the challenge to this claim.

D. *Summaries of Cited Art*

1. *Post (Ex. 1005)*

Post is titled “Boot Data Storage Schedules for Electronic Devices.” Ex. 1005, code (54). Post discloses systems and methods “for storing and retrieving boot data (e.g., a first stage bootloader) in and from a nonvolatile memory (‘NVM’), such as a NAND flash memory.” *Id.*, Abstract. Post describes that to “prevent a boot ROM [Read Only Memory] from having to maintain a NVM-specific map of which pages are used to store boot data, the map may be maintained in the NVM itself,” such as “in the form of a

linked list, where each page storing boot data can include a pointer that points to the next page that stores boot data.” *Id.*

Post’s Figure 1 is reproduced below.

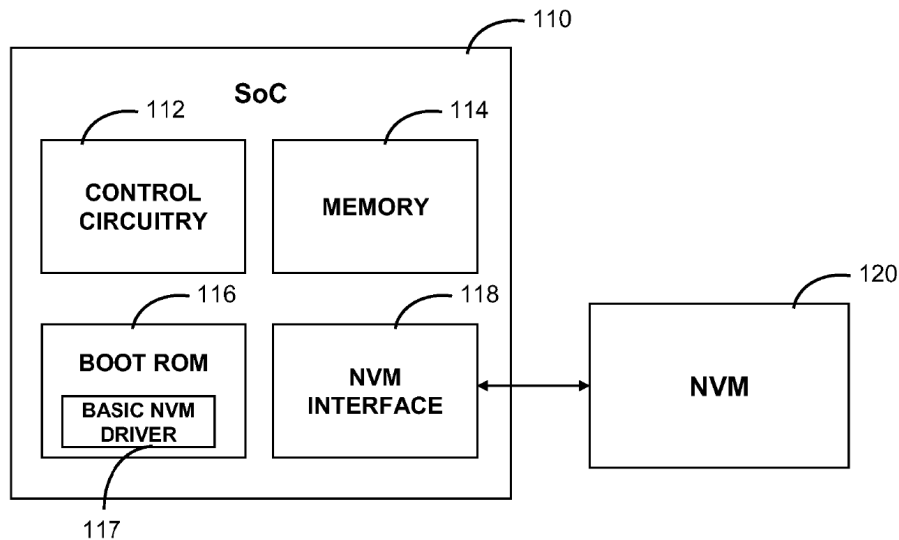


Figure 1 is a block diagram of an electronic device with non-volatile memory. Ex. 1005, 2:36–38. Figure 1 shows electronic device 100, which may be a portable media player, cellular telephone, pocket-sized personal computer, personal digital assistant, desktop or laptop computer, or other device. *Id.* at 2:60–67. The device includes system-on-a-chip 110, control circuitry 112, memory 114, boot ROM 116, NVM interface 118, and non-volatile memory 120. *Id.* at 3:33–36. Boot data may be stored in a subset of memory locations in the boot data storage area of NVM 120. *Id.* at 4:42–47. The subset of pages selected to store boot data may be nonconsecutive. *Id.* at 5:39–40. According to Post, “the loading of boot data can be accomplished by incrementally processing and validating pages as the pages are being read.” *Id.* at 5:50–52.

Post describes that each block of data in NVM can store both boot data and metadata, where the metadata can include a “next page pointer” and

“prior page pointer, and “boot data page indicator.” *Id.* at 6:28–37. Post discloses that:

the electronic device can use the next page pointers to verify the integrity of boot data storage. For example, since next page pointer 302 points to page 2, page 2 is expected to contain boot data. The electronic device may confirm that page 2 provides boot data by, for instance, employing an error correcting code on the contents of page 2.

Id. at 6:54–59.

Post issued from an application filed on April 14, 2011. *Id.*, code (22). The earliest priority date claimed for the '084 patent is March 15, 2013. Ex. 1001, code (60), 1:9–10. Post is prior art to the '084 patent.

2. *Wilson (Ex. 1013)*

Wilson is titled “On-Chip Decompression Engine.” Ex. 1013, code (54). Wilson states that a “microcontroller is often referred to as ‘a computer on a chip.’” *Id.* at 1:13–14. Wilson explains that “[m]any microcontrollers require a boot-up process when initially powered up,” which “allows a microcontroller to perform various actions to prepare the microcontroller for ordinary operation, such as load initialization values in its registers, load instructions in its instruction memory, etc.” *Id.* at 1:24–28. Wilson further explains that “removing errors in a microcontroller’s boot operation becomes difficult if all boot data of a microcontroller is stored internally in a ROM,” but “storing the boot data outside of the microcontroller poses other difficulties.” *Id.* at 1:40–45.

To address this, Wilson discloses a “microcontroller which includes a Register Load Assist (RLA) engine.” *Id.* at 1:55–57. In this way, “most of the boot data can be stored at a non-volatile memory external to the microcontroller.” *Id.* at 1:58–60. “The RLA engine can then decompress

the boot data and store various portions of it in their final destinations (such as, for example, in respective registers).” *Id.* at 2:1–3. As a result, “by allowing the remote receipt of compressed boot data, the RLA engine provides for a microcontroller which can receive a large amount of boot data from an external source, but nevertheless feature a relatively fast boot-up sequence.” *Id.* at 2:4–8. Figure 1 of Wilson is reproduced below.

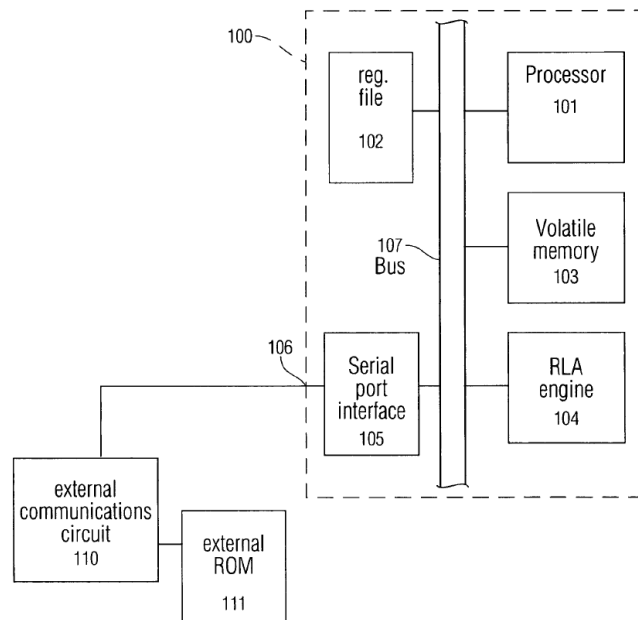


Figure 1 is a diagram of Wilson’s microcontroller and related components. *Id.* at 2:12–14. Figure 1 shows microcontroller 100, programmable processor 101, register file 102, volatile memory 103, RLA engine 104, tied together with bus 107, as well as external ROM 111, which may store boot and other data, and which can send the boot data to the microcontroller through port 106 and serial port interface 105. *Id.* at 2:51–3:32.

Wilson issued on December 28, 2010. *Id.*, code (45). The earliest priority date claimed for the ’084 patent is March 15, 2013. Ex. 1001, code (60), 1:9–10. Wilson is prior art to the ’084 patent.

3. *Kao (Ex. 1004)*

Kao is titled “System Method for Performing a Direct Memory Access for Automatically Copying Initialization Boot Code in a New Memory Architecture.” Ex. 1004, code (54). Kao discloses that devices “such as cell phones, PDAs, portable music players, appliances, and so on typically incorporate an embedded computer system.” *Id.* at 1:15–17. Kao describes when an “embedded system is powered up, the embedded system typically performs a boot sequence to properly load the operating code from the nonvolatile memory to the volatile memory and initialize the host with the operating code.” *Id.* at 1:43–46. “In order to perform the boot sequence, the host typically accesses boot code which is stored in a predefined area of non-volatile memory.” *Id.* at 1:46–49. Kao further explains:

In some cases, the host may transfer the operating code from the nonvolatile memory to the volatile memory using direct memory access (DMA). DMA allows a portion of memory to be quickly and automatically moved from one storage location to another storage location. To perform the DMA transfer, the host may need to initialize the DMA engine. In some cases, the DMA engine may be located in the host. To initialize the DMA engine, the host may use information and/or instructions provided by the boot code. Once the DMA engine is initialized, the host may issue commands to the DMA engine to load operating code from the non-volatile memory into the volatile memory. The host may then begin executing the operating code and any other code necessary for proper operation of the host.

Id. at 1:54–67.

Kao discloses that:

By utilizing the DMA engine located on the volatile memory chip to load boot and controller codes, the processing workload necessary to boot the embedded system may be offloaded from a host processor to the volatile memory chip, thereby reducing

the complexity (and therefore the overall cost) of the host processor and the embedded system.

Id. at 3:3–9. Then, the “host may perform boot operations,” which “may include issuing commands to the DMA engine to load operating system code in the volatile memory.” *Id.* at 3:13–17.

Kao issued on February 10, 2009. *Id.*, code (45). The earliest priority date claimed for the ’084 patent is March 15, 2013. Ex. 1001, code (60), 1:9–10. Kao is prior art to the ’084 patent.

E. Consideration of Alleged Obviousness Over Post, Wilson, and Kao

Petitioner asserts that “the combination of Post, Kao, and Wilson disclosed all limitations of the challenged claims.” Pet. 25. The Petition states that:

In the resulting combination, Post’s control circuitry 112 would be modified to include circuitry that performed the register loading functionality of Wilson’s RLA engine as well as circuitry to perform the boot data transfer functionality of Kao’s DMA engine; and Post’s boot data included initial register values, as taught by Wilson.

Id. Patent Owner argues that the Petition should be denied because: “Petitioner’s alleged reason for combining Post and Wilson is inconsistent with the resulting combination of these references” (Prelim. Resp. 8); “Petitioner’s proposed combination fails to disclose or suggest any DMA descriptors as part of the POR sequencer descriptor” (*id.* at 10); and “Post does not disclose any verification of the integrity of a POR sequencer descriptor” (*id.*). We begin our analysis of the contentions of the parties with a consideration of whether claim 1 would have been obvious over Post, Wilson, and Kao. *See* Pet. 25–56.

*1. Element-by-Element Analysis of Claim 1*⁸

a) A computer-implemented method, comprising

Petitioner does not take a position on whether the preamble of claim 1 is limiting. Pet. 26 (“Intel does not concede that the preamble is limiting.”). Petitioner relies on Post as disclosing “a computer-implemented method.” *Id.* Patent Owner does not discuss the preamble of claim 1 in the Preliminary Response. *See generally* Prelim. Resp.

As set forth in the Petition (Pet. 26), Post discloses an “electronic device 100 [that] can be . . . a pocket-sized personal computer, a personal digital assistance (‘PDA’), a desktop computer, a laptop computer, and any other suitable type of electronic device or system.” Ex. 1005, 2:64–67.

Petitioner cites Figures 4–6 of Post as disclosing processes or methods. Pet. 26. Post states:

FIG. 4 is a flowchart of an illustrative process for initiating bootup of an electronic device in accordance with various embodiments of the invention;

FIG. 5 is a flowchart of an illustrative process for reading and processing boot data from a non-volatile memory in accordance with various embodiments of the invention; and

FIG. 6 is a flowchart of an illustrative process for storing boot data in a non-volatile memory in accordance with various embodiments of the invention.

Ex. 1005, 2:48–56. We determine that Post teaches a “computer-implemented method.” As there is no dispute between the parties as to the preamble and Post teaches a “computer-implemented method,” we need not decide whether the preamble is limiting.

⁸ We adopt the Petitioner’s parsing of claim 1 in order to follow the presentation in the Petition. *See* Pet. 26–56.

b) releasing components of an embedded system from reset

Petitioner relies on Post for this limitation. Pet. 26–30. Patent Owner does not dispute Petitioner’s showing as to this limitation. *See generally* Prelim. Resp.

With regard to the “*embedded system*” element of this limitation, the Petition states:

Post’s system-on-chip (“SOC”) 110 and NVM 120 were collectively *an embedded system*. (INTEL-1003, ¶ 71.) The ’084 patent describes an “embedded system” as including a system with one or more processors. (INTEL-1001, 1:28-30 (“embedded system will be used to refer to a single CPU or a multiprocessor system in this paper”).) Post’s electronic device 100 “include[d] system-on-a-chip (‘SoC’) 110 and non-volatile memory (‘NVM’) 120.” (INTEL-1005, 3:1-2.) . . . Because Post’s SoC 110 and NVM 120 had at least one processor (control circuitry 112, implemented as “one or more processors”), a POSITA^[9] would have understood SOC 110 and NVM 120 to be an example of an “embedded system” as the ’084 patent used that term. (INTEL-1003, ¶¶ 71-72.)

Pet. 26–27 (alteration in original). We determine Post teaches an embedded system.

With regard to “*releasing components of an embedded system from reset*,” the Petition states:

When a system is powered on, components of the system are initially held in a reset state before they can be booted up. (INTEL-1004, 6:65-67 (“When the system 100 is initially powered up, each of the system components may enter a reset state 402.”); INTEL-1018, 2 (“When power is applied to a microprocessor, the internal registers ‘come up’ in arbitrary states that contain random data. Applying a reset command to

⁹ POSITA is an acronym commonly used in patent matters to refer to person of ordinary skill in the art.

the microprocessor ... reset[s] all internal circuits at a predictable starting point.”); INTEL-1003, ¶ 78.) For example, the '084 patent describes a “process of releasing ... CPUs 201 from their reset state” where each CPU is released from reset before executing system firmware. (INTEL-1001, 8:25-40, 3:62-67.)

Post’s control circuitry 112 and NVM interface 118 were components of SoC 110 (“*components of [the] embedded system*”). (INTEL-1001, 3:33-34.) Post disclosed that after the device including SoC 110 was powered on, control circuitry 112 was booted by executing program code from boot ROM: “at bootup of electronic device 100, control circuitry 112 may operate under the control of program code stored in boot ROM 116.” (*Id.*, 3:41-43.) Post also disclosed that NVM interface 118 helped control circuitry 112 load additional boot data from NVM 120. (*Id.*, 3:57-4:5 (under control of the program code, control circuitry 112 “beg[a]n loading additional information from NVM 120 into memory 114,” including, “with the aid of NVM interface 118 ... boot data from NVM 120”).) A POSITA would have understood Post’s disclosure of control circuitry 112 and NVM interface 118 performing the function of copying data from NVM 120 after “bootup of electronic device 100” to indicate that these components had been released from reset. (INTEL-1018, 2; INTEL-1003, ¶¶ 79-80.). . . .

Further, Post taught that process 400 for “initiating bootup” (INTEL-1005, 8:15-16 . . . , 8:25-31.) A POSITA would have understood this to further indicate that the control circuitry and NVM interface—the components that performed the process of transferring boot data from nonvolatile memory (*id.*, 3:52-4:5, 8:9-12)—had been released from reset before performing the above operations. (INTEL-1003, ¶¶ 81.)

Pet. 28–30 (alterations in original). We determine that Post teaches this limitation.

- c) *detecting a power-on reset (POR) sequencer descriptor in a nonvolatile memory; wherein the POR sequencer descriptor comprises information to initialize configuration registers of the embedded system, Direct*

Memory Access (DMA) descriptors used to fetch other POR sequencer descriptor fragments, and a system firmware

Petitioner relies on a combination of the teachings of Post, Wilson, and Kao for this limitation. Pet. 30–48. Petitioner relies on Post as teaching a “*power-on reset (POR) sequencer descriptor*.” *Id.* at 30–32. The Petition states:

“When an electronic device is being powered up or rebooted, an operating system can be booted up from a NVM of the electronic device.” (INTEL-1005, 1:16-19.) Post described this boot sequence as facilitated by boot data and associated pointer metadata stored in NVM 120. At system bootup, control circuitry 112 “beg[a]n loading additional information from NVM 120 into memory 114” that included “firmware, such as a first stage bootloader.” (*Id.*, 3:55-59.) Post referred to this “firmware [] or other data” as “boot data.” (*Id.*, 3:63-65.) The boot data was stored in a linked-list format in pages of NVM 120, along with metadata pointers that provided the address of next and prior boot data pages and were used to identify additional boot data pages to retrieve. (*Id.*, 6:19-53, 9:47-62; *id.*, 6:22-24, 8:9-12.) Control circuitry 112 used the boot data to “load a second stage bootloader or the kernel and operating system of electronic device 100.” (*Id.*, 3:59-63.) Once the operating system was loaded, “control of processor 112 [w]as transferred to the operating system.” (*Id.*, 4:5-7; *id.*, 1:19-26.) Post taught that “successful completion of this bootup procedure is typically critical for the electronic device to function properly.” (*Id.*, 1:26-28, 1:34-37.)

Post’s stored boot data and associated metadata were thus a “*POR [power-on reset] sequencer descriptor*” that defined Post’s boot up sequence and enabled the retrieval of other information in the same way that the ’084 patent’s POR sequencer descriptor was described as being processed and used to control the boot-up sequence. (INTEL-1003, ¶¶ 83-84.)

Id. at 30–31 (alterations in original).

With regard to “*detecting a power-on reset (POR) sequencer descriptor in a nonvolatile memory*,” Petitioner relies on “Post disclos[ing] that its boot data and associated metadata [a]re stored in nonvolatile memory 120” and “Post also disclos[ing] searching for and identifying boot data (*‘detecting a POR sequencer descriptor’*) in the nonvolatile memory.” Pet. 47 (citing Ex. 1005, 3:52–65, 4:27–29, 5:9–23, 6:22–24, 7:34–50, 8:7–9:2, 8:34–35). Petitioner’s contentions with regard to “*detecting a power-on reset (POR) sequencer descriptor in a nonvolatile memory*” are supported by the cited passages in Post and we determine that Post teaches these elements of this limitation.

With regard to the first portion of the “wherein” clause of this limitation (“*wherein the POR sequencer descriptor comprises information to initialize configuration registers of the embedded system*”), Petitioner relies on a combination of the teachings of Post and Wilson. Pet. 25 (“In the resulting combination, Post’s control circuitry 112 would be modified to include circuitry that performed the register loading functionality of Wilson’s RLA [Register Load Assist] engine . . .”), 37 (“combine Post with Wilson’s teaching of including initial values for configuration registers in boot data. In the resulting combination, Post’s boot data stored in NVM 120 included information to initialize configuration registers of SoC 110 (*e.g.*, configuration registers of control circuitry 112).”).

With regard to Wilson providing teachings of “*information to initialize configuration registers of the embedded system*,” the Petition states:

Wilson taught that many microcontrollers required a boot-up process that allowed the microcontroller to “perform various actions to prepare [it] for ordinary operation, such as

load initialization values in its registers.” (*Id.*, 1:24-28.) The microcontroller’s boot data, stored in external nonvolatile memory ROM 111, thus included “predefined initial values” to be loaded into the microcontroller’s registers during the boot process: “The microcontroller can require that certain specific predefined initial values are stored in one or more of the registers of the register file before the microcontroller assumes ordinary operation [T]hese *initial* values can be a part of boot data that is loaded in the microcontroller during a boot-up procedure.” (*Id.*, 2:53-63; *id.*, 5:1-4 (“The predefined [register] values can be sent from an element external to the microcontroller ... to the microcontroller at time of boot up.”); *id.*, 5:30-45 (boot data obtained from external nonvolatile ROM 111), 3:26-34.)

A POSITA would have understood that these “initial values” stored in registers during a boot process were “information to initialize configuration registers of the embedded system”—for example, information to initialize the registers that configured display colors of the mobile phone touchscreen (“*configuration registers of the embedded system*”). (INTEL-1003, ¶ 89.) These initial register values were included in boot data stored in external nonvolatile ROM and copied to volatile memory on the microcontroller. (INTEL-1013, 5:57-61 (microprocessor receives block of boot data 400 and stores boot data portion in volatile memory 103), 5:46-51 (“The boot data can be sent in a single block. ...”), 6:42-44 (boot data includes initial register values, optionally with other data/code), 1:61-67 (boot data sent from external nonvolatile memory to internal microcontroller volatile memory).).

Pet. 33–34 (alterations in first paragraph in original). Petitioner’s contentions with regard to “*information to initialize configuration registers of the embedded system*” are supported by the cited passages in Wilson and we determine that Wilson teaches these elements of this limitation.

With regard to combining the relevant teachings of Post and Wilson, Petitioner contends, “[a] POSITA would have found it obvious to modify

Post in view of Wilson’s teachings, such that Post’s boot data in NVM 120 included information to initialize configuration registers of control circuitry 112 (including its processors).” Pet. 34. The Petition states:

As Post is silent about techniques for loading initialization values into its registers during boot, a POSITA would have looked to Wilson for implementation details for a register initialization technique. (INTEL-1003, ¶¶ 92-93.)

* * *

A POSITA would have been motivated to similarly custom design Post’s SoC 110, including control circuitry 112, to realize the benefits of performing specific predefined functions like configuring screen color and touchscreen functionality, which required initialization of configuration registers, as taught by Wilson. (INTEL-1003, ¶ 94.)

Further, a POSITA would have viewed Wilson as providing a teaching, suggestion, or motivation to include initial register values in Post’s boot data because Wilson taught that register values *must* be loaded before a microcontroller can begin its normal operation. (INTEL-1013, 4:59-61 (“for microcontroller 100 to commence ordinary operation, it can be required that certain predefined values be loaded in the registers of register file 102”).) Wilson also taught that including initial register information along with other boot data, such as initial memory values and instructions, could improve the speed of the boot process, which a POSITA would have viewed as a further teaching, suggestion, or motivation for this combination.

* * *

[A] POSITA would have found this combination obvious because Wilson’s teaching of including initial register values in boot data was a known technique for solving the known problem of how to initialize processor register information at bootup (INTEL-1013, 1:25-28, 2:54-63; INTEL-1016, 555).

Id. at 34–36.

Patent Owner argues:

Summarizing Petitioner’s theory, in order for Post’s control circuitry 112 (including processors) to begin normal operations—such as detecting and copying a POR sequencer descriptor from nonvolatile memory to RAM—control circuitry 112’s registers values must first be loaded and initialized for Post’s processors to operate normally. But Petitioner asks Post’s control circuitry 112 to perform the operations of loading boot code from NVM 120 to memory 114 before its own registers are loaded or initialized. Under these circumstances, according to Petitioner, Post’s control circuitry 112 cannot begin normal operations to pre-load its own initialized register information. Indeed, Petitioner offers no explanation how Post’s control circuitry 112 can begin the operation of loading boot code into memory 114 without the requisite initialized register values for control circuitry 114 [*sic*] first being loaded. This bootstrapping theory for how control circuitry 112 can prepare itself for its own normal operation is the result of a hindsight-based recreation of the challenged claim language, and fails as illogical.

Prelim. Resp. 15. Patent Owner’s argument appears to be based on a misunderstanding of Petitioner’s contentions. As shown above, in Petitioner’s combination, the POR sequence descriptor would include “*information to initialize configuration registers of the embedded system*” as taught by Wilson in Post’s boot data. *See, e.g.*, Pet. 37 (“In the resulting combination, Post’s boot data stored in NVM 120 included information to initialize configuration registers of SoC 110 (*e.g.*, configuration registers of control circuitry 112).”). The Petition states that, “[a]s Post is silent about techniques for loading initialization values into its registers during boot, a POSITA would have looked to Wilson for implementation details for a register initialization technique” (*id.* at 34–35 (citing Ex. 1003 ¶¶ 92–93)) and “Wilson . . . taught that initial values for those registers would be

included as part of the boot data” (*id.* at 35 (citing Ex. 1013, 6:49-7:3.)). And, the Petition points to explicit teachings in Wilson of including initialization of registers in the boot process. *Id.* at 33 (citing Ex. 1013, 2:56–63 (“The microcontroller can require that certain specific predefined initial values are stored in one or more of the registers of the register file before the microcontroller assumes ordinary operation. . . . [T]hese initial values can be a part of boot data that is loaded in the microcontroller during a boot-up procedure.”), 5:1–4 (“The predefined values can be sent from an element external to the microcontroller (such as, for example, external communication circuit 110) to the microcontroller at time of boot up.”)). As this argument does not accurately or persuasively address the showing in the Petition, we reject it. We determine that Petitioner has shown that a skilled artisan would have been motivated to combine the relevant teachings of Post and Wilson as asserted in the Petition.

With regard to “*wherein the POR sequencer descriptor comprises . . . Direct Memory Access (DMA) descriptors used to fetch other POR sequencer descriptor fragments,*” Petitioner relies on a combination of the teachings of Post and Kao. Pet. 37–46. Petitioner relies on Post for teaching “*descriptors used to fetch other POR sequencer descriptor fragments.*” *Id.* at 37–42. The Petition states:

Post disclosed storing boot data in fragments stored in different pages of nonvolatile memory. “[B]oot data may be stored in a subset of the memory locations in the boot data storage area” and “the subset of pages [of NVM] selected to store boot data may be nonconsecutive.” (INTEL-1005, 4:46-47, 5:39-40; *id.*, 5:19-23; *id.*, 1:55-64.) The boot data was stored in a “linked-list format,” where “each of the boot data pages can store both data . . . and metadata . . . includ[ing] a next page pointer 302 and a prior page pointer 304.” (*Id.*, 6:19-21, 28-31; *id.*, 6:22-

24.) These pointers “reference[d] the next boot page in the sequence” by “provid[ing] the address” of the next page of boot data. (*Id.*, 6:38-42.) Boot data pages could be in the same block or distributed across different blocks of different storage devices. (*Id.*, 6:48-53 (a “next page pointer can be an address referencing another page in the same block” or “another page in a different block of the same or a different integrated circuit”).)

Id. at 37–38 (alterations in original). Petitioner’s citations to Post support its showing that Post teaches “*descriptors used to fetch other POR sequencer descriptor fragments.*” We determine that Post teaches these elements.

Petitioner acknowledges that “[a]lthough Post taught that the control circuitry included ‘any suitable components, circuitry, or logic’ ([Ex. 1005,] 3:36-38), it did not specifically identify Direct Memory Access (DMA) components.” Pet. 42. For DMA components and related teachings, Petitioner relies on Kao. *Id.* at 42–43. The Petition states:

Kao taught techniques for “booting an embedded system” in which “boot codes and controller codes stored in a non-volatile memory may be loaded by a DMA engine.” (INTEL-1004, 2:65-3:3; *id.*, 9:10-13 (“when the PowerUp_Trigger signal is received, the DMA engine 120 may automatically load the boot and controller codes 132, 134”), Fig. 1 (showing boot codes 132, controller codes 134 in nonvolatile NAND flash memory).)

* * *

The DMA engine used data within the boot codes to load controller codes from the nonvolatile NAND flash memory into a DRAM device 110. (*Id.*, 10:28-35 (“the controller codes 134 may be loaded using data stored in the boot codes 132 ... by loading the data in the boot codes 132 directly into a register in the DMA engine 120”), 10:36-50 (describing process for obtaining source address and size of controller codes from data in boot codes), 5:25-32 (“The controller codes 134 may be loaded ... using instructions in the boot codes 132”).) The

DMA engine's transfer of data from nonvolatile memory occurred "without interaction with the host." (*Id.*, 3:19-21.)

Id.

With regard to combining the relevant teachings of Post and Kao, Petitioner contends that:

A POSITA would have found it obvious to modify Post in view of Kao's teachings, such that Post's controller circuitry 112 included a DMA engine that used Post's pointer metadata to load pages of boot data subsequent to the first identified boot data page from nonvolatile memory 120 into RAM 114. (INTEL-1003, ¶ 111.) A POSITA would understand that in this combination, the next-page pointer metadata—as used by a DMA engine in identifying a next page of boot data—would be a "DMA descriptor." (INTEL-1003, ¶ 111.)

Id. at 43. Petitioner contends that a skilled artisan would have been motivated to combine the relevant teachings of Post and Kao in this way, because:

Kao provided a teaching, suggestion, or motivation for making this combination because Kao teaches that using DMA to perform memory transfers offloads work from the processor, thus reducing the processor's workload in booting an embedded system and also reducing system complexity and cost.

* * *

A POSITA would have understood reducing the workload of the processor to be a "universal" motivation within the field of embedded systems, which further motivates the combination of Kao's teachings with Post.

* * *

Kao disclosed that its DMA techniques had been used to improve one device (an embedded system including one or more processors, INTEL-1004, 3:22-27), and a POSITA would have recognized that these techniques improved Post's similar device (an SoC including one or more processors and NVM

(*see* INTEL-1005, 3:33-38, 2:60-3:2; *supra* Sec. VI.B.2)) in the same way (by offloading processor workload and reducing processor complexity and cost).

* * *

[B]oth Post and Kao addressed the problem of transferring data from NVM during a boot of an embedded system, and Kao's DMA engine and DMA techniques were a known way to address that problem.

* * *

[A] POSITA would have viewed this combination as involving the simple substitution of one known element (DMA techniques and circuitry, as taught by Kao) for another (using the core processor to transfer boot data from nonvolatile memory to RAM, as taught by Post) and would have understood that Post's processor-based system was ready for the improvements of reduced processor workload, system complexity, and cost.

Id. at 43–46.

Patent Owner argues that “Petitioner’s challenge of this claim element . . . fails because Petitioner improperly characterizes the claimed ‘descriptors’ as a pointer in Post (which does not disclose a DMA controller, much less DMA descriptors), and because Kao’s DMA engine also does not disclose or use DMA descriptors.” Prelim. Resp. 16. However, Patent Owner attacks Petitioner’s showing by arguing the references individually, rather than addressing the combination of teachings relied upon in the Petition. Specifically, Patent Owner argues that Petitioner has not shown that Post discloses the use of DMA components for which Petitioner relies on Kao and Patent Owner argues that Kao does not disclose a descriptor for which Petitioner relies on Post. *See id.* at 16 (“Post’s pointers are not DMA descriptors, and are not included in any DMA descriptors”), 20 (“Petitioner does not and cannot identify any DMA controller descriptor in Kao.”).

As noted above in discussing claim construction (*see supra* Section II.C.), Patent Owner argues that “the plain and ordinary meaning of a DMA descriptor includes more than Post’s pointer.” Pet. 20. However, Patent Owner does not clearly state its contention as to the plain and ordinary meaning of “DMA descriptor” and does not clearly identify what, beyond a pointer, is required by that descriptor. *See* Prelim. Resp. 11 (Patent Owner arguing that “Petitioner’s proposed characterization of the claimed ‘Direct Memory Access (DMA) descriptors’ as satisfied by nothing more than a pointer in Post (which does not disclose a DMA controller or a DMA descriptor) is incorrect and does not comport with the plain and ordinary meaning of this claim term,” but not stating the purported plain and ordinary meaning), *see also id.* at 16–20.

We reject Patent Owner’s argument and determine that the cited art teaches “*wherein the POR sequencer descriptor comprises . . . Direct Memory Access (DMA) descriptors used to fetch other POR sequencer descriptor fragments.*”

With regard to “*wherein the POR sequencer descriptor comprises . . . a system firmware,*” Petitioner cites Post. Pet. 46–47. The Petition states:

Post disclosed that the boot data in NVM 120 included firmware, such as a first-stage bootloader. “The program code may be executed to begin loading additional information from NVM 120 into memory 114. This additional information may include firmware, such as a first stage bootloader....” (INTEL-1005, 3:55-61.) “[T]he firmware (or other data) retrieved from NVM 120 by the mask ROM code may sometimes be referred to as ‘boot data.’” (*Id.*, 3:55-65.)

Id. Patent Owner does not argue that the cited art fails to teach that, “*the POR sequencer descriptor comprises . . . a system firmware.*” See generally Prelim. Resp. We determine that Post teaches these elements.

We determine that the Petition shows that the cited art teaches all the elements of this limitation.

d) copying the POR sequencer descriptor from the nonvolatile memory to a random access memory (RAM)

Petitioner relies on Post for this limitation. Pet. 48–49. The Petition states:

Post disclosed and rendered obvious copying boot data and associated metadata (which, as modified in view of the teachings of Kao and Wilson, was a “POR Sequencer Descriptor”) from nonvolatile memory to a random access memory (“RAM”). (INTEL-1003, ¶ 120.) Post disclosed that control circuitry 112 copied boot data from nonvolatile memory 120 into memory 114. (INTEL-1005, 3:53-65 (control circuitry copied boot data from NVM 120 to memory 114), 5:13-23 (boot data was copied from NVM 120 to memory 114); *id.*, 6:28-29, 5:43-47, Fig. 2.) Memory 114 was a random access memory. (*Id.*, 3:44-49 (“Memory 114 can include any suitable type of volatile memory, such as random access memory (RAM)”)).

Post also disclosed that pointer metadata from each boot page was saved during the process of loading boot data into memory: “If the current page [of boot data] is not the last page, process 500 may instead continue to step 516, where the electronic device can *save the current page’s next page pointer for use in processing subsequent pages.*” (*Id.*, 9:15-18, 9:56-60; *id.*, 8:9-12, 9:7-11.)

Id. at 48 (alteration in original). Patent Owner does not dispute the showing for this limitation. See generally Prelim. Resp. We determine that Post teaches this limitation.

e) verifying an integrity of the POR sequencer descriptor

Petitioner relies on Post for this limitation. Pet. 50. The Petition states:

Post disclosed using next page pointers 302, prior page pointers 304, and boot data page indicators 306 to *verify an integrity* of the boot data. (INTEL-1003, ¶¶ 126-130.) “[T]he electronic device can use the next page pointers to verify the integrity of boot data storage.” (INTEL-1005, 6:54-55; *id.*, 7:21-23 (“As with the next page pointers, the prior page pointers may allow an electronic device to verify and account for all of the boot data stored in the NVM.”), 7:58-64 (boot data page indicators 306 “provide another way in which the electronic device can verify that a page is a boot data page”), 1:55-2:3 (device used “pointers ... [to] load the boot data and verify that pages expected to have boot data actually contain boot data (and vice versa)”); *id.*, 6:22-24, 8:52-66, 9:33-46, 10:52-54; INTEL-1003, ¶¶ 127, 128, 131.) Post also used error correcting codes to verify an integrity of the boot data. (*Id.*, 6:57-59 (“The electronic device may confirm that page 2 provides boot data by, for instance, employing an error correcting code on the contents of page 2.”); INTEL-1003, ¶¶ 129, 131.) Each of these techniques is an example of Post verifying an integrity of the POR sequencer descriptor. (INTEL-1003 ¶¶ 126-131.)

Id. (alterations in original).

f) detecting any error in the POR sequencer descriptor

Petitioner relies on Post for this limitation. Pet. 51. The Petition states:

Post used error correction codes in conjunction with pointer metadata to detect “uncorrectable errors” in the boot code (“*any error in the POR sequencer descriptor*”) indicating that “boot data ha[d] been lost” . . . (INTEL-1005, 6:54-7:5; *id.*, 8:52-66 (detecting error of first boot data page being lost); *id.*, 10:8-32 (identifying “a sign of errors in the boot data”); INTEL-1003 ¶¶ 132-134.)

Id. (alteration in original).

Patent Owner argues that the Petition fails to establish that the cited art teaches “*verifying an integrity of the POR sequence descriptor*” and “*detecting any error in the POR sequence descriptor*.” Prelim. Resp. 22–25. Specifically, Patent Owner contends that the “process of accounting for all boot data occurs with respect to fragments—pages—of boot code data in Post” (*id.* at 22) and “Post is not verifying the integrity of boot code data that has been stored in or loaded from NVM, but instead is using the presence of pointers to ensure that all boot data pages—or fragments—are accounted for” (*id.* at 23). However, as shown in the passages cited in the Petition, Post teaches that, “the loading of boot data can be accomplished by incrementally processing and validating pages as the pages are being read” (Ex. 1005, 5:50–52); “the electronic device can use the next page pointers to verify the integrity of boot data storage” (*id.* at 6:54–55); “the prior page pointers may allow an electronic device to verify and account for all the of the boot data stored in the NVM” (*id.* at 7:21–23); and “the electronic device can perform several checks on the validity and accuracy of the boot data stored in the non-volatile memory” (*id.* at 9:42–44).

Patent Owner also argues that the Petition’s reference to Post’s use of “error correcting code” (*see* Pet. 50–51) does not support the Petitioner’s contentions as to these limitations because the “error correcting code” is “not to check the page for errors.” Prelim. Resp. 24. But the passage in Post cited by the parties teaches “employing an error correcting code on the contents of page 2.” Ex. 1005: 6:59.

We determine that Post teaches “*verifying an integrity of the POR sequence descriptor*” and “*detecting any error in the POR sequence descriptor.*”

g) *wherein the computer-implemented method allows the embedded system to verify an integrity of user data and program code used in a boot process in order to provide a resilient boot up sequence and wherein the computer-implemented method reduces an amount of the nonvolatile memory used in the boot process*

With regard to “*wherein the computer-implemented method allows the embedded system to verify an integrity of user data and program code used in a boot process in order to provide a resilient boot up sequence,*” the Petition states:

As described above . . . , Post described using error correcting codes and pointer metadata to verify an integrity of boot data. (INTEL-1005, 6:54-7:5, 8:52-66, 10:8-11, ; *id.*, 6:22-24, 8:9-12, 8:27-29, 9:7-11; INTEL-1003, ¶¶ 126-134.) This integrity verification was performed by control circuitry 112, a part of Post’s “*embedded system.*” (INTEL-1005, 8:9-12 (processes of Figs. 4-5 performed by control circuitry 112). Post’s boot data was “any information . . . loaded from the NVM based on code stored in a boot ROM” and included “a bootloader” (“*program code used in a boot process*”). (*Id.*, 1:33-43; *id.*, 3:53-65).) As Post’s first-stage bootloader was used to load the device-specific kernel and operating system (*id.*, 3:57-61), this was also *user data*. (INTEL-1003, ¶¶ 136-137.)

Pet. 52. And, Petitioner contends that “[b]y verifying integrity of the boot data, alerting users to errors, and allowing for appropriate firmware updates or other corrective actions, Post’s techniques provided a resilient boot up sequence.” *Id.* at 53.

With regard to “*wherein the computer-implemented method reduces an amount of the nonvolatile memory used in the boot process,*” the Petition states:

Post disclosed that the “NVM 120 can be flash memory” and “can include multiple integrated circuits.” (INTEL-1005, 3:1-14; *id.*, 4:33-34 (“The boot data storage area can include any suitable number of integrated circuits, blocks, or pages.”), 6:22-27 (example of Figure 3 showed “blocks in the boot data storage area of NVM 120” that were “in different integrated circuits or NVM packages”).)

For nonvolatile memory that included multiple integrated circuits, Post’s linked-list technique allowed the amount of nonvolatile memory used in the first integrated circuit to be reduced by distributing boot data fragments across other integrated circuits. (*Id.*, 6:38-53 (“the next page pointer [of a boot data page] can reference another [boot data] page in ... the same or a different integrated circuit”); INTEL-1003, ¶ 147.)

Id. at 54 (alterations in original).

The passages in Post cited in the Petition support the showing for this limitation. Patent Owner does not dispute Petitioner’s showing as to this limitation. We determine that the cited art teaches this limitation.

h) Summary as to Claim 1

We determine that Petitioner has shown a reasonable likelihood of establishing the obviousness of claim 1 and, on that basis, institute *inter partes* review of all the challenged claims. See 37 C.F.R. § 42.108(a) (“When instituting *inter partes* review, the Board will authorize the review to proceed on all the challenged claims.”).

2. Independent Claim 19

Independent claim 19 recites “*An article of manufacture, comprising: a non-transitory computer-readable medium having stored thereon*

instructions operable to permit an apparatus to perform a method comprising . . .” Ex. 1001, 10:49–52. Petitioner shows that these elements of claim 19 are taught by the cited art. *See* Pet. 59–60. For the remaining limitations of claim 19, Petitioner contends that claim 19 “recite[s] limitations that are analogous to those of claim[] 1.” *Id.* at 60. For these remaining limitations of claim 19, Petitioner relies on its showing for claim 1. *Id.* at 61. Patent Owner does not separately address claim 19. *See* Prelim. Resp. 8 (“For the purpose of Petitioner’s challenges, claim 19 is substantially similar to claim 1.”). We determine that, on the current record, Petitioner has met its burden to show the limitations of independent claim 19 of the ’084 patent are taught by the cited art.

3. Dependent Claims 5, 8, 9, and 20

Petitioner shows that the cited art teaches the additional limitations recited in dependent claims 5, 8, 9, and 20. *See* Pet. 56–59, 60–61. The cited art supports Petitioner’s contentions as to the challenged dependent claims. Patent Owner does not address the dependent claims. *See generally* Prelim. Resp. We determine that, on the current record, Petitioner has met its burden to show the limitations of dependent claims 5, 8, 9, and 20 of the ’084 patent are taught by the cited art.

F. Conclusion

For the reasons discussed above, we determine that Petitioner has demonstrated a reasonable likelihood of showing at least one of the claims challenged in the Petition would have been obvious.

III. ORDER

In consideration of the foregoing, it is hereby

ORDERED that, pursuant to 35 U.S.C. § 314(a), an *inter partes* review of the '084 patent is instituted with respect to the challenged claims and the ground set forth in the Petition; and

FURTHER ORDERED pursuant to 35 U.S.C. § 314(a) and 37 C.F.R. § 42.4(b), *inter partes* review of the '084 patent shall commence on the entry date of this Order, and notice is hereby given of the institution of a trial.

PETITIONER:

Christina McCullough
Lori Gordon
Christopher Kelley
PERKINS COIE
mccullough-ptab@perkinscoie.com
gordon-ptab@perkinscoie.com
kelley-ptab@perkinscoie.com

PATENT OWNER:

Wayne Helge
NOSSAMAN LLP
whelge@nossaman.com

James Wilson
jwilson@dbjg.com