

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

SANDISK TECHNOLOGIES, INC. AND WESTERN DIGITAL
TECHNOLOGIES, INC.,
Petitioner

v.

POLARIS POWERLED TECHNOLOGIES, LLC,
Patent Owner

Case IPR2025-00517
U.S. Patent No. 8,554,968

**PATENT OWNER PRELIMINARY RESPONSE
UNDER 37 C.F.R. § 42.107(a)**

Mail Stop "PATENT BOARD"
Patent Trial and Appeal Board
U.S. Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

TABLE OF CONTENTS

I.	INTRODUCTION	1
II.	OVERVIEW OF THE TECHNOLOGY	2
A.	The '968 patent describes a memory controller that manages a host processing unit's completion queue and sends the host an interrupt when the host has an unprocessed completion status.	2
B.	In contrast to the '968 patent, in Borchers', the host monitors its response queue and sends the number of outstanding responses to an interrupt processor.....	14
C.	The advances of the '968 patent are more than just the alleged "interrupt coalescing."	17
D.	For the POPR, Patent Owner applies Petitioners' POSA definition...	19
E.	No term requires a construction to deny institution.	21
III.	GROUND 1: BORCHERS DOES NOT DISCLOSE THE CLAIM ELEMENTS.	21
A.	Borchers does not disclose an interrupt manager (claim 1), an interrupt manager controller (claim 8), or a nonvolatile memory interface (claim 16) "to alert the host processing unit of the unprocessed completion status in the completion queue."	21
1.	Borchers' interrupt processor is not "an interrupt manager" or an "interrupt manager controller."	23
2.	Borchers' interrupt processor does not "alert" the host of unprocessed responses in the queue.	26
B.	Borchers' interrupt processor does not "generat[e]" a completion queue state (claims 1 and 16).	29
C.	Borchers does not disclose "a completion queue state memory" (claim 8).	34
IV.	GROUND 2: THE PCI STANDARDS DO NOT CURE THE DEFICIENCIES IN GROUND 1.....	37
V.	GROUND 3: SEREBRIN DOES NOT CURE THE DEFICIENCIES IN GROUNDS 1 AND 2.....	40
VI.	THE BOARD SHOULD NOT CREDIT DR. MIN'S OPINIONS.	45
VII.	CONCLUSION	47

PATENT OWNER'S UPDATED EXHIBIT LIST

Exhibit	Description
2001	List of Docket Entries, <i>Polaris PowerLED Techs., LLC v. Western Digital Corp. et al.</i> , 1-24-cv-00101 (W.D. Tex.)
2002	List of Docket Entries, <i>Polaris PowerLED Techs., LLC v. Western Digital Corp. et al.</i> , 2-24-cv-02864 (C.D. Cal)
2003	List of Docket Entries, <i>Micron Technology Inc. et al. v. Polaris PowerLED Techs., LLC</i> , 1-25-cv-00073 (D. Idaho)
2004	Scheduling and Case Management Order, <i>Polaris PowerLED Techs., LLC v. Western Digital Corp. et al.</i> , 2-24-cv-02864 (C.D. Cal Jan. 27, 2025), ECF No. 54
2005	Western Digital's Notice of Motion and Motion to Stay Action Pending <i>Inter Partes</i> Review, <i>Polaris PowerLED Techs., LLC v. Western Digital Corp. et al.</i> , 2-24-cv-02864 (C.D. Cal Feb. 3, 2025), ECF No. 58
2006	Minute Order, <i>Polaris PowerLED Techs., LLC v. Western Digital Corp. et al.</i> , 2-24-cv-02864 (C.D. Cal Feb 3, 2025), ECF No. 59
2007	Plaintiff's Opposition to Motion to Stay, <i>Polaris PowerLED Techs., LLC v. Western Digital Corp. et al.</i> , 2-24-cv-02864 (C.D. Cal Feb. 10, 2025), ECF No. 60
2008	Patent L.R. 3-3 and 3-4 Invalidity Disclosures of Western Digital Defendants, <i>Polaris PowerLED Techs., LLC v. Western Digital Corp. et al.</i> , 2-24-cv-02864 (C.D. Cal), ECF No. 60-3 and Appendices A-C of the Invalidity Disclosures
2009	Defendants' Identification of Claim Terms for Construction, <i>Polaris PowerLED Techs., LLC v. Western Digital Corp. et al.</i> , 2-24-cv-02864 (C.D. Cal), ECF No. 60-4
2010	Stipulated Protective Order, <i>Polaris PowerLED Techs., LLC v. Western Digital Corp. et al.</i> , 2-24-cv-02864 (C.D. Cal), ECF No. 61

Exhibit	Description
2011	Western Digital's Reply in Support of Motion to Stay Action Pending <i>Inter Partes</i> Review, <i>Polaris PowerLED Techs., LLC v. Western Digital Corp. et al.</i> , 2-24-cv-02864 (C.D. Cal Feb. 14, 2025), ECF No. 62
2012	USPTO Appeal Statistics (Mar. 31, 2025), <i>available at</i> https://www.uspto.gov/patents/ptab/statistics
2013	Patent Trial & Appeal Board (PTAB) Dashboard, <i>available at</i> https://www.uspto.gov/dashboard/ptab/
2014	Press Release: Sandisk Celebrates Nasdaq Listing After Completing Separation from Western Digital (Feb. 24, 2025), <i>available at</i> https://shop.sandisk.com/company/newsroom/press-releases/2025/sandisk-celebrates-nasdaq-listing-after-completing-separation
2015	U.S. Provisional Patent Application No. 61/374,242
2016	Redline Comparison Between U.S. Patent Application No. 12/756,477 and U.S. Patent Application No. 12/537,733
2017	U.S. Patent No. 10,296,249 to Benisty
2018	IDS Submitted in U.S. Application No. 15/585,808
2019	NVM Express, Revision 1.0 (Mar. 1, 2011), <i>available at</i> https://web.archive.org/web/20110305032748/www.intel.com/standards/nvmhci
2020	U.S. Patent No. 7,620,784 to Panabaker
2021	U.S. Patent No. 8,244,946 to Gupta <i>et al.</i>
2022	U.S. Patent Application Publication No. 2008/0320214 to Ma <i>et al.</i>
2023	U.S. Patent Application Publication No. 2010/0185808 to Yu <i>et al.</i>

Exhibit	Description
2024	Defendants' Identification of Claim Terms for Construction (Apr. 18, 2025), <i>Polaris PowerLED Techs., LLC v. Western Digital Corp. et al.</i> , 2-24-cv-02864 (C.D. Cal. Feb. 10, 2025)
2025	Defendants' Proposed Preliminary Claim Constructions and Intrinsic and Extrinsic Evidence, <i>Polaris PowerLED Techs., LLC v. Western Digital Corp. et al.</i> , 2-24-cv-02864 (C.D. Cal. May 9, 2025)
2026	PCI-SIG Specifications, <i>available at</i> https://pcsig.com/specifications?field_technology_value%5B%5D=express&field_revision_value%5B%5D=2&speclib=

I. INTRODUCTION

The inventors of the '968 patent designed a memory interface where an interrupt manager in a nonvolatile memory controller monitors a host processing unit's completion queue and alerts the host processing unit of an unprocessed completion status. By using *an interrupt manager in the nonvolatile memory controller* to alert the host about an unprocessed completion status, the host would no longer need to monitor its queue. The inventors explained: "Because the interrupt manager alerts the host processing unit of the unprocessed completion status in the completion queue, the host processing unit need not include resources for monitoring the completion queue for an unprocessed completion status."

EX1001, 1:51-55; *see also* EX1001, 16:28-32.

In contrast to the '968 patent, Borchers (the primary reference asserted in all Petitioners' Grounds) discloses a system that used *its host* to monitor the host's queues and manage the interrupt processing—the precise function the inventors of the '968 patent sought to avoid. Specifically, the host monitors its own queue and sends a signal to the interrupt processor titled: "Number of outstanding responses in host queue." *See* EX1005, ¶60, FIG. 2. The interrupt processor then generates an interrupt based on this information. EX1005, ¶¶60-65, FIG. 2.

Since Borchers uses *the host* to monitor its queues and manage interrupts, it does not disclose a nonvolatile memory controller or interrupt manager to alert the

host of an unprocessed completion queue status, as the claims require. This distinction overcomes all three grounds in the Petition.

II. OVERVIEW OF THE TECHNOLOGY

A. **The '968 patent describes a memory controller that manages a host processing unit's completion queue and sends the host an interrupt when the host has an unprocessed completion status.**

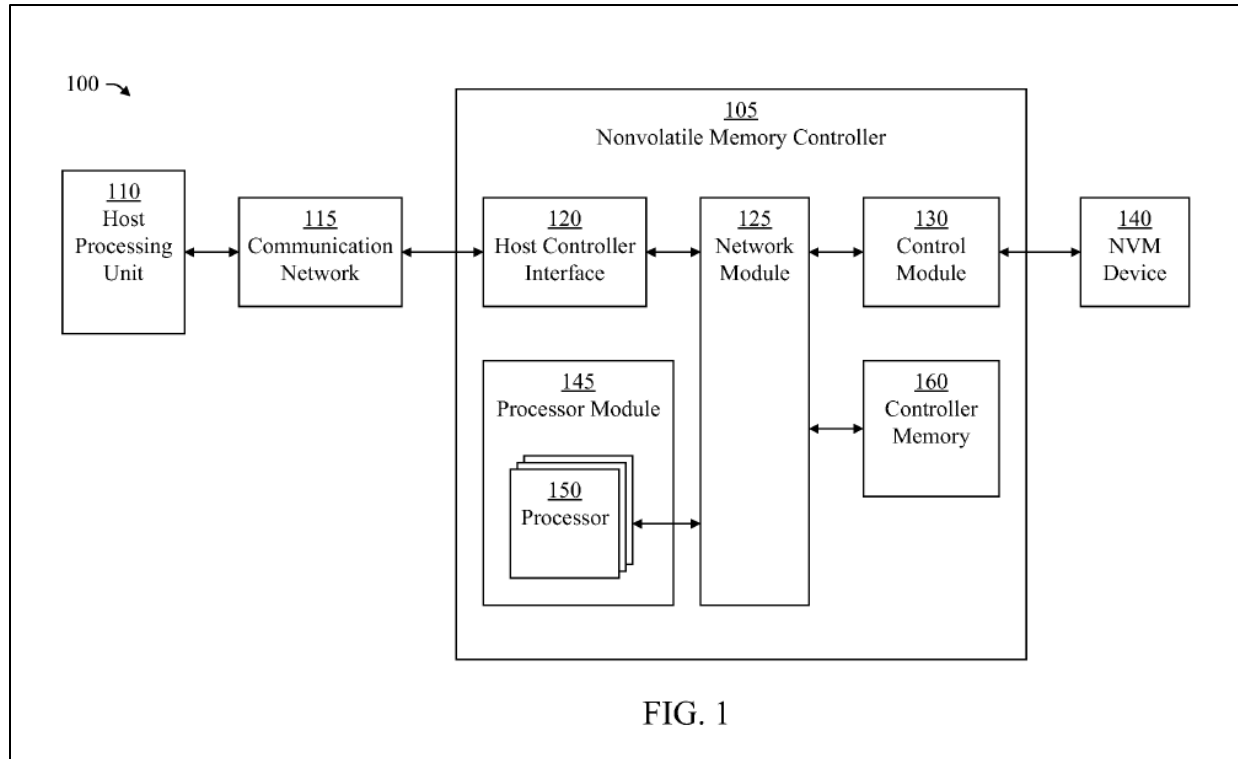
The '968 patent relates to the interface between a host processing unit and a memory controller. When the '968 patent was filed, “Nonvolatile Memory Express (NVMe) [wa]s a standard defining a register interface, command set, and feature set for high-performance Peripheral Component Interconnect Express (PCIe) based solid-state drives (SSDs).” EX1001, 1:15-17. “Although the NVMe standard specifie[d] a register set and a standard command set for designing an NVMe interface, the NVMe standard le[ft] other implementation details open to a designer of the non-volatile memory subsystem.” EX1001, 1:32-35.

As detailed below, a key concept is that the completion queue is managed external to the host processing unit. The inventors of the '968 patent designed a specific implementation of this interface, which was not described in the NVMe standard. Indeed, during prosecution, the applicant provided the examiner with a copy of revision 1.0 of the NVMe standard. EX1002, 144. The examiner considered the standard and determined that it did not disclose the invention: “After close review of the document it does not show the details of the steps

performed by the memory controller as claimed.” EX1002, 157.

A high-level block diagram of an embodiment is shown in Figure 1.

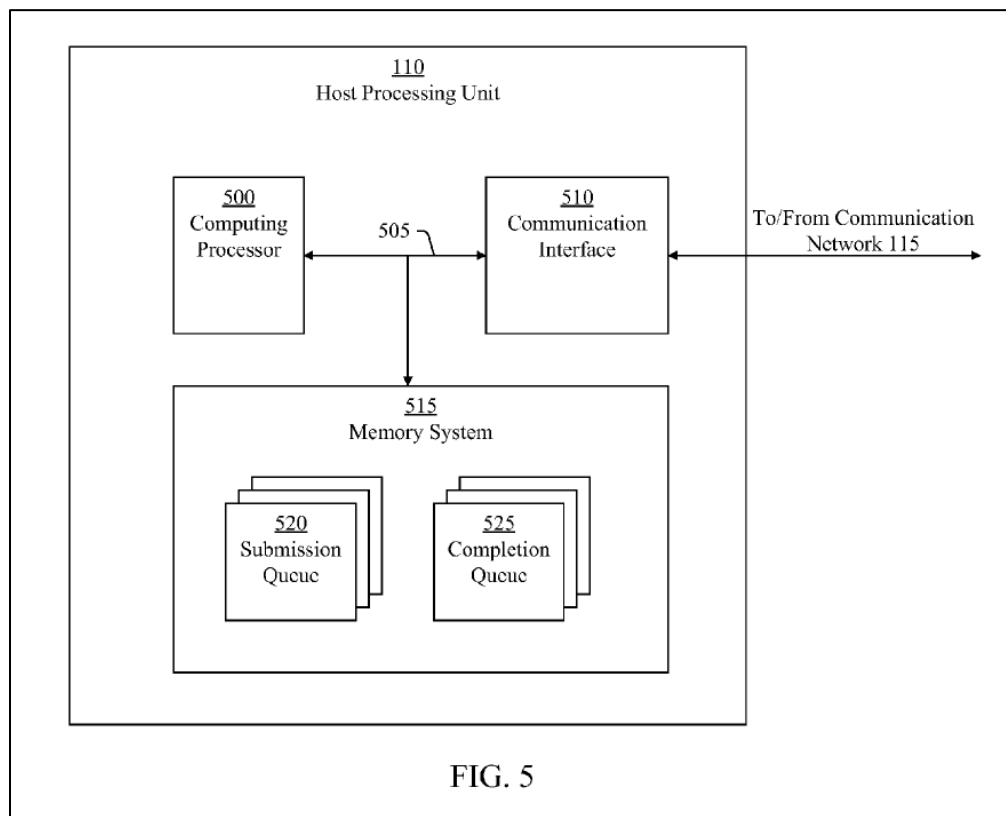
EX1001, 3:42-43. Shown below, “[t]he computing system 100 includes a host processing unit 110, a communication network 115, a nonvolatile memory controller 105, and a nonvolatile memory device 140.” EX1001, 3:43-46. The nonvolatile memory controller 105 can be a flash controller, and the nonvolatile memory device 140 can be flash memory. EX1001, 3:50-53.



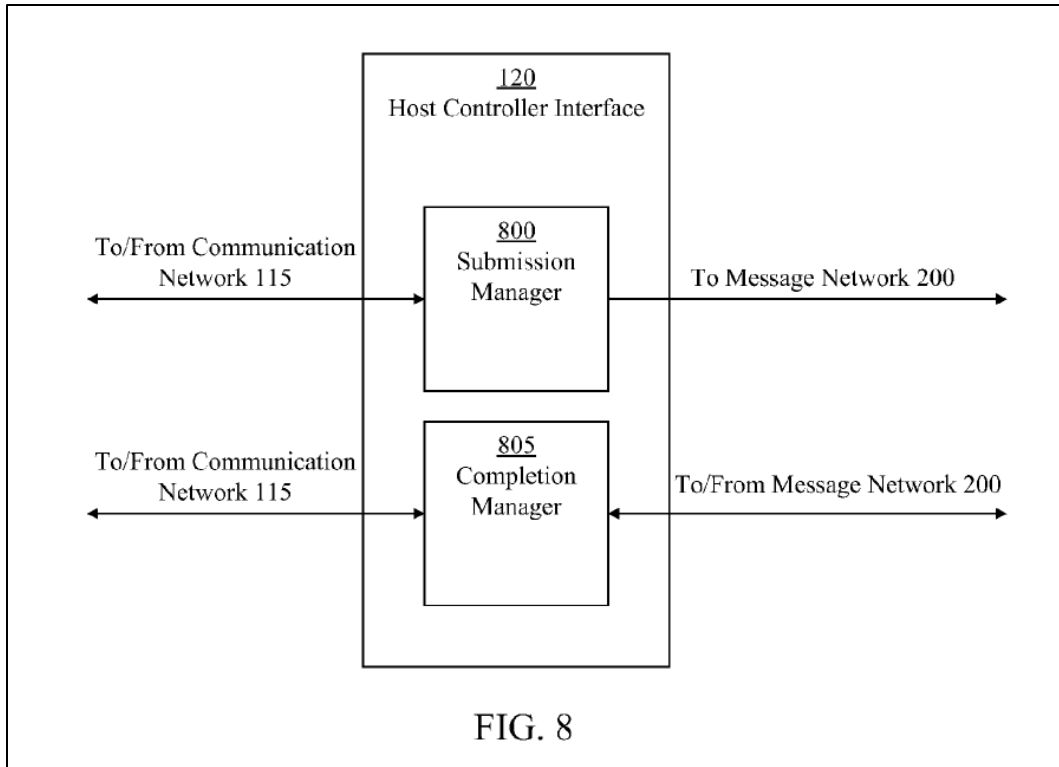
EX1001, FIG. 1.

Figure 5 (below) drills down into the host processing unit 110, and Figure 8 (also below) drills down into the host controller interface 120. Shown below, the host processing unit 110 includes a submission queue 520, which stores

nonvolatile memory commands, and a completion queue 525, which stores completion statuses corresponding to the nonvolatile memory commands. EX1001, 9:38-46. Shown in Figure 8, these queues are managed by the submission manager 800 and completion manager 805 in the nonvolatile memory controller's host controller interface 120. EX1001, 10:65-11:41. The queues are therefore managed external of the host processing unit.



EX1001, FIG. 5.



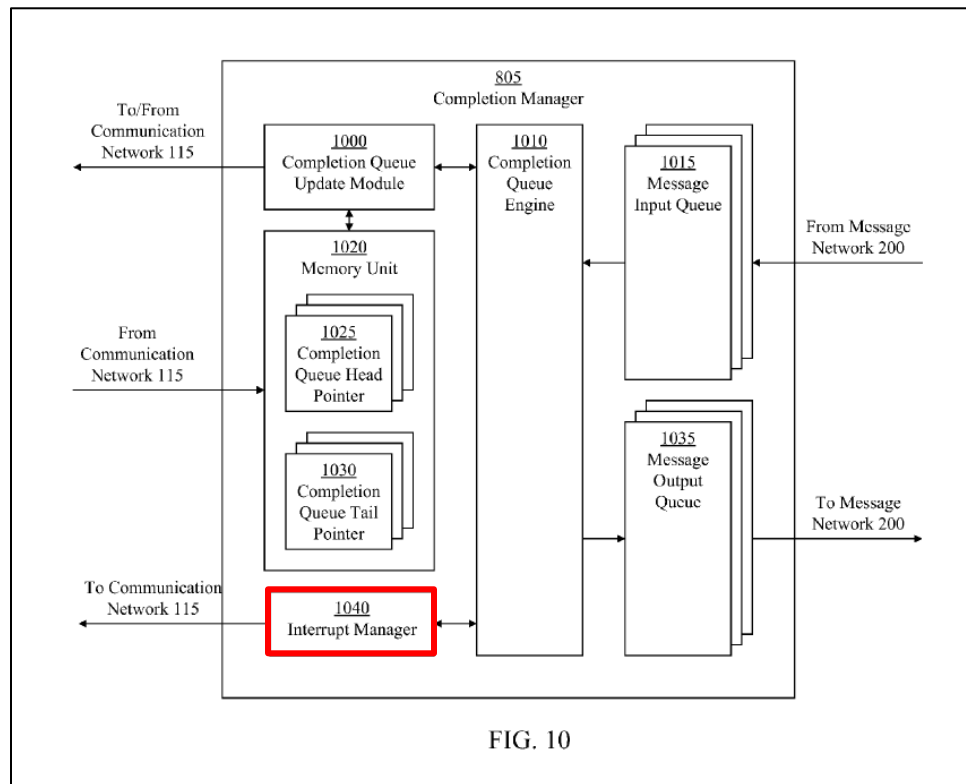
EX1001, FIG. 8.

Unlike other systems at the time, the '968 patent describes and claims a system wherein the host processing unit's commands are managed by *the nonvolatile memory controller's* host controller interface 120. For example, the memory controller directs the retrieval of commands from the host in the following way: "[T]he host processing unit 110 generates nonvolatile memory commands and stores the nonvolatile memory commands in one or more submission queues contained in the host processing unit 110." EX1001, 4:61-64. After the host processing unit generates a command, "[t]he host controller interface 120 determines that a submission queue in the host processing unit 110 contains a nonvolatile memory command, generates a request packet, and transmits the

request packet to the host processing unit 110 through the communication network 115.” EX1001, 4:64-5:2; *see also* EX1001, 11:5-27. In response, the host processing unit 110 generates a completion packet including the nonvolatile memory command and transmits the completion packet to the host controller interface 120. EX1001, 5:2-8. The memory controller’s host controller interface 120 then retrieves the completion packet so that the memory controller can process the command. EX1001, 5:9-20. In this way, the memory controller directs the reception of commands from the host.

The nonvolatile memory controller’s host controller interface 120—specifically its completion manager 805—also manages the host’s completion queue. Once the memory controller processes the command, the nonvolatile memory controller 105 “generates a completion message packet including a completion status (e.g., a completion entry)” EX1001, 5:17-19; *see also* EX1001, 9:55-58, 11:28-37. The nonvolatile memory controller 105 then sends the completion message packet to the host processing unit 110. EX1001, 5:23-26, 9:55-62, 11:28-34. “In turn, the host processing unit 110 stores the completion status of the request message packet received from the host controller interface 120 in a completion queue contained in the host processing unit 110.” EX1001, 5:26-30; *see also* EX1001, 11:37-41. In this way, the memory controller transmits responses to the host.

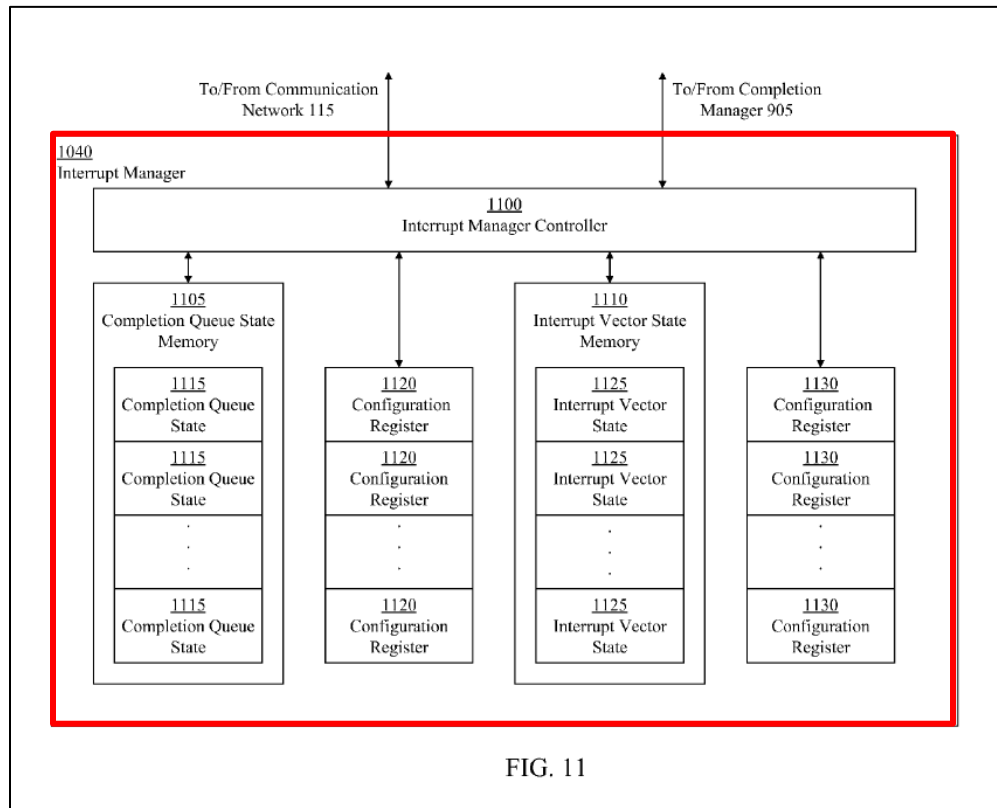
The '968 patent explains how the nonvolatile memory controller's completion manager 805 operates in Figure 10. Shown below, the completion manager 805 includes an interrupt manager 1040, which alerts the host processing unit that the host's completion queue contains an unprocessed completion status. EX1001, 1:44-51, 16:6-11.



EX1001, FIG. 10 (annotated).

The interrupt manager 1040, shown with more detail in Figure 11 below, includes: (i) an interrupt manager controller 1100; (ii) a completion queue state memory 1105 with completion queue states 1115; (iii) configuration registers 1120; (iv) an interrupt vector state memory 1110 with interrupt vector states 1125;

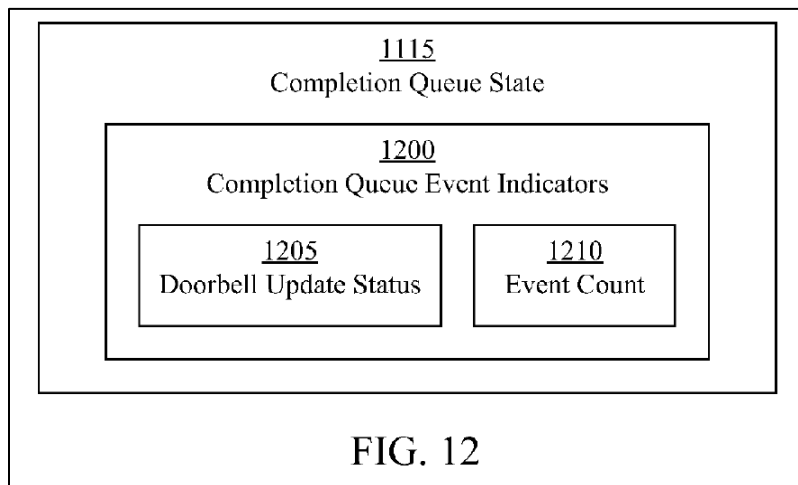
and (v) configuration registers 1130. EX1001, FIG. 11.



EX1001, FIG. 11 (annotated).

“[T]he interrupt manager 1040 (e.g., the interrupt manager controller 1100) monitors completion queue events and generates completion queue states 1115” EX1001, 15:13-17. “The interrupt manager 1040 stores the completion queue states 1115 into the completion queue state memory 1105.” EX1001, 15:20-22. Figure 12 (below) shows an embodiment of a completion queue state 1115 that includes two completion queue event indicators 1200: (1) a doorbell update status 1205 and (2) an event count 1210. EX1001, 17:64-18:2. “The doorbell update status 1205 indicates whether the host processing unit 110 has performed a

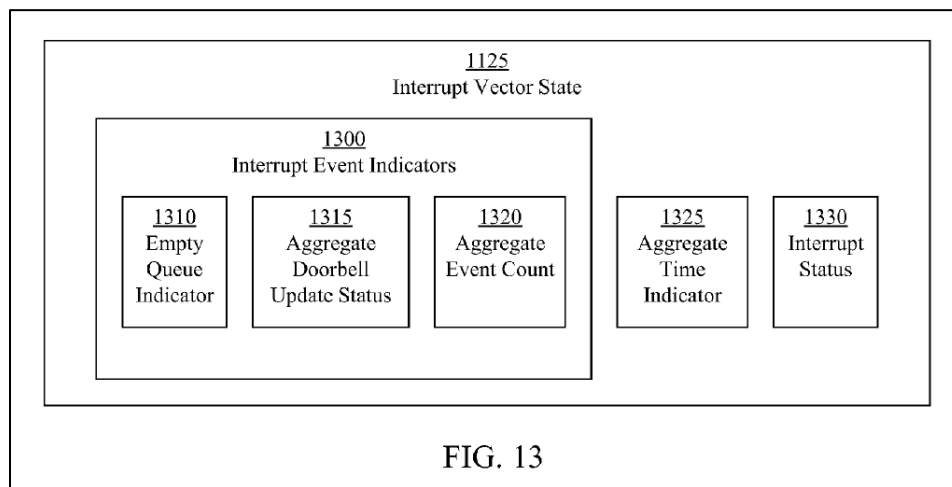
doorbell update event since the last time the interrupt manager 1040 cleared the completion queue state 1115.” EX1001, 18:3-6. In a doorbell update event, the host processing unit 110 updates the completion queue head pointer 1025 of the completion queue 525. EX1001, 18:6-9. The completion queue event indicators 1200 also include an event count 1210. EX1001, FIG. 12. The event count 1210 indicates the number of completion packets generated in which a completion queue event flag is set since the last time the interrupt manger 1040 cleared the completion queue state 1115. EX1001, 18:23-28.



EX1001, FIG. 12.

Referring back to Figure 11, “the interrupt manager 1040 generates interrupt vector states 1125 based on the completion queue states.” EX1001, 15:23-24. “The interrupt vector state 1125 of an interrupt vector indicates whether one or more completion queues 525 associated with the interrupt vector state 1125 contain an unprocessed completion status” EX1001, 15:27-30. The interrupt vector states

1125 are also stored in memory—interrupt vector state memory 1110. EX1001, 15:31-32. Shown below in Figure 13, the interrupt vector state 1125 can have multiple interrupt event indicators 1300, an aggregate time indicator 1325, and an interrupt status 1330. EX1001, 18:37-41. In this embodiment, the interrupt event indicators include an empty queue indicator 1310, an aggregate doorbell update status 1315, and an aggregate event count 1320. EX1001, 18:41-44.



EX1001, FIG. 13.

“The interrupt manager 1040 determines based on the interrupt vector state 1125 whether any completion queue 525 associated with the interrupt vector state 1125 contains an unprocessed completion status.” EX1001, 16:3-6. For example, the empty queue indicator 1310 indicates that the completion queues associated with the interrupt vector state 1125 are empty. EX1001, 18:59-67. If the empty queue indicator 1310 is set, the interrupt manager 1040 knows that the completion queues 525 associated with the interrupt vector state 1125 do not have an

unprocessed completion status. EX1001, 18:59-19:5. In contrast, “[i]f the interrupt manager 1040 determines at least one of the completion queues 525 contains an unprocessed completion status, the interrupt manager 1040 generates an interrupt message packet, which includes the interrupt vector associated with the interrupt vector state 1125.” EX1001, 16:6-11. If certain conditions are met (e.g., an event counter threshold is met, and/or a timer threshold is met), the interrupt manager 1040 transmits the interrupt message packet to the host processing unit 110 to alert the host of an unprocessed completion status. EX1001, 16:18-28, 19:15-57.

“Because the nonvolatile memory controller 105 alerts the host processing unit 110 to an unprocessed completion status in the completion queue 525, the host processing unit 110 need not include resources for monitoring the completion queue 525 for an unprocessed completion status.” EX1001, 16:28-32. In this way, the memory controller manages the submission and response queues in the host.

The ’968 patent claims recite such a nonvolatile memory controller with an interrupt manager that: generates a completion queue state, generates an interrupt vector state based on the completion queue state, determines whether the completion queue of a host processing unit contains an unprocessed completion status based on the interrupt vector state, and generates an interrupt message packet for triggering an interrupt. As shown below, all three independent claims also require the non-volatile memory controller to “alert” the host processing unit of an

uncompleted completion status in the completion queue.

1. *A nonvolatile memory controller for alerting a host processing unit to an unprocessed completion status contained in a completion queue of the host processing unit, the nonvolatile memory controller comprising:*

an interrupt manager configured to generate a completion queue state for indicating the occurrence of a completion queue event associated with the completion queue, generate an interrupt vector state based on the completion queue state, determine the completion queue of the host processing unit contains an unprocessed completion status based on the interrupt vector state, and generate an interrupt message packet for triggering an interrupt in the host processing unit to alert the host processing unit of the unprocessed completion status in the completion queue, and wherein the completion queue state includes a doorbell update status indicating whether the host processing unit has performed a doorbell update event in which the host processing unit updates a head pointer stored in the nonvolatile memory controller for the completion queue.

EX1001, 29:23-42 (emphasis added).

8. *A nonvolatile memory controller comprising:*

a completion queue state memory configured to store a plurality of completion queue states corresponding to a plurality of completion queues in a host processing unit;

an interrupt vector state memory configured to store a plurality of interrupt vector states corresponding to a plurality of interrupt vectors in the host processing unit; and

an interrupt manager controller coupled to the completion queue state memory and the interrupt vector state memory, *the interrupt manager controller configured to map at least one completion queue state of the plurality of completion queue states to an interrupt vector state of the plurality of interrupt vector states, generate the interrupt vector state based on each completion queue state mapped to the interrupt vector state, determine a completion queue corresponding to a completion queue state mapped to the interrupt vector state contains an unprocessed completion status based on the interrupt vector state, and to generate an interrupt message packet based on the interrupt vector state for triggering an interrupt in the host processing unit to alert the host processing unit of the unprocessed completion status in the completion queue.*

EX1001, 30:5-29 (emphasis added).

16. A method comprising:

generating a completion queue state by a nonvolatile memory controller for indicating the occurrence of a completion queue event associated with a completion queue in a host processing unit, the completion queue state including a doorbell update status indicating whether the host processing unit has performed a doorbell update event in which the host processing unit updates a head pointer stored in the nonvolatile memory controller for the completion queue;

generating an interrupt vector state by the nonvolatile memory controller based on the completion queue state, the interrupt vector state associated with an interrupt vector in the host processing unit;

determining by the nonvolatile memory controller based on the interrupt vector state that the completion queue contains an unprocessed completion status; and

generating an interrupt message packet by a nonvolatile memory interface for triggering an interrupt in the host processing unit to alert the host processing unit of the unprocessed completion status in the completion queue.

EX1001, 31:4-32:7 (emphasis added).

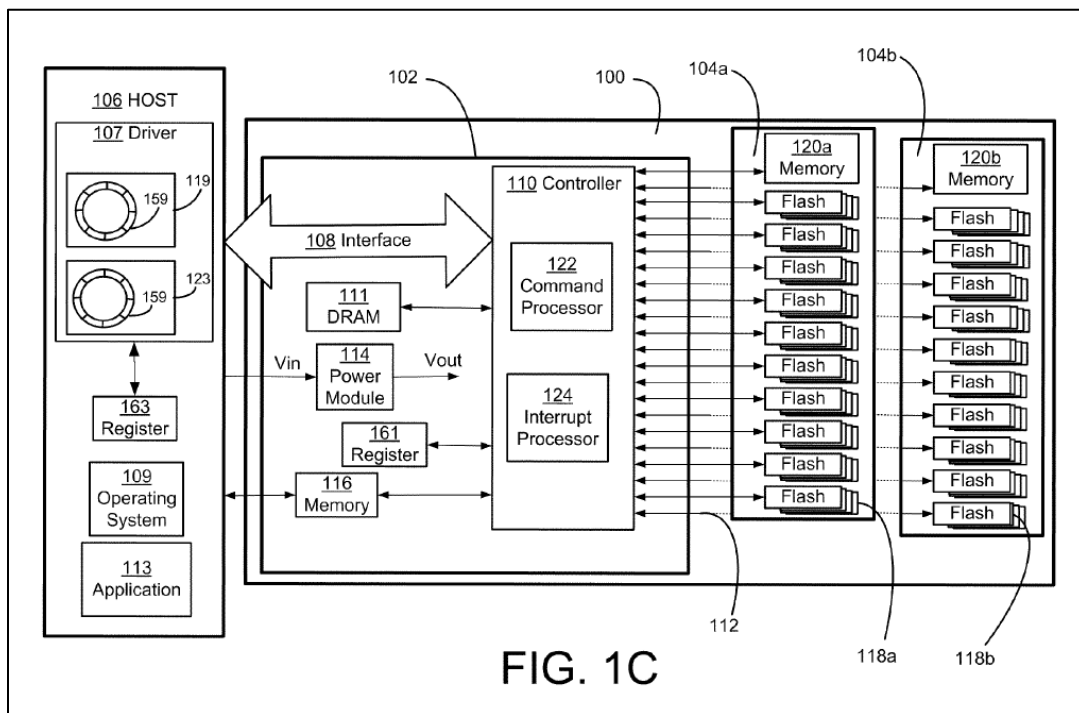
B. In contrast to the '968 patent, in Borchers', the host monitors its response queue and sends the number of outstanding responses to an interrupt processor.

Petitioners rely on the same primary reference, Borchers, for all three grounds. Pet., 18. Borchers is a reference that was already cited and overcome during prosecution. EX1002, 120, 157.

Borchers describes a host that uses a controller interface to execute commands for storing and retrieving data in/from flash memory. *See* EX1005, ¶109, FIG. 1C. But Borchers, unlike the '968 patent, does *not* use a memory controller to monitor the host's completion queue and alert the host processing unit of an unprocessed completion status in the completion queue. Borchers also does not reduce the processing load at the host for monitoring the completion queue for

an unprocessed completion status. Rather, Borchers uses *the host* to monitor its command queue and manage the interrupt signaling. *See, e.g.,* EX1005, ¶¶30, 60, FIG. 2 (maintaining its response queue's tail pointer and reading the response queue's head pointer to determine the number of outstanding responses and sending the number of outstanding responses to the interrupt processor).

Figure 1C (below) is a block diagram of circular queues used to communicate information between a host 106 and a data storage device 100. EX1005, ¶30. Shown below, the host 106 has a circular queue 159 for commands and a circular queue 159 for responses. EX1005, ¶30.

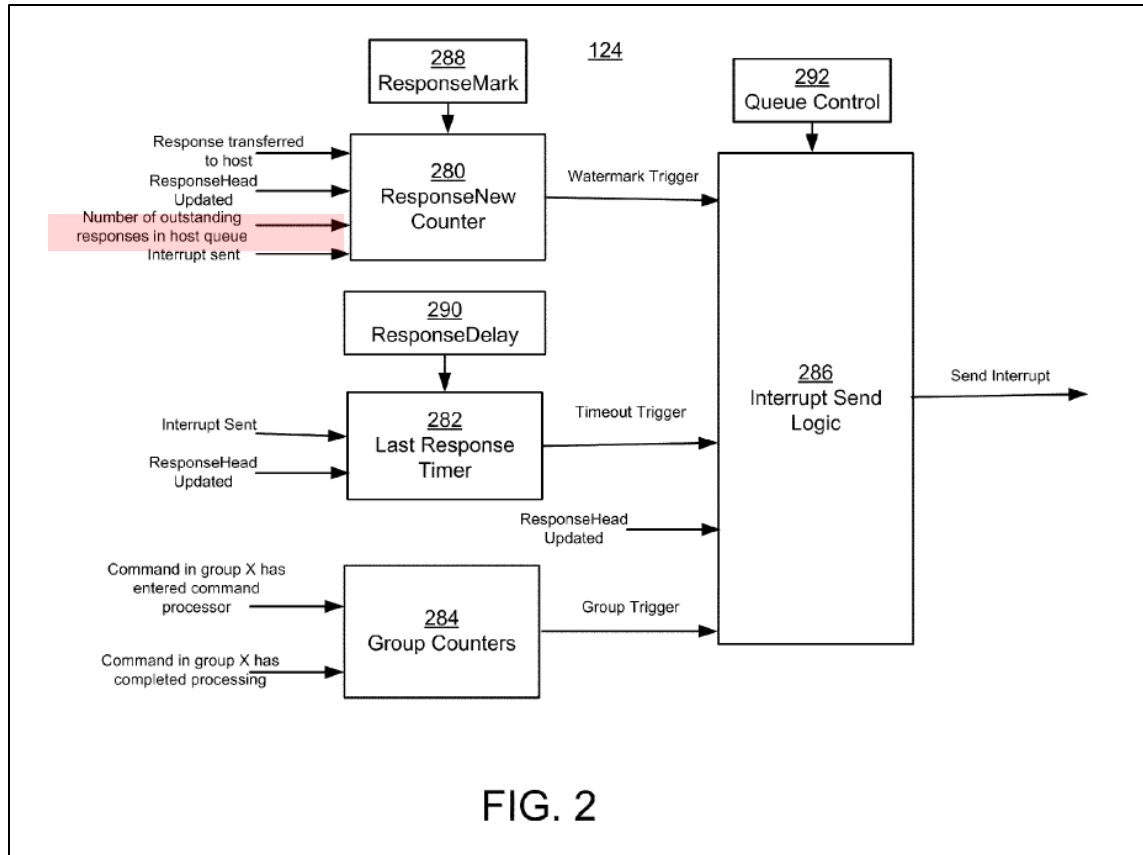


EX1005, FIG. 1C.

Borchers explains, on multiple occasions, that *the host* manages the interrupt

processing. Borchers states: “In one exemplary implementation, *the host 106*, through its driver 107, may control when the interrupt processor 124 should generate interrupts.” EX1005, ¶36 (emphasis added). Borchers also states that “[t]he *host 106* may use one or more different interrupt mechanisms, including a combination of different interrupt mechanisms, to provide information to the interrupt processor 124 regarding interrupt processing.” EX1005, ¶36 (emphasis added). The host also manages interrupt conditions, including “a ResponseMark parameter, which determines the water mark” and “the ResponseDelay parameter, which determines the timeout.” EX1005, ¶37. Both of these parameters are described in more detail in reference to Figure 2.

Figure 2 of Borchers (below), shows an interrupt processor 124 configured to send interrupts “based on the interrupts mechanism or mechanisms enabled by the host 106.” EX1005, ¶59. Again, *the host* is responsible for monitoring its queues and managing the interrupts. *See, e.g.*, EX1005, ¶¶30, 60, FIG. 2. For example, inputs to the interrupt processor 124, received from host 106, include the “[n]umber of outstanding responses in host queue.” *See* EX1005, FIG. 2. The inputs to the interrupt processor 124 also include the other parameters managed by the host. EX1005, ¶37, FIG. 2.



EX1005, FIG. 2 (annotated).

In Borchers, the host already knows the number of outstanding responses in its queue and communicates this information to the interrupt processor. This is in stark contrast to the '968 patent, which uses an interrupt manager to monitor the host's queue and alert the host processing unit of an unprocessed completion status in the queue. *See* EX1001, 1:51-55, 16:28-42.

C. The advances of the '968 patent are more than just the alleged “interrupt coalescing.”

In their attempt to equate the '968 patent and Borchers, Petitioners mischaracterize the invention as mere “interrupt coalescing”—a term not in the

'968 patent or Borchers. Petitioners introduce this term in the first sentence of the Petition and refer to it throughout. *See* Pet., 7, 16, 17, 20, 22, 24, 89, 90. Petitioners also allege that “[t]he purported novelty of the '968 patent is to hold off on sending the interrupt signal until certain conditions (e.g., by a timer or response counter) are met,” a technique they say is “referred to as interrupt coalescing.” Pet., 7.

Although the '968 patent *includes* functionality where the nonvolatile memory controller does not send an interrupt until certain conditions are met (e.g., the completion queue is not empty, an event counter threshold is met, and/or a timer threshold is met), 19:15-56, the '968 patent does not characterize the invention as interrupt coalescing. Instead, the inventors disclosed key features, including monitoring the host processing unit's submission queue and completion queue *at a nonvolatile memory controller* instead of either the host processing unit or nonvolatile memory device. *E.g.*, EX1001, 1:51-55, 16:28-42. By using the nonvolatile memory controller, the host processing unit and nonvolatile memory device are freed from this processing burden. The '968 patent explains:

Because the interrupt manager alerts the host processing unit of the unprocessed completion status in the completion queue, the host processing unit need not include resources for monitoring the completion queue for an unprocessed completion status.

EX1001, 1:51-55.

Because the nonvolatile memory controller 105 alerts the host processing unit 110 to an unprocessed completion status in the completion queue 525, the host processing unit 110 need not include resources for monitoring the completion queue 525 for an unprocessed completion status. Furthermore, because the interrupt manager 1040 generates interrupt message packets for alerting the host processing unit 110 of unprocessed completion statuses in the completion queues 525 of the host processing unit 110, the processors 150 of the nonvolatile memory controller 105 need not consume computing resources for determining whether the completion queues 525 of the host processing unit 110 contain unprocessed completion statuses or for alerting the host processing unit 110 of unprocessed completion statuses in the completion queues 525.

EX1001, 16:28-42.

As the examiner determined during prosecution, “independent claim 1 includes the limitations of the details of the operation of the interrupt manager’s operation which are not shown in the prior art.” EX1002, 157. Borchers, which the examiner already considered (EX1002, 120), does not include this operation.

D. For the POPR, Patent Owner applies Petitioners’ POSA definition.

For the purposes of this preliminary response, Patent Owner applies Petitioners’ proffered level of skill in the art with a caveat regarding the last sentence:

A person of ordinary skill in the art (“POSITA”) in the field of the ’968 patent, as of its earliest possible filing date of August 16, 2010, would have been someone knowledgeable and familiar with memory systems. A POSITA would have had a bachelor’s degree in electrical engineering, or an equivalent, and approximately two years of professional working [sic] in the field of memory systems. Lack of professional experience can be remedied by additional education, and vice versa. Ex.1003, ¶¶21-23.

Pet., 17. To the extent the last sentence means that additional education could completely supplant experience, Patent Owner disagrees. A POSITA would be a professional with active work experience in the field of memory technologies, including at least some hands-on experience in that domain.

Petitioners’ expert, Dr. Min, does not appear to have any “professional [experience] working in the field of memory systems.” *See* EX1003, ¶¶7-20; EX1004 (Dr. Min’s CV). While Dr. Min appears to have expertise in networking and wireless communications, his experience with memory is limited to incorporating high speed memory modules into network chips that he has designed, based on his description in his declaration. *See* EX1003, ¶¶7-20. His lack of experience in the field of memory systems is reflected in the opinions in his declaration, which follow the basic formula of: providing a block quote from a reference, and then a conclusory statement that a claim element is taught or obvious. *See, e.g.*, EX1003, ¶¶86, 88. As discussed in this POPR, such conclusory

opinions are insufficient to demonstrate obviousness and are entitled to little weight. *See Xerox Corp. v. Bytemark, Inc.*, IPR2022-00624, Paper 12 at 15 (P.T.A.B. Feb. 10, 2023) (precedential).

E. No term requires a construction to deny institution.

Claim terms need only be construed to the extent necessary to resolve the controversy. *Wellman, Inc. v. Eastman Chem. Co.*, 642 F.3d 1355, 1361 (Fed. Cir. 2011). Because none of the issues in this POPR turn on the precise construction of a claim term, Patent Owner takes no position at this time.

III. GROUND 1: BORCHERS DOES NOT DISCLOSE THE CLAIM ELEMENTS.

Borchers does not disclose many core elements of the '968 patent claims. It does not disclose an interrupt manager, interrupt manager controller, or nonvolatile memory interface “to alert the host processing unit of the unprocessed completion status in the completion queue” as recited in all three independent claims. Borchers’ interrupt processor also does not “generat[e]” a completion queue state as recited in independent claims 1 and 16. Nor does it include a “completion queue state memory” as recited in claim 8. For these reasons, Petitioners do not have a reasonable likelihood of success under Ground 1.

A. Borchers does not disclose an interrupt manager (claim 1), an interrupt manager controller (claim 8), or a nonvolatile memory interface (claim 16) “to alert the host processing unit of the unprocessed completion status in the completion queue.”

Borchers fails to disclose “an interrupt manager” (claim 1), “an interrupt

manager controller” (claim 8), or “a nonvolatile memory interface” (claim 16) that generates an interrupt message packet “to alert the host processing unit of the unprocessed completion status in the completion queue.” Instead, Borchers’ host monitors its queue and sends the interrupt processor the number of outstanding responses in the host’s queue. *Supra* Section II.B. This failure dooms ground 1.

Claims 1 and 8 both recite a nonvolatile memory controller comprising either an “interrupt manager” (claim 1) or an “interrupt manager controller” to “alert the host processing unit of the unprocessed completion status in the completion queue.” In district court, Western Digital has proposed the same construction for both “interrupt manager” (claim 1) and “interrupt manager controller” (claim 8)—“functionality performed within the nonvolatile memory controller for alerting the host processing unit to an unprocessed completion status.” EX2025, *3.¹ Western Digital has also claimed that the preamble of claim

¹ In addition to the arguments in Paper 9, the Board should deny institution because the Petition did not set forth how the challenged claims are to be construed. In the Petition, Petitioners alleged “no claim term requires express construction.” Pet., 17. Yet Western Digital proposed constructions for seven claim terms in district court. EX2025, *3-7. The Petition “must” set forth how the challenged claim is to be construed. 37 C.F.R. § 42.104(b)(3). Petitioners did not meet this requirement.

1 is limiting, which also specifies that the nonvolatile memory controller is for “alerting a host processing unit to an unprocessed completion status contained in a completion queue of the host processing unit.” EX2025, *3.

Although claim 16, as a method claim, does not recite an “interrupt manager” or “interrupt manager controller,” it also includes the “alert” limitation, reciting “generating an interrupt message packet by a nonvolatile memory interface for triggering an interrupt in the host processing unit to alert the host processing unit of the unprocessed completion status in the completion queue.”

Borchers does not include an interrupt manager, an interrupt manager controller, or a nonvolatile memory interface to “alert” the host processing unit of an unprocessed completion status in the completion queue. In Borchers, the host—not the interrupt processor—manages the interrupts. The host also monitors its queue, knows that it contains unprocessed responses, and does not need alerts.

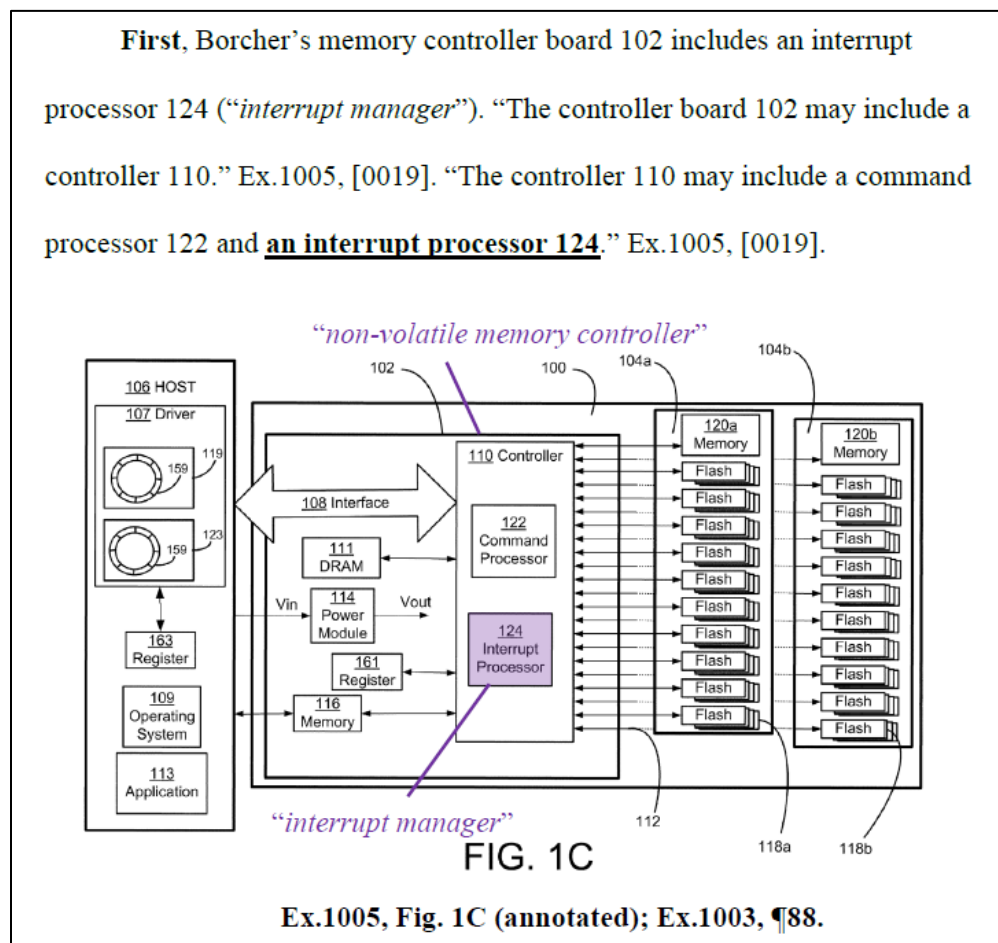
1. Borchers’ interrupt processor is not “an interrupt manager” or an “interrupt manager controller.”

Borchers’ interrupt processor 124 is not an interrupt manager or an interrupt manager controller as recited in claims 1 and 8 respectively. Nor does Borchers’ interrupt processor satisfy Petitioners’ own district-court construction for “interrupt manager” and “interrupt manager controller,” i.e., “functionality performed within the nonvolatile memory controller for alerting the host processing unit to an unprocessed completion status,” EX2025, *3, because it does not “alert” a host

processing unit to an unprocessed completion status.

Petitioners do not provide *any* basis for concluding that the interrupt processor 124 is a “manager.” For this limitation, Petitioners include a single conclusory sentence, a single quote, and a single annotated figure. Pet., 31. They never explain *why* the processor is a manager. And they ignore the construction that they proposed in district court, failing to explain how the interrupt processor 124 “alerts” the host processing unit to an unprocessed completion status.

Petitioners’ analysis is so trivial that it can be reproduced below in its entirety.



Pet., 31 (annotating EX1005, FIG. 1C).

Petitioners' expert declaration fares no better. Dr. Min reproduces the same first sentence verbatim, the same quote, and the same annotated figure. EX1003, ¶88. Neither Petitioners nor Dr. Min provide any reason for making the logical leap from an "interrupt processor" to an "interrupt *manager*" or "interrupt *manager* controller." *See* Pet., 31, 62; EX1003, ¶¶88, 162. Such conclusory arguments should be given little to no weight. *Xerox Corp. v. Bytemark, Inc.*, IPR2022-00624, Paper 12 at 15 (P.T.A.B. Feb. 10, 2023) (precedential).

Borchers does not use its interrupt processor 124 as a manager because that function is performed by the host. In Borchers, *the host* 106 manages when the interrupt processor 124 generates interrupts. EX1005, ¶36. For example, "the host 106 may use one or more different interrupt mechanisms, including a combination of different mechanisms, to provide information to the interrupt processor 124 regarding interrupt processing." EX1005, ¶36. The host 106 also sets interrupt parameters. EX1005, ¶37. The host 106 may "poll the response buffer 123 to determine if there are responses ready for processing." EX1005, ¶40. The host 106 may also "poll ahead and avoid interrupts from the interrupt processor 124." EX1005, ¶37. And the host 106 may use a group interrupt mechanism to dictate when the interrupt processor 124 generates and sends interrupts. EX1005, 38. In contrast to the host 106, which is in charge of interrupt processing, the interrupt processor 124 merely executes instructions and/or parameters set by the host 106.

See EX1005, ¶59.

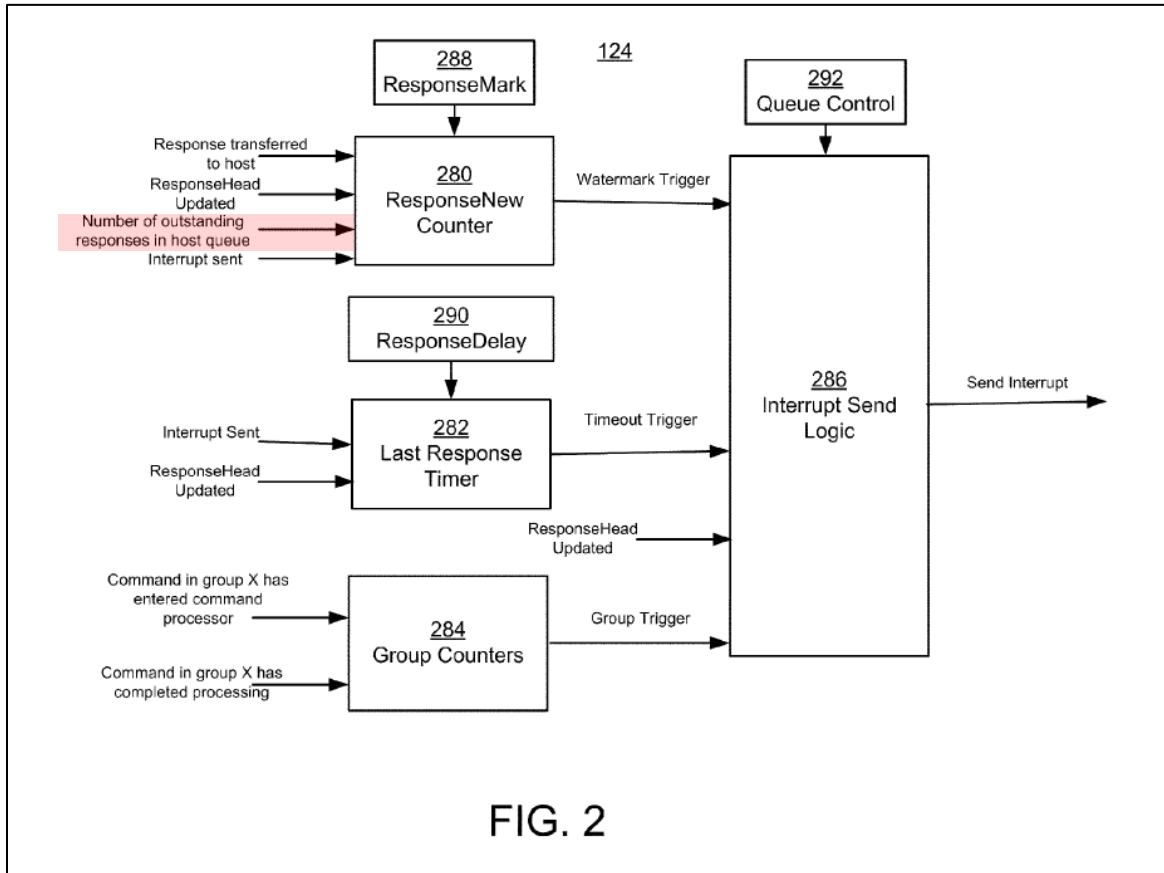
Borchers' interrupt processor does not “manage” but merely executes instructions, so it cannot disclose the “interrupt manager” or “interrupt manager controller” in claims 1 and 8. Nor does Borchers' interrupt processor satisfy Petitioners' own district-court construction for “interrupt manager” and “interrupt manager controller” because it does not “alert” the host processing unit to an unprocessed completion status, as explained in the next section.

2. Borchers' interrupt processor does not “alert” the host of unprocessed responses in the queue.

Even more problematic for Petitioners, Borchers' interrupt processor 124 does not “alert the host processing unit of the unprocessed completion status in the completion queue.” *See* Pet., 41-42, 66-67, 75. Instead, Borchers' host 106 monitors the number of outstanding responses in its queue and sends this information to the interrupt processor 124. *See* EX1005, ¶30 (maintaining its response queue's tail pointer and reading the response queue's head pointer to determine the number of outstanding responses), 60, FIG. 2 (sending the “Number of outstanding responses in host queue” to the interrupt processor).

Shown below, the interrupt processor's ResponseNew Counter 280 receives the “Number of outstanding responses in the host queue.” EX1005, ¶60. Both

parties agree that this signal is “from the host system.” See Pet., 31; EX1003, ¶89.²



EX1005, FIG. 2 (annotated).

In Borchers, the host already knows the “number of outstanding responses in

² Petitioners argue that “Borchers’ interrupt processor 124 (shown in detail in Fig. 2) receives information **from the host system** as to the status of the response queue (*‘completion queue’*).” Pet., 31 (bold emphasis added). Petitioners then state that “the response queue status information includes . . . the number of outstanding items in the response queue.” Pet., 31.

[its] host queue,” so an interrupt sent by interrupt processor 124 would not “alert” the host of an outstanding response in its queue.

This difference between Borchers and the '968 patent claims is not a distinction without a difference. The '968 patent's use of an “alert” to the host processor of an unprocessed completion status is precisely what the inventors explained was an improvement over prior systems. In the first paragraph of the '968 patent's summary section, the inventors explained that such alerts would free up resources otherwise spent by the host processing unit: “Because the interrupt manager *alerts* the host processing unit of the unprocessed completion status in the completion queue, the host processing unit need not include resources for monitoring the completion queue for an unprocessed completion status.” EX1001, 1:51-55 (emphasis added). The inventors reiterated this point when discussing an exemplary embodiment.

Because the nonvolatile memory controller 105 alerts the host processing unit 110 to an unprocessed completion status in the completion queue 525, the host processing unit 110 need not include resources for monitoring the completion queue 525 for an unprocessed completion status.

EX1001, 16:28-42.

Borchers' system does not use its interrupt processor to monitor the host's queue and “alert the host processing unit of the unprocessed completion status in

the completion queue.” Accordingly, Borchers fails to disclose the “alert” limitation, which is recited in all three independent claims.

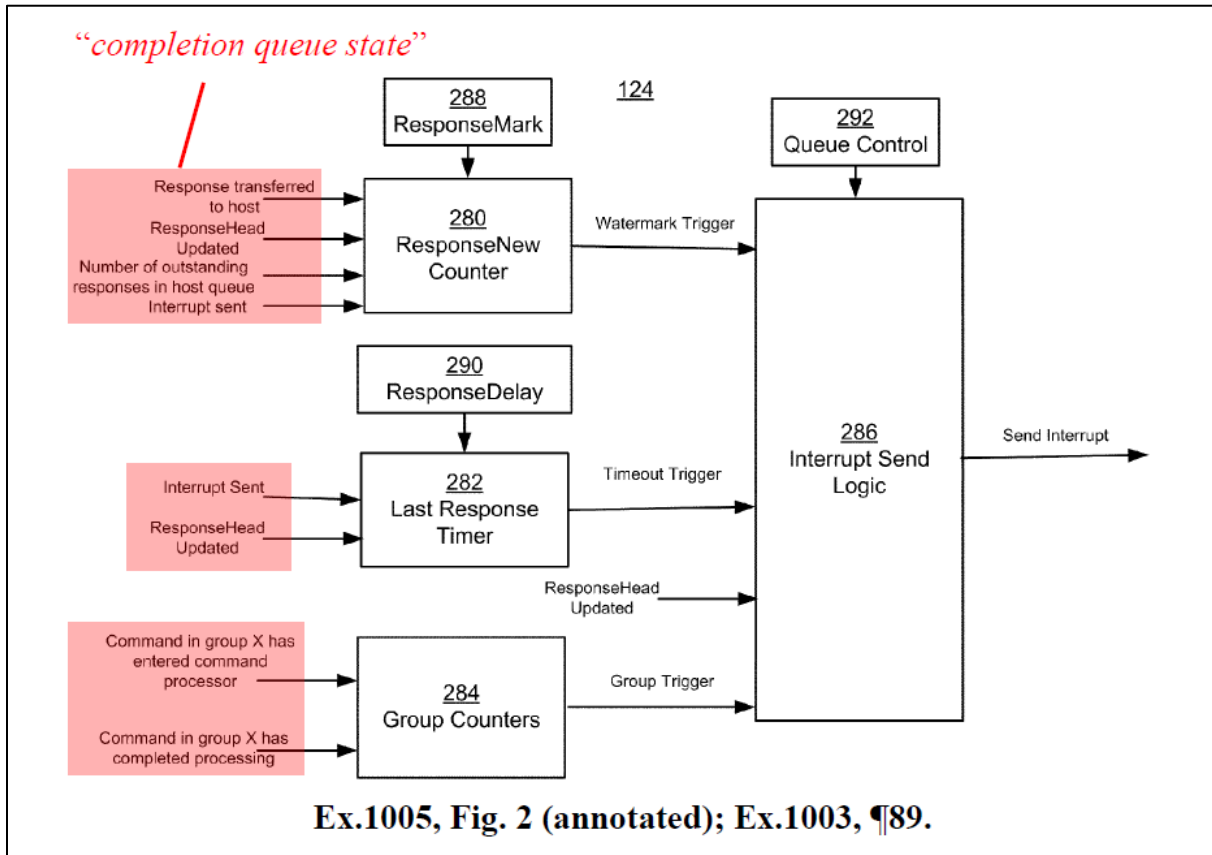
B. Borchers’ interrupt processor does not “generat[e]” a completion queue state (claims 1 and 16).

Ground 1 also fails because Borchers’ interrupt processor does not “generate a completion queue state for indicating the occurrence of a completion queue event associated with the completion queue” as recited in claim 1 and similarly recited in claim 16. Instead, Borchers’ interrupt processor receives the alleged completion queue state information from the host.

Claims 1 and 16 both require a nonvolatile memory controller that “generat[es]” a completion queue state. Claim 1 recites a “nonvolatile memory controller comprising: an interrupt manager configured to generate a completion queue state for indicating the occurrence of a completion queue event associated with the completion queue.” Claim 16 recites a method that “generat[es] a completion queue state by a nonvolatile memory controller for indicating the occurrence of a completion queue event associated with a completion queue in a host processing unit.”

For this element, Petitioners rely on receipt of information from the host as the alleged “completion queue state.” Pet., 31-34, 74. According to Petitioners, “Borchers’ interrupt processor 124 (shown in detail in Fig. 2) *receives information from the host system* as to the status of the response queue (‘completion queue’).”

Pet., 31 (emphasis removed and new emphasis added). Petitioners then annotate all input signals to interrupt processor 124 as the alleged “completion queue state.”



Pet., 32 (annotating EX1005, FIG. 2).

Petitioners are inconsistent on which signal(s) constitute the “completion queue state.” For this element and others, Petitioners allege all the signals are collectively the “completion queue state.” *E.g.*, Pet., 32, 44. At other times, Petitioners allege each input signal is individually a “completion queue state.” *E.g.*, Pet., 65. Petitioners’ annotations above also suggest that they may allege that different subsets of the signals are each a completion queue state. *See* Pet., 32. The ambiguity is itself a basis for denying institution. *See* 35 U.S.C. § 312(a)(3)

(requiring petitioners to identify the grounds “with particularity”); 37 C.F.R. § 42.104(b) (requiring petitioners to identify challenge with specificity).

Regardless of whether Petitioners are alleging that all the input signals are collectively a “completion queue state,” each input signal is individually a “completion queue state,” or some subset of signals is the completion queue state, the claims do not recite “receiving” a completion queue state. Claim 1 recites “an interrupt manager configured to *generate* a completion queue state,” and claim 16 recites “*generating* a completion queue state by a nonvolatile memory controller.” Petitioners have not shown such “generation.”

As far as Patent Owner can tell, Petitioners are interpreting the term “generat[e/ing]” as: (1) “reciev[e/ing]” and/or (2) “stor[e/ing].” despite alleging, at Pet., 17, that “no claim term requires express construction” and applying the plain and ordinary meaning. Pet., 33-34. Petitioners state: “[t]his status information is “*generate[d]*” by the interrupt processor when it is received and/or stored by Borchers’ interrupt processor 124 as illustrated in Fig. 2.” Pet., 33. This approach of disguising constructions violates 37 C.F.R. § 42.104(b)(3), which requires Petitioners to set forth how the challenged claims are to be construed.

Not only is Petitioners’ approach procedurally defective, Petitioners have offered no reason why the Board should construe “generat[e/ing]” as “receiv[e/ing]” and/or “stor[e/ing].” Indeed, Petitioners’ interpretation is especially

problematic because the '968 patent differentiates these terms within the specification and claim language. “In the absence of any evidence to the contrary, we must presume that the use of these different terms in the claims connotes different meanings.” *CAE Screenplates Inc. v. Heinrich Fielder GMBH & Co.*, 224 F.3d 1308, 1317 (Fed. Cir. 2000). Petitioners have not overcome that presumption.

First, Petitioners argue that the interrupt processor 124 “generates” the alleged completion queue state because it “receives” the signals. Pet., 33. Receiving information by itself does not “generate” the information—those terms have different meanings and are used differently in the '968 patent.

On multiple occasions, the '968 patent refers to both receiving information and generating information in the same sentence. As shown below, the '968 patent does not equate the two, and states that some processing of received information occurs to “generate” new information.

The functional unit *receiving* a request message packet from the processor 150 through the network module 125 processes the command in the request message packet, *generates* a completion message packet including a completion status based on the request packet, and provides the completion message packet to the network module 125.

EX1001, 5:46-51 (emphasis added).

[A] processor 150 *receives* a request message packet including a nonvolatile memory command from the host controller interface 120 through the network module 125 and processes the nonvolatile memory

command in the request message packet by *generating* additional request message packets, each of which includes a command.

EX1001, 7:38-43 (emphasis added).

In some cases, each functional unit *receiving* a request message packet from the processor module 145 *generates* one or more request data packets based on the request message packet.

EX1001, 6:3-6 (emphasis added). There are numerous other examples of the '968 patent differentiating receiving and generating. *E.g.*, EX1001, 5:56-64, 7:48-53, 7:58-66, 11:28-34.

The claims also differentiate the two terms. *See* EX1001, Claims 1 (“an interrupt manager configured to *generate* a completion queue state”), 12 (“completion message packets *received* by a host controller interface”). Despite the specification and claims using both terms, Petitioners have offered no evidence to rebut the presumption that different terms connote different meanings.

Petitioners’ second claim interpretation also fails. Petitioners next allege that the interrupt processor 124 “generates” the alleged completion queue state because it “stores” the information. Pet., 33. Even if the alleged completion queue state information were stored (it is not, as explained in the next section), storing is not the same thing as generating. Indeed, the claims differentiate between a “memory configured to *store* a plurality of completion queue states” (claim 8) and “*generat[ing]* a completion queue state” (claims 1, 16). Linguistic differentiation

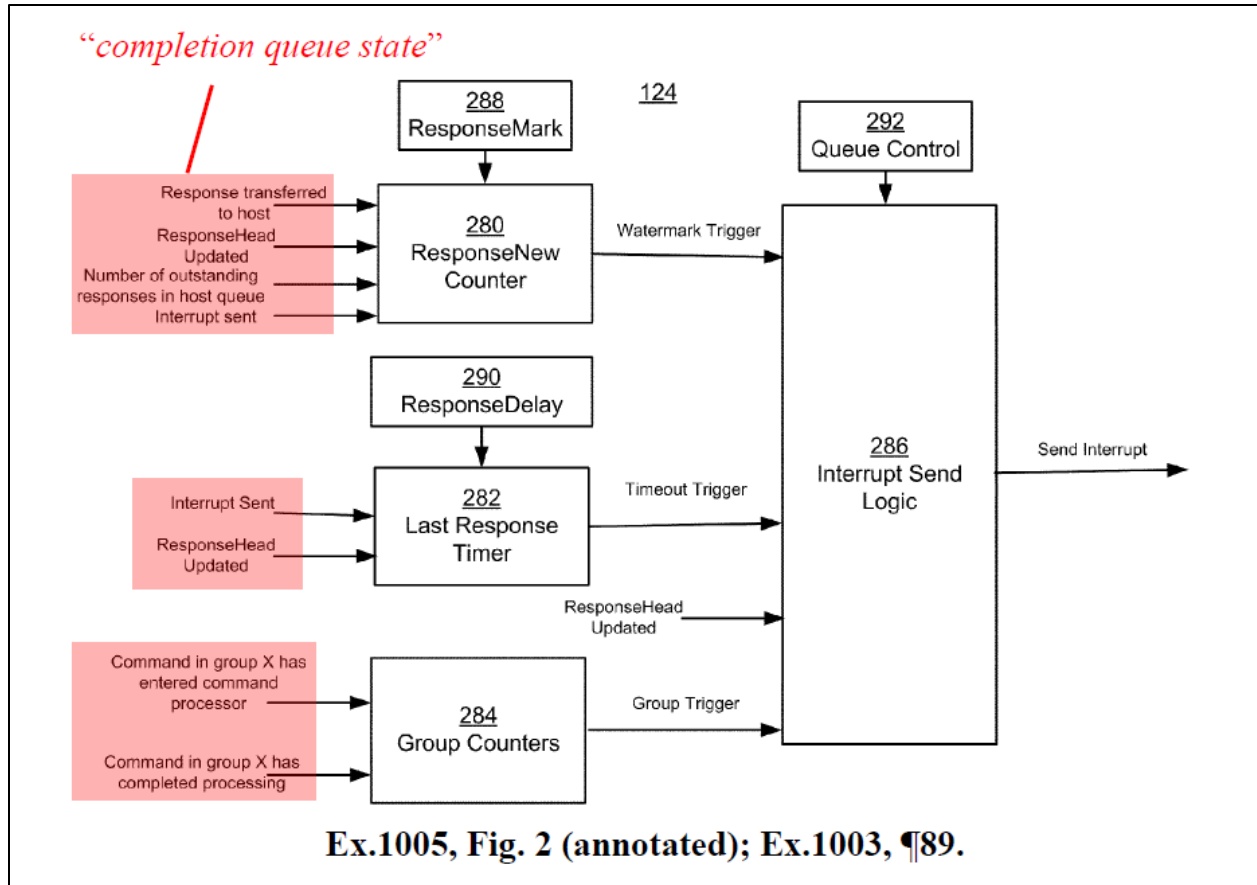
precludes these terms from having the same meaning, and Petitioners have cited no support to the contrary. *See* Pet., 33.

Borchers' interrupt processor does not “generate” a completion queue state for indicating the occurrence of a completion queue event as recited in claim 1 and similarly recited in claim 16. Nor has Petitioners provided any evidence showing that “generating” means “receiving” and/or “storing.” Ground 1 deficient for these additional reasons.

C. Borchers does not disclose “a completion queue state memory” (claim 8).

Borchers does not disclose “a completion queue state memory” and “an interrupt vector state memory” as recited in claim 8 because the alleged state information is transitory and merely exists as signals input to control logic.

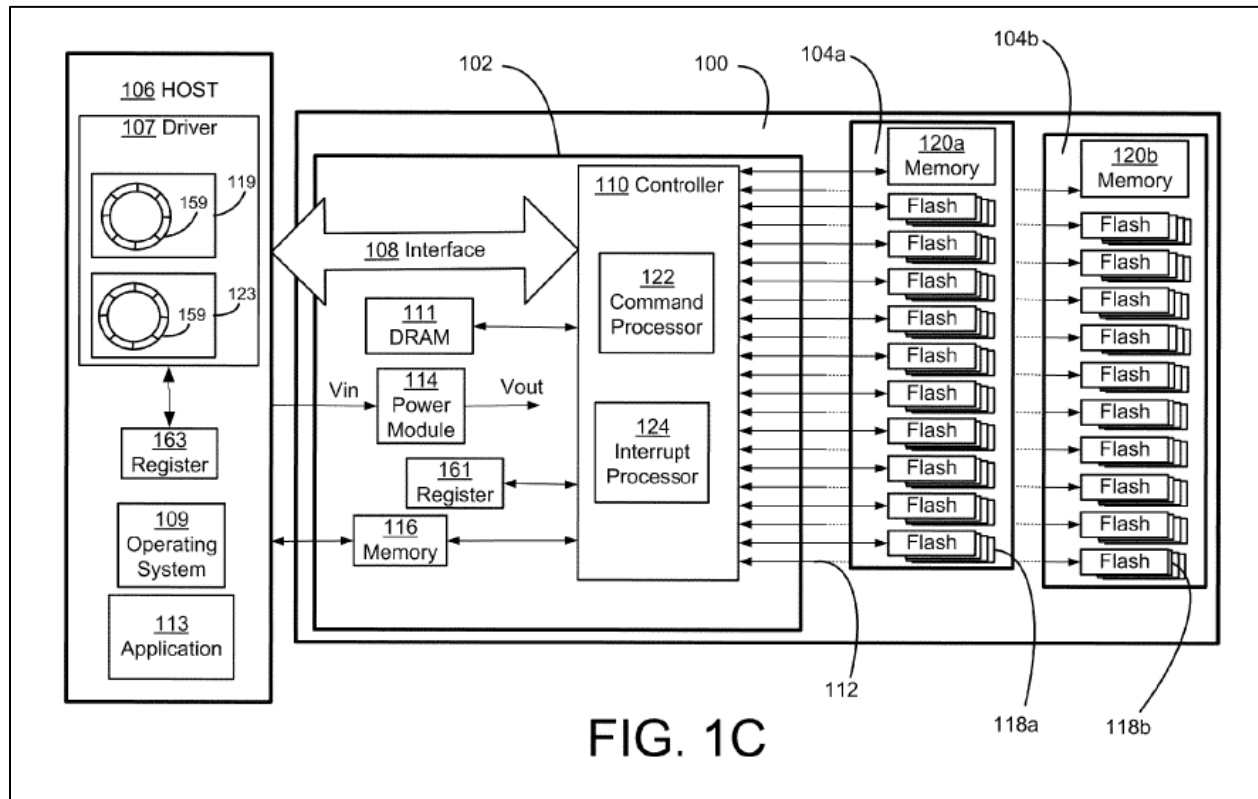
As shown again below, Petitioners allege that signals received as inputs to interrupt processor 124 are “a completion queue state.” Regardless of whether Petitioners allege that all of these signals are collectively the completion queue state, each subset of the signals is a completion queue state, or individual signals are a completion queue state—it is entirely unclear, in violation of 35 U.S.C. § 312(a)(3) and 37 C.F.R. § 42.104(b)—Borchers never discloses that data received via these signals are stored in memory.



Pet., 32 (annotating EX1005, FIG. 2).

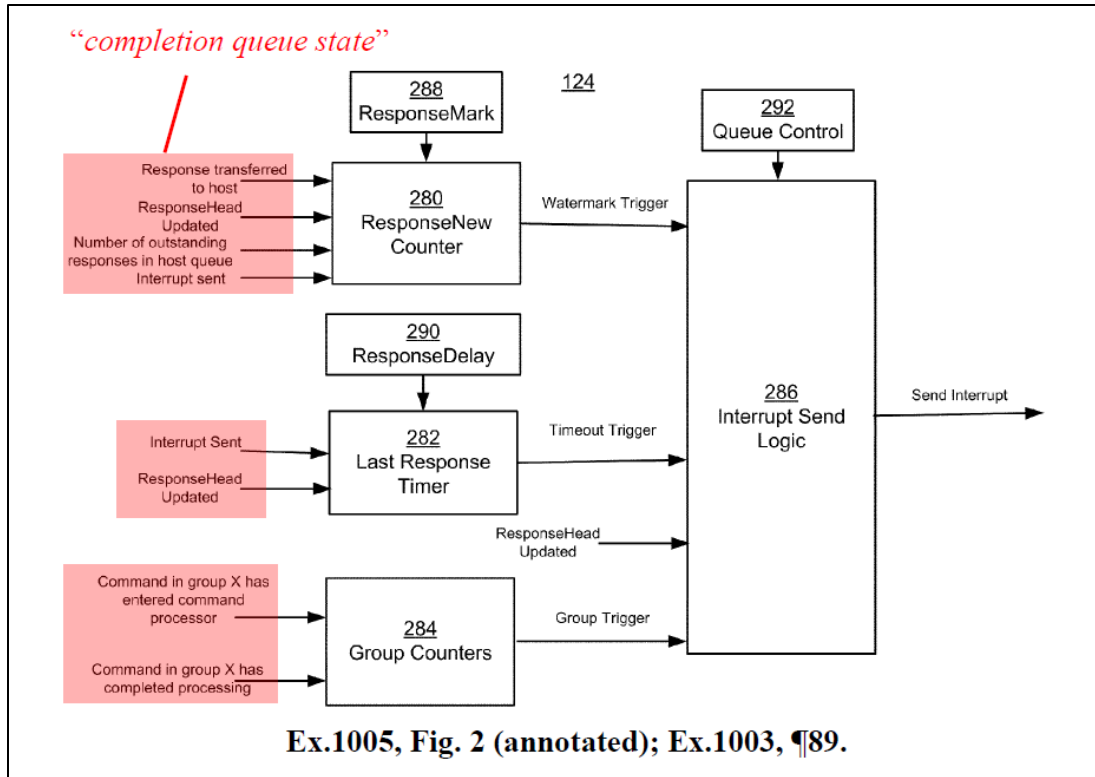
According to Petitioners, “Borchers’ memory controllers includes [sic] ‘register 161’ that *may* be used to store or buffer such status information.” Pet., 56 (citing EX1005, ¶30) (emphasis added). But shown below, register 161 is not part of controller 110 or interrupt processor 124. The register 161 is also used to store different information—values of the tail pointer of command buffer 119. EX1005, ¶30. As Petitioners implicitly admit by using permissive language “may” (Pet., 56), Borchers does not disclose that register 161 is used to store the alleged completion queue state / any of the information input to interrupt processor 124. Nor do

Petitioners provide any reason to do so.



EX1005, FIG. 1C.

There would have been no need for Borchers to buffer or temporarily store the alleged completion queue state as Petitioners allege (at 56). Borchers receives the alleged completion queue state as inputs to ResponseNew Counter 280, Last Response Timer 282, and Group Counters 284. As Petitioners' annotated Figure 2 shows, the signals are used to increment a counter/timer, which is then used as a comparator to compare its value to a threshold. See EX1005, ¶¶60-64. Once the signals increment the respective counters, there is no reason to save the information. Nor does Petitioner allege there is a reason to save the info.



Pet., 32 (annotating EX1005, FIG. 2).

Borchers does not disclose “[a] nonvolatile memory controller comprising: a completion queue state memory configured to store a plurality of completion queue states” as recited in claim 8. Nor have Petitioners articulated sufficient reasons to store additional information in register 161. Claim 8 and its dependent claims are therefore patentable over Borchers.

IV. GROUND 2: THE PCI STANDARDS DO NOT CURE THE DEFICIENCIES IN GROUND 1.

Petitioners admit “[t]he claim analysis for Ground 2 is substantially the same as that of Ground 1,” Pet., 80, so Ground 2 fails for the same reasons as Ground 1.

While it is unclear which claim elements Petitioners allege that the PCI

Standards disclose in Ground 2 (Pet., 76-80), Petitioners' discussion about MSI and MSI-x interrupts does not fix the issues with Borchers. For Ground 2, the Petition includes a "Summary of the PCI Standards" and "Reasons to combine the PCI standard with Borchers." Pet., 77-80. In these sections, Petitioners claim that it would have been natural for a POSITA to implement Borchers' memory interface 108 in accordance with industry standards such that Borchers' interrupts would use MSI and/or MSI-x interrupts as described in the PCI Standards. Pet., 77-79. Petitioners then allege that MSI and MSI-x interrupts use interrupt "vectors." Pet., 79. Neither MSI / MSI-x interrupts, nor interrupt vectors are relevant to Borchers' deficiencies identified in this POPR for Ground 1. When it comes to the deficiencies of Borchers, such as not having an "interrupt manager" / "interrupt manager controller" to "alert" the host processor of an unprocessed completion status in the completion queue, Petitioners do not cite the PCI Standards. *E.g.*, Pet., 30 (relying on Borchers' interrupt processor 124), 31 (relying on Borchers' interrupt signal output from interrupt send logic 286). The PCI Standards therefore do not cure the fatal flaws identified above for Ground 1.

Ground 2 is also legally flawed because Petitioners presented the PCI Standards as a single reference, despite it comprising two specifications from different releases of different standards, PCI and PCIe. *See* Pet., 18-19. When relying upon multiple references to show obviousness, the challenger must explain

why one of ordinary skill in the art would be motivated to combine or modify the references and what would have provided the basis of a reasonable expectation of success. *See KSR Int’l Co. v. Teleflex Inc.*, 550 U.S. 398, 418 (2007). Yet Petitioners have provided *no* motivation to combine analysis or basis for treating these two references as a single reference. *See Pet.*, 77-80.

Petitioners’ approach is particularly problematic here, where they combine two distinct standards developed in different eras. PCI was first developed in the early 1990s whereas PCIe was developed in the mid 2000s. EX1006, 2; EX1007, 2. Additionally, Petitioners have not explained why they chose the particular version and revision of each respective standard. They rely on Revision 3.0 of the PCI Local Bus Specification (EX1006) and Revision 0.9 of the PCI Express 2.0 Base Specification (EX1007). And Revision 0.9 of PCIe 2.0 is not even a final, adopted specification; it is a draft or prerelease. *Compare* EX1007 (dated Sept. 11, 2006), *with* EX2026, 7 (PCIe 2.0 released Dec. 20, 2006). Petitioners’ cherry picking of versions, revisions, *and drafts*, and how these specifications interrelate, remains a mystery.

The Board’s handling of similar facts in *Verint* is instructive. *See Verint Sys. Inc. v. Elad Barkan*, IPR2018-01579, Paper 8 (Mar. 7, 2019). In *Verint*, the petitioner consolidated several documents describing different versions of a standard for Global System for Mobile (“GSM”) encrypted communications, used

the shorthand “the GSM Standard” to refer to the collective, and included them in an obviousness ground. *See Verint*, IPR2018-01579, Paper 8 at 5-6. Petitioner then cited to the documents interchangeably, without any reasons for combining them. *Id.* at 13. In its decision *denying* institution, the Board determined that “[t]his consolidated approach renders Petitioner’s position inconsistent, confusing, and unclear.” *Id.* at 13-14. It also “fail[ed] to satisfy the requirements of 35 U.S.C. § 312(a)(3) and 37 C.F.R. §§ 42.22(a)(2), 42.104(b)(2), 42.104(b)(4)-(5).

The Board should issue the same result here. Grouping two different standards together—not just two versions of the same standard—and treating them as a single reference in an obviousness combination is insufficient. Petitioners must instead articulate *a reason* to combine multiple references in an obviousness combination and present different versions of a standard. *KSR*, 550 U.S. at 418. They must also identify the grounds of unpatentability with particularity. 35 U.S.C. § 312(a)(3) and 37 C.F.R. §§ 42.22(a)(2), 42.104(b)(2), 42.104(b)(4)-(5). Failing to do so, as is the case here, should lead to denial of institution. *See Verint*, IPR2018-01579, Paper 8 at 31.

For these reasons and the reasons explained above for Ground 1, Petitioners do not have a reasonable likelihood of success under Ground 2.

V. GROUND 3: SEREBRIN DOES NOT CURE THE DEFICIENCIES IN GROUNDS 1 AND 2.

Similar to Ground 3, Petitioners admit “[t]he claim analysis for Ground 3 is

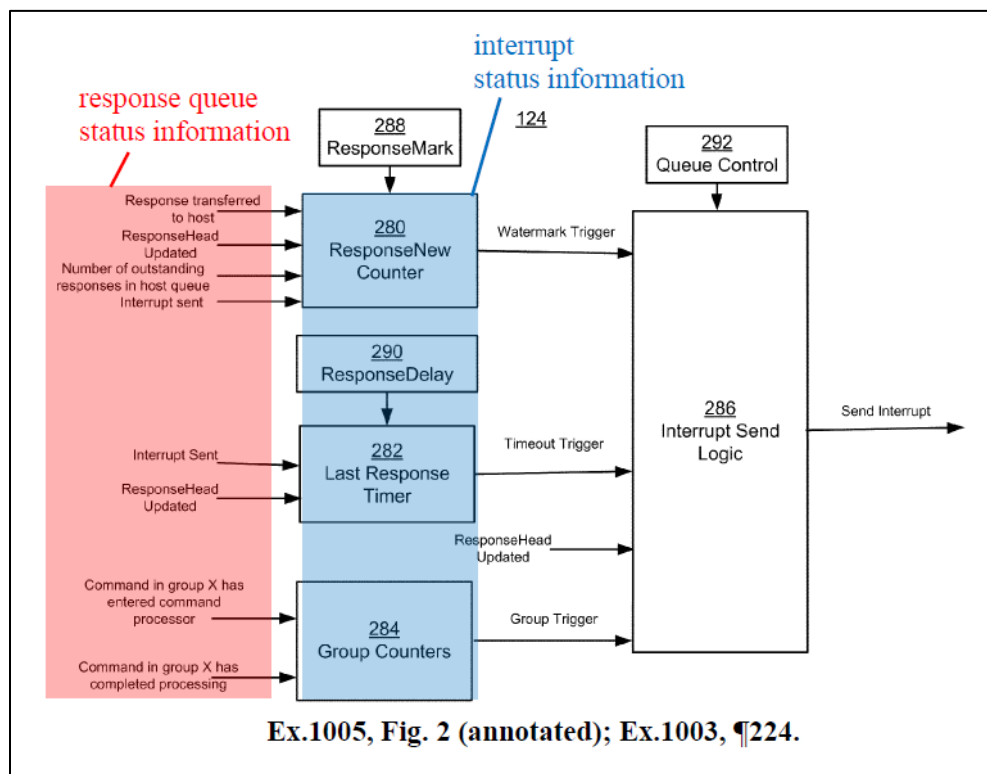
substantially the same as that of Ground 1,” Pet., 87, so Ground 3 fails for the same reasons as Ground 1. Ground 3 also includes the PCI Standards from Ground 2, although Petitioners never discuss the standards within Ground 3. *See* Pet., 80-87.

While it is unclear which claim elements Petitioners allege that Serebrin discloses in Ground 3 (Pet., 80-87), Petitioners’ discussion about Serebrin’s caches and registers does not fix the issues with Borchers. For Ground 3, Petitioners allege that “Serebrin provides an example of using caches or registers to store state information.” Pet., 84. Petitioners then conclude that “a POSITA would have found it obvious for Borchers’ interrupt manager to similarly include cache/registers for storing data.” Pet., 86. When it comes to the deficiencies of Borchers not having an “interrupt manager” / “interrupt manager controller” to “alert” the host processor of an unprocessed completion status in the completion queue, Petitioners do not cite Serebrin. *E.g.*, Pet., 30 (relying on Borchers’ interrupt processor 124), 31 (relying on Borchers’ interrupt signal output from interrupt send logic 286). Serebrin therefore does not cure the fatal flaws identified above for Ground 1.

Petitioners’ motivation-to-combine analysis in Ground 3 is also flawed. “[A] clear, evidence-supported account of the contemplated workings of the combination is a prerequisite to adequately explaining and supporting a conclusion that a relevant skilled artisan would have been motivated to make the combination and reasonably expect success in doing so.” *Personal Web Techs., LLC v. Apple*,

848 F.3d 987, 994 (Fed. Cir. 2017). But here, Petitioners never explained *why* or *how* a POSA would have combined Borchers and Serebrin.

Petitioners never explained *why* a POSA would have used cache or registers to store information sent to Borchers' interrupt processor 124. *See* Pet., 83-87. For Ground 3, Petitioners show that Borchers' interrupt processor 124 receives signals that Petitioners label response queue status information. Pet., 84. Petitioners also show that Borchers' interrupt processor 124 has counters that Petitioners label interrupt status information. Pet., 84.



Pet., 84 (annotating EX1005, FIG. 2).

Serebrin discloses a substantially different system—a virtual machine with two host processors 30A and 30B, as shown below. EX1008, ¶51. Borchers, in

contrast, is for communicating commands between a single host and a separate flash memory. EX1005, Abstract, FIG. 1A. As shown below, Serebrin's host 20 includes processors 30A/33B, Advanced Programmable Interrupt Controllers (APICs) 32A/32B, a memory controller (MC) 42, a memory interface circuit (MIF) 46, and memory 56. EX1008, ¶51. Memory 56 stores, among other things, the guest's Advanced Programmable Interrupt Controller (gAPIC) state information. EX1008, ¶53. This state information is not the state of a response queue. *See id.*

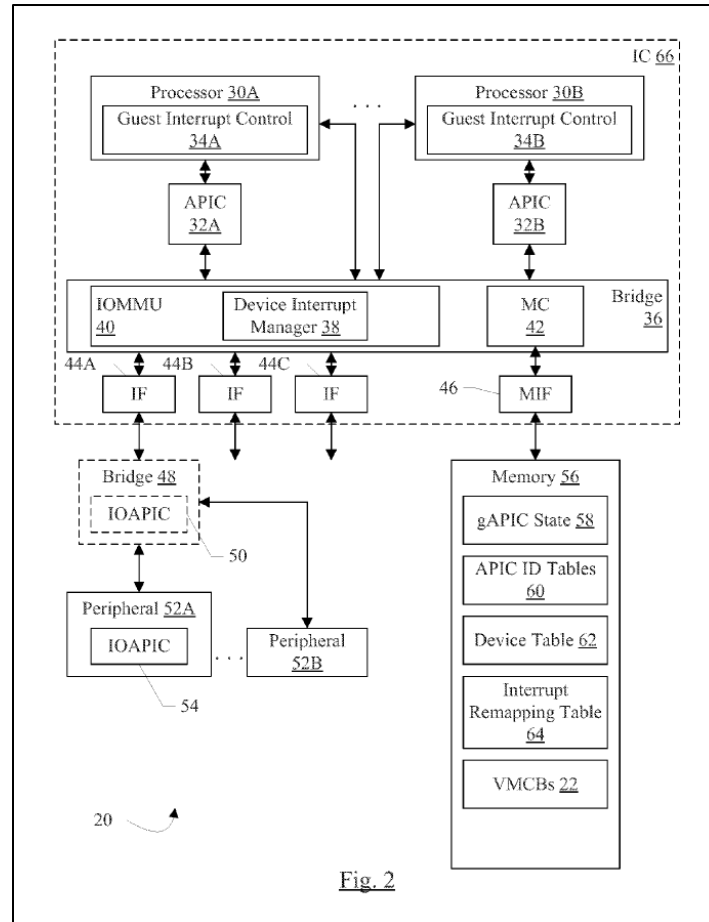


Fig. 2

EX1008, FIG. 2.

After pointing to Serebrin's memory 56, which stores gAPIC state

information, Petitioners conclude that “a POSITA would have found it obvious for Borchers’ interrupt manager to similarly include cache/registers for storing state data.” Pet., 86. As support, Petitioners cite Dr. Min’s declaration (§227). Dr. Min’s testimony is identical to the Petition, is conclusory, does not cite any evidence, and should be given no weight. *See Xerox*, IPR2022-00624, Paper 12 at 15.

Contrary to Dr. Min’s conclusory testimony, there is no reason for Borchers to include additional cache/registers for storing the alleged response queue status information and alleged interrupt status information because Borchers already operates as intended. *Supra* Section III.C. In Borchers’ logic circuit, the signals input to interrupt processor 124 are used to increment counters, e.g., via pulses. *See* EX1005, ¶¶60-65. Why Petitioners believe that storing these pulses (or some other information) would be beneficial is unclear. There must be “*a reason* that would have prompted a person of ordinary skill in the relevant field to combine the elements in the way the claimed new invention does.” *KSR*, 550 U.S. at 418 (emphasis added). Neither Petitioners nor Dr. Min provided such a reason.

Petitioners also never explained *how* a POSA would have combined Borchers and Serebrin. They never explained how Borchers’ interrupt manager would include a cache/registers, where this cache/registers would be located, how the input received via a series of signals would be captured in the cache/registers, or how the information in the counters would get transferred and stored.

Petitioners' conclusory leap to obviousness, without explaining how the combination would actually work, is impermissible. They provided no reason *why* a POSA would have added additional cache/registers. Nor did they explain *how* a POSA would have implemented the cache/registers.

For these reasons and the reasons explained above for Ground 1, Petitioners do not have a reasonable likelihood of success under Ground 3.

VI. THE BOARD SHOULD NOT CREDIT DR. MIN'S OPINIONS.

The Board should not credit Dr. Min's opinions because they are conclusory and merely repeat Petitioners' arguments. "[C]onclusory expert testimony is inadequate to support an obviousness determination" *TQ Delta, LLC v. Cisco Systems, Inc.*, 942 F.3d 1352, 1359 (Fed. Cir. 2019); *see also Magnum Oil Tools*, 829 F.3d at 1380 ("To satisfy its burden of proving obviousness, a petitioner cannot employ mere conclusory statements."). Accordingly, the Board does not credit declaration testimony that "merely repeats, verbatim, the conclusory assertion for which it is offered to support." *Xerox*, IPR2022-00624, Paper 9 at 15-16 (emphasis omitted); *see also id.* at 15 ("[T]he cited declaration testimony is conclusory and unsupported, adds little to the conclusory assertion for which it is offered to support, and is entitled to little weight."); 37 C.F.R. § 42.65(a).

Key opinions in Dr. Min's declaration are conclusory and merely repeat the Petition's equally conclusory arguments, with the primary difference between the

declaration and Petition being that Dr. Min block quotes the references while the Petition provides in-line quotes. For example, Dr. Min's opinions about the claimed "interrupt manager" / "interrupt manager controller" to "alert" the host processing unit of an unprocessed completion status tracks Petitioners' arguments nearly verbatim. *Compare* Pet., 30-31, with EX1003, ¶¶86, 88. The same is true of Dr. Min's opinions for the motivation to combine Borchers with the PCI Standards and Serebrin. *Compare* Pet., 77-80, 83-87, with EX1003, ¶¶207-213, 223-29.

At bottom, Dr. Min's declaration is the quintessential cut-and-paste copy of conclusory attorney argument that *TQ Delta*, *Magnum Oil Tools*, and *Xerox* instruct should be given no weight.

VII. CONCLUSION

The Petition fails to show a reasonable likelihood that the challenged claims are unpatentable in view of the cited art. The Board should deny institution.

Respectfully submitted,

STERNE, KESSLER, GOLDSTEIN & FOX PLLC

/Richard M. Bemben/

Richard M. Bemben
Registration No. 68,658
Attorney for Patent Owner

Date: May 27, 2025

1101 K Street, NW, 10th Floor
Washington, DC 20005
(202) 371-2600

CERTIFICATE OF WORD COUNT (37 C.F.R. § 42.24(d))

1. This Patent Owner Preliminary Response complies with the type-volume limitation of 14,000 words, comprising 8,306 words, excluding the parts exempted by 37 C.F.R. § 42.24(a)(1).

2. This Patent Owner Preliminary Response complies with the general format requirements of 37 C.F.R. § 42.6(a) and has been prepared using Microsoft® Word 2016 in 14-point Times New Roman font.

Respectfully submitted,

STERNE, KESSLER, GOLDSTEIN & FOX PLLC

/Richard M. Bemben/

Richard M. Bemben
Registration No. 68,658
Attorney for Patent Owner

Date: May 27, 2025

1101 K Street, NW, 10th Floor
Washington, DC 20005
(202) 371-2600

CERTIFICATE OF SERVICE (37 C.F.R. § 42.6(e))

I certify that the above-captioned **PATENT OWNER PRELIMINARY
RESPONSE UNDER 37 C.F.R. § 42.107(a)** and associated Exhibits 2025-2026
were served in their entireties on May 27, 2025, upon the following parties via
electronic mail:

David O'Dell (Lead Counsel)
Hong Shi (Back-up Counsel)
Calmann J. Clements (Back-up Counsel)
HAYNES AND BOONE, LLP
david.odell.ipr@haynesboone.com
hong.shi.ipr@haynesboone.com
calmann.clements.ipr@haynesboone.com

Respectfully submitted,

STERNE, KESSLER, GOLDSTEIN & FOX PLLC

/Richard M. Bemben/

Richard M. Bemben
Registration No. 68,658
Attorney for Patent Owner

Date: May 27, 2025

1101 K Street, NW, 10th Floor
Washington, DC 20005
(202) 371-2600