

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

ADVANCED MICRO DEVICES, INC. and PENSANDO SYSTEMS, INC.,

Petitioners,

v.

CONCURRENT VENTURES, LLC and XTREAMEDGE, INC.,

Patent Owners.

Case IPR2025-00478

U.S. Patent 8,924,596

PATENT OWNERS' RESPONSE

TABLE OF CONTENTS

I.	INTRODUCTION	1
II.	BACKGROUND	4
A.	Overview of the '596 Patent.....	4
B.	Overview of Petitioners' Prior Art.....	9
1.	U.S. Patent No. 8,559,439 ("Dongare") (EX1005)	9
2.	U.S. Patent No. 8,051,227 ("Gewirtz") (EX1006)	12
3.	U.S. Patent No. 7,398,528 ("Khawand") (EX1008).....	13
4.	U.S. Patent No. 5,805,930 ("Rosenthal") (EX1009)	15
5.	U.S. Patent No. 6,810,462 ("Matsunami")	16
6.	U.S. Patent No. 8,028,284 ("Ohkawa") (EX1010).....	16
C.	Alleged Unpatentability Grounds.....	16
III.	PROSECUTION HISTORY	17
IV.	PRIORITY DATE OF THE '596 PATENT	17
V.	LEVEL OF ORDINARY SKILL	17
VI.	RELATED MATTERS	18
VII.	CLAIM CONSTRUCTION	18
VIII.	PETITIONER HAS NOT PROVEN THAT ANY CHALLENGED CLAIM IS UNPATENTABLE	19
A.	Petitioners' Arguments Stem from a Fundamental Misunderstanding of the Problem Solved by the '596 Patent.....	19
B.	Petitioners Fail to Prove their Combination of Dongare and Gewirtz (Ground 1 and 2) Discloses or Renders Obvious Claims 1, 2, 5–8, 11–14, 17, and 18.....	20
1.	Claim 1	20

a.	Dongare and Gewirtz Do Not Render Obvious the Second Processing Element Accessing and Utilizing the Reservation Register, as Required by Limitations 1[e], 1[g], 1[h], and 1[i].	21
b.	Dongare and Gewirtz Do Not Render Obvious a Second Processing Element Sending Notifications to a First Processing Element to Issue Commands, as Required by Limitations 1[i] and 1[j].	29
c.	Dongare Does Not Render Obvious an “Input Queue Implemented in Hardware” As Required By Limitation 1[a].	34
2.	Independent Claims 7 and 13.....	37
a.	Limitations 7[a], 7[e], 7[h]–7[i] and 13[a], 13[e], 13[h]–13[i] Fail for the same reasons as equivalent in claim 1.	37
b.	Dongare and Gewirtz Did Not Render Obvious a Second Processing Element Receiving a Command from the First Processing Element and Issuing a Command to the Input Queue on Behalf of the First Processing Element, as Required by Limitations 7[g]/[j] and 13[g]/[j].	37
c.	Dongare and Gewirtz Did Not Disclose or Render Obvious Limitation 13[k].	42
3.	Dependant claims.....	43
a.	Dongare and Gewirtz Did Not Disclose or Render Obvious Claims 5, 11, and 17.	43
b.	Dongare and Gewirtz Did Not Render Obvious Claims 6, 12, and 18.	44
4.	Ground 2 Does Not Address The Problems In Ground 1.....	46
C.	Petitioners Fail to Prove their Combination of Khawand, Rosenthal, Ohkawa, and Matsunami (Grounds 3, 4, and 5) Discloses or Renders Obvious Claims 1–4, 6–10, 12–16, 18.	46

1.	Claim 1	46
a.	The References Teach Away From Petitioners’ Proposed Combination.	46
i.	Rosenthal Teaches Away From Petitioners’ Proposed Combination for Limitations 1[e]– 1[j].	46
ii.	Khawand Teaches Away From Petitioners’ Proposed Combination For Limitation 1[h].	49
b.	In Any Event, Petitioners’ Proposed Combination Does Not Render Obvious Claim 1.	53
i.	Limitations 1[i]–1[j] Are Not Obvious in Light of Khawand and Rosenthal.	54
ii.	Limitations 1[c] and 1[j] Are Not Obvious in Light of Khawand and Rosenthal.	58
2.	Claims 7 and 13.	59
a.	Limitations 7[e]–7[i] and 7[k], and 13[e]–13[i] Fail for the same reasons as equivalent in claim 1.	59
b.	Khawand and Rosenthal Did Not Disclose or Render Obvious Limitations 7[g]/[j] and 13[g]/[j].	60
c.	Khawand and Rosenthal Did Not Disclose or Render Obvious Limitation 13[k].	62
3.	Dependant claims.	65
a.	Khawand and Rosenthal Did Not Disclose or Render Obvious Claims 6 and 12.	65
b.	Petitioners Do Not Even Allege Claim 18 Would Be Obvious Over Khawand and Rosenthal.	65
4.	Ground 4 Does Not Address The Problems In Ground 3.	66
5.	Ground 5 Does Not Address The Problems In Ground 3.	66

IX. CONCLUSION.....66

CLAIM APPENDIX.....2

PATENT OWNERS' EXHIBIT LIST

Exhibit	Description
2001	Text Order Denying AMD/Pensando's Motion to Stay Case (Judge Robert Pitman) in <i>Concurrent Ventures, LLC and XstreamEdge, Inc. v. Advanced Micro Devices, Inc. and Pensando Systems, Inc.</i> , No. 1-24-cv-00335 (W.D. Tex. 2024)
2002	First Amended District Court Scheduling Order (Dkt. 77) in <i>Concurrent Ventures, LLC and XstreamEdge, Inc. v. Advanced Micro Devices, Inc. and Pensando Systems, Inc.</i> , No. 1-24-cv- 00335 (W.D. Tex. 2024)
2003	Webpage printouts of Docket Navigator Time to Trial regarding the District Court for the Western District of Texas and Judge Alan D. Albright
2004	Webpage printout of "AMD Reports Fourth Quarter and Full Year 2024 Financial Results" dated March 4, 2025 (https://ir.amd.com/news-events/press-releases/detail/1236/amd-reports-fourth-quarter-and-full-year-2024-financial-results)
2005	AMD/Pensando's Invalidity Contentions (Cover Pleading) in <i>Concurrent Ventures, LLC and XstreamEdge, Inc. v. Advanced Micro Devices, Inc. and Pensando Systems, Inc.</i> , No. 1-24-cv-00335 (W.D. Tex. 2024)
2006	AMD/Pensando's Opening Claim Construction Brief in <i>Concurrent Ventures, LLC and XstreamEdge, Inc. v. Advanced Micro Devices, Inc. and Pensando Systems, Inc.</i> , No. 1-24-cv-00335 (W.D. Tex. 2024)
2007	Declaration of Dr. R. Jacob Baker submitted by AMC/Pensando in support of AMD/Pensando's Opening Claim Construction Brief in <i>Concurrent Ventures, LLC and XstreamEdge, Inc. v. Advanced Micro Devices, Inc. and Pensando Systems, Inc.</i> , No. 1-24-cv-00335 (W.D. Tex. 2024)
2008	Declaration of Darrell Long submitted by AMC/Pensando in support of AMD/Pensando's Opening Claim Construction Brief in <i>Concurrent Ventures, LLC and XstreamEdge, Inc. v. Advanced Micro Devices, Inc. and</i>

	<i>Pensando Systems, Inc.</i> , No. 1-24-cv-00335 (W.D. Tex. 2024)
2009	Declaration of Bill Lin submitted by AMC/Pensando in support of AMD/Pensando’s Opening Claim Construction Brief in <i>Concurrent Ventures, LLC and XstreamEdge, Inc. v. Advanced Micro Devices, Inc. and Pensando Systems, Inc.</i> , No. 1-24-cv-00335 (W.D. Tex. 2024)
2010	Declaration of Kevin Jeffay submitted by AMC/Pensando in support of AMD/Pensando’s Opening Claim Construction Brief in <i>Concurrent Ventures, LLC and XstreamEdge, Inc. v. Advanced Micro Devices, Inc. and Pensando Systems, Inc.</i> , No. 1-24-cv-00335 (W.D. Tex. 2024)
2011	CV/XE’s Responsive Claim Construction Brief in <i>Concurrent Ventures, LLC and XstreamEdge, Inc. v. Advanced Micro Devices, Inc. and Pensando Systems, Inc.</i> , No. 1-24-cv-00335 (W.D. Tex. 2024)
2012	Declaration of Dr. Paul Min submitted by CV/XE in support of CV/XE’s Responsive Claim Construction Brief in <i>Concurrent Ventures, LLC and XstreamEdge, Inc. v. Advanced Micro Devices, Inc. and Pensando Systems, Inc.</i> , No. 1-24-cv-00335 (W.D. Tex. 2024)
2013	AMD/Pensando’s Reply Claim Construction Brief in <i>Concurrent Ventures, LLC and XstreamEdge, Inc. v. Advanced Micro Devices, Inc. and Pensando Systems, Inc.</i> , No. 1-24-cv-00335 (W.D. Tex. 2024)
2014	CV/XE’s Objection to the Report and Recommendation of the U.S. Magistrate Judge in <i>Concurrent Ventures, LLC and XstreamEdge, Inc. v. Advanced Micro Devices, Inc. and Pensando Systems, Inc.</i> , No. 1-24-cv-00335 (W.D. Tex. 2024)
2015	AMD/Pensando’s Motion for Reconsideration of Venue Order in <i>Concurrent Ventures, LLC and XstreamEdge, Inc. v. Advanced Micro Devices, Inc. and Pensando Systems, Inc.</i> , No. 1-24-cv-00335 (W.D. Tex. 2024)
2016	Order denying AMD/Pensando’s Motion for Reconsideration of Venue Order in <i>Concurrent Ventures, LLC and XstreamEdge, Inc. v. Advanced Micro Devices, Inc. and Pensando Systems, Inc.</i> , No. 1-24-cv-00335 (W.D. Tex. 2024)

2017	September 11, 2025 Scheduling Order in <i>Concurrent Ventures, LLC and XtreamEdge, Inc. v. Advanced Micro Devices, Inc. and Pensando Systems, Inc.</i> , No. 1-24-cv-00335 (W.D. Tex. 2024)
2018	Petitioners' Request For Rehearing Of Decision Denying Institution Of <i>Inter Partes</i> Review in IPR2025-00223, Paper 10.
2019	Declaration of Zachary L. Jacobs in Support of Motion for Admission <i>Pro Hac Vice</i>

NEW EXHIBITS

New Exhibit	Description
2020	Declaration of Dr. Paul S. Min in Support of Patent Owners' Response
2021	Curriculum Vitae of Dr. Paul S. Min
2022	Deposition of Dr. Samuel Russ in IPR2025-00478, conducted December 19, 2025
2023	July 31, 2025 Memorandum

Concurrent Ventures, LLC and XstreamEdge, Inc. (“Patent Owners”) respectfully submit this Patent Owners’ Response (“POR”) to the Petition for *Inter Partes* Review (“IPR”) of U.S. Patent No. 8,924,596 (“the ’596 patent”) filed by Advanced Micro Devices, Inc. and Pensando Systems, Inc. (“Petitioners”), and to the Board’s October 9, 2025 Institution Decision (“ID”) (Paper 11).

I. INTRODUCTION

The Petition fails to render obvious the inventions of the ’596 patent. Petitioners rely solely on obviousness arguments, but, for each ground, there are limitations wholesale missing. Petitioners were unable to find any reference that teaches or suggests the missing limitations. Implicitly recognizing this, Petitioners seek to rewrite their references, but provide no more support than their say so—no teaching or suggestion in the art, and no reasoned or supported expert opinion. Petitioners ask the Board to find that the ’596 patent was obvious without *any* suggestion in the art of its core innovations—the definition of an improper hindsight analysis.

Each independent claim of the ’596 patent requires a first processing element (“1PE”) which issues commands to be processed. But, importantly, the 1PE does not itself determine *when* it can issue the command—that responsibility is offloaded to a second processing element (“2PE”). The ’596 patent further requires a “reservation register” (“RR”) that stores a value indicative of the space available in the input

queue. The 2PE (not the 1PE) checks this value and determines whether a command may be issued. In claim 1, the 2PE next notifies the 1PE when there is space for the particular command, and the 1PE issues it as instructed. In claims 7 and 13, the 2PE receives the command from the 1PE and issues the command on its behalf. But regardless of these differences, each claim captures this core innovative division of responsibilities: the processor that initiates the commands (the 1PE) is *not* the same processor that accesses the RR and determines when the command should be issued (the 2PE). This division is entirely missing from the Petition’s combinations.

For Grounds 1 and 2, Petitioners rely on Dongare’s “Embedded Processor 760” (“EP 760”) as the claimed 2PE. But Dongare’s EP 760 is a processor that only adds tags to commands *after* they have been added to an input queue and then reorders them into the correct sequences after they have been processed. Crucially, the Host 710 (the claimed 1PE) issues the commands *before* EP 760 does anything—there is nothing in either Dongare or Petitioners’ other reference, Gewirtz, to even suggest that EP 760 should decide when the Host issues commands; Petitioners make it up out of whole cloth.

It is similar for Grounds 3–5. For those grounds, Petitioners rely on Khawand’s “Inter-Processor Manager 308” (“IPM 308”) as the 2PE. But again, this alleged 2PE does not perform the functions required of the 2PE by the claims. In particular, IPM 308 evaluates the queue to determine whether it can accept new

service applications in general. It does **not** determine whether there is sufficient space available in the queue for a particular command. Additionally, IPM 308 does not notify Processor 1 (the alleged 1PE in Khawand) to issue a particular command—which is unsurprising given that IPM 308 never evaluates whether there is space for a particular command. Instead, IPM 308 merely *allows* Processor 1 to issue any commands it might want.

Worse yet, there is no suggestion in either Khawand or Petitioners’ other reference, Rosenthal, that Processor 1 receives a response regarding commands it has issued. Nor would such an arrangement make sense in Khawand. Processor 1 and Processor 2 (the processor executing the command) are parallel processors. Once Processor 1 has delegated a command to Processor 2, there is no reason for anything related to that command to go back to Processor 1. And Petitioners’ arguments for claims 7 and 13—which require the IPM to write commands *on behalf of* Processor 1—are even more off base.

Petitioners attempt to make up for these glaring flaws by waiving their hands and alleging the ’596 patent’s core innovations would simply be “obvious.” Petitioners’ expert, Dr. Samuel Russ, likewise oversimplifies the claims as only requiring the use of the RR without any attention to ***which processor*** is using the RR. Petitioners and Dr. Russ fail to identify a combination of art that renders obvious the claims and so resort to hindsight to manufacture limitations that are otherwise

missing. The result is that the Petition does not demonstrate by a preponderance of the evidence that any claim is unpatentable under any of the grounds.

II. BACKGROUND

A. Overview of the '596 Patent

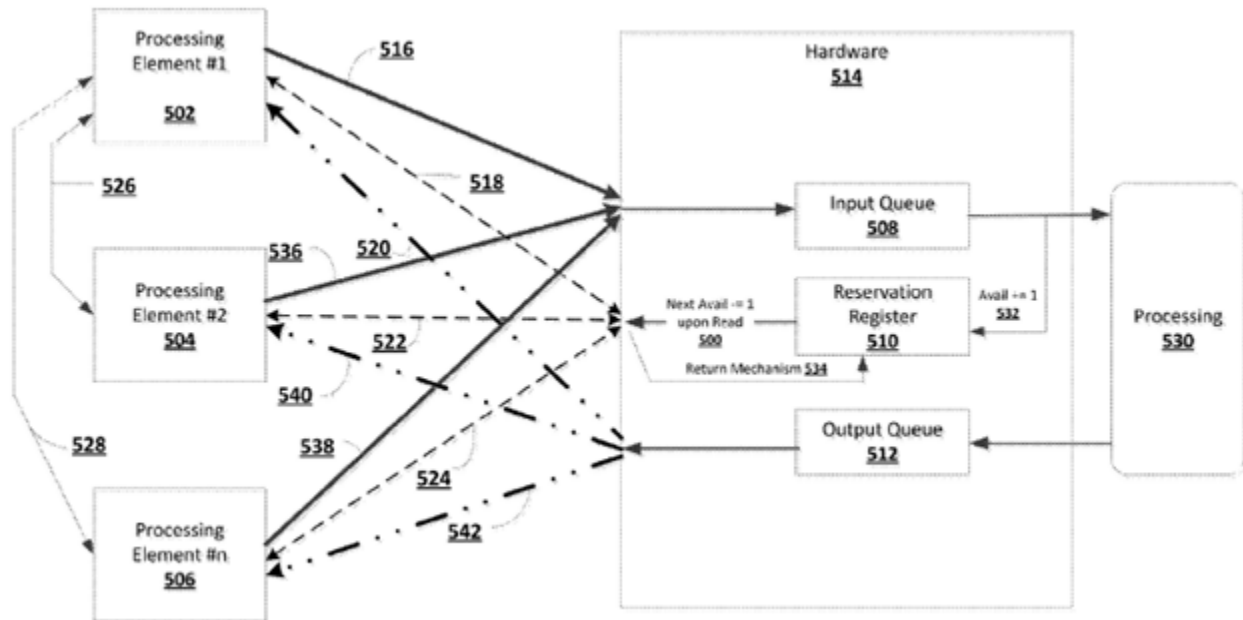
The inventors of the '596 patent identified a problem in the art of computer processing relating to “load distribution.” EX1001, 1:9–10. “In the prior art, the processing load (of a processing element) to determine what command to issue [wa]s relatively high.” *Id.*, 1:15–16. Before the '596 patent, the task of using a RR to determine what command to issue was performed by either: (1) the processing element that executes the command; or (2) the processing element that issues the command. The inventors of the '596 patent discovered that it was preferable for that role to be performed by a different processing element, distinct from both the executing and issuing processors so the determination could be performed quickly and efficiently. *Id.*, 5:3–7:58.

But this innovative division of labor itself created a new problem with “synchronization . . . amongst the multiplicity of processing elements.” *Id.*, 1:17–25. That is, the multiple processors involved in issuing a command had to be properly synchronized. *Id.* Prior art systems attempted to address this problem by “using software techniques, such as semaphores and mutexes to a shared available space count variable in memory.” *Id.*, 1:25–27. These software techniques, however, were

typically managed by the issuing processor, requiring “additional processing time” and thus were “non-optimal,” particularly because “there may be a need to keep the time from command issue to execution minimized.” *Id.*, 1:27–32. Software techniques that operate on a shared “available space” variable necessarily require memory, which meant that the IPE had less processing power to accomplish other tasks. *Id.*

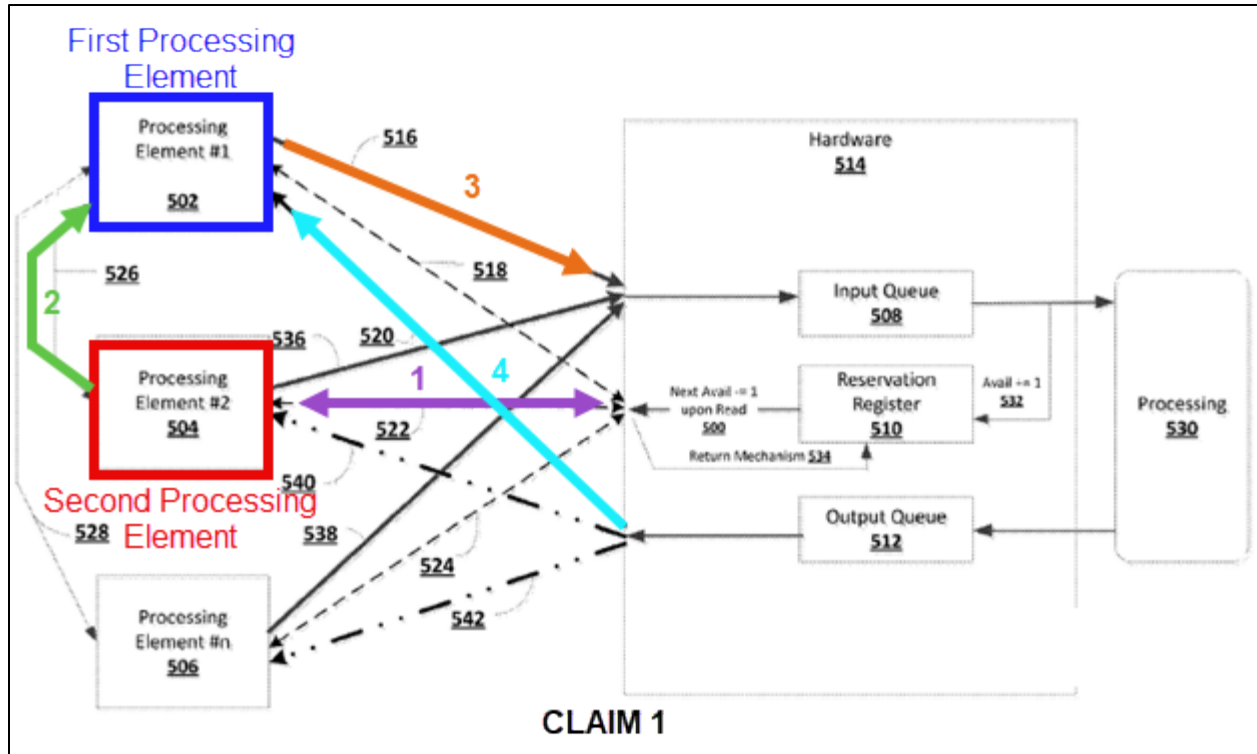
To address this problem, the inventors of the ’596 patent developed “a system for dividing and synchronizing a processing task across a plurality of processing elements” that utilized input and output queues and a RR implemented in hardware, among other claim elements. *Id.*, 2:13–14. The invention uses a hardware RR that tracks available queue space and automatically decrements when read by a processing element. *Id.*, 3:43–47. The hardware RR guarantees mutual exclusion without software locks because a processor need only read the RR to determine if there is sufficient space to add a command to a queue, and it may automatically decrement on read, thereby eliminating race conditions between two processors competing for limited space in a queue. *Id.*, 3:59–62, 3:23–29. The result is an invention that requires at least two processing elements that share access to the RR and the input/output queues. *Id.*, 1:15–16.

Figure 5 of the '596 patent, reproduced below, depicts the high-level relationship between the 1PE, 2PE, and hardware RR tracking the space available in an input queue:

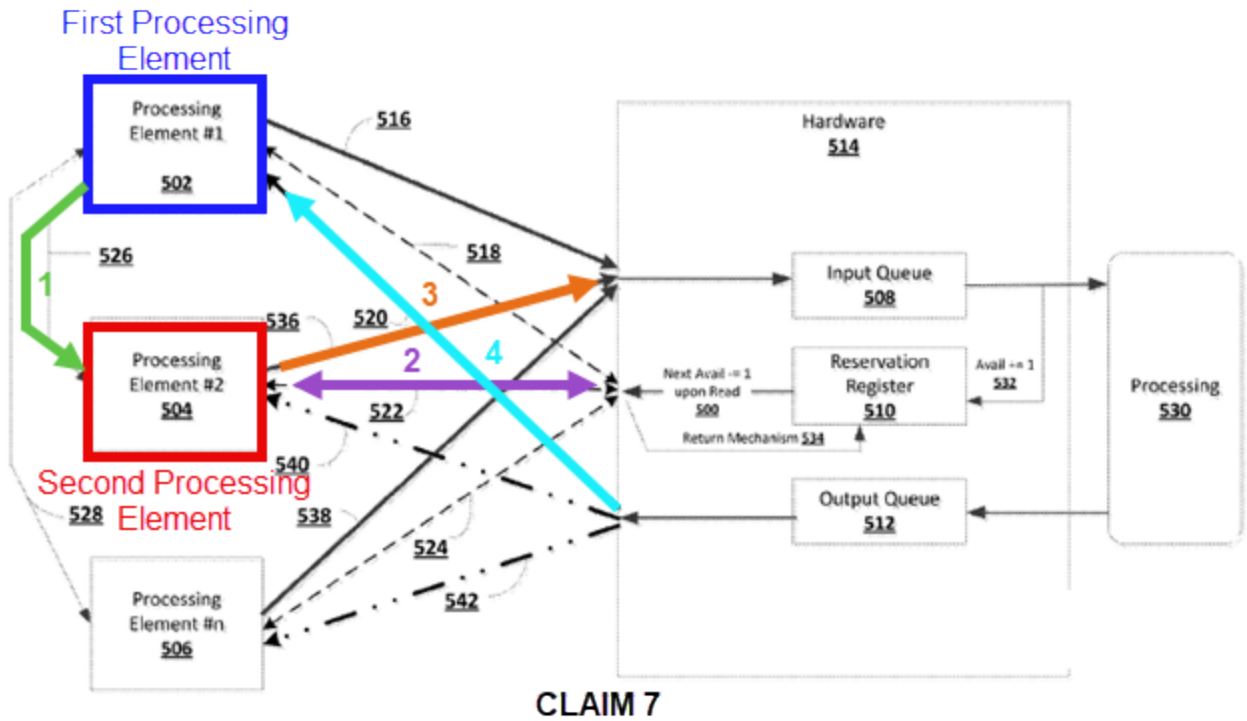


EX1001, FIG. 5.

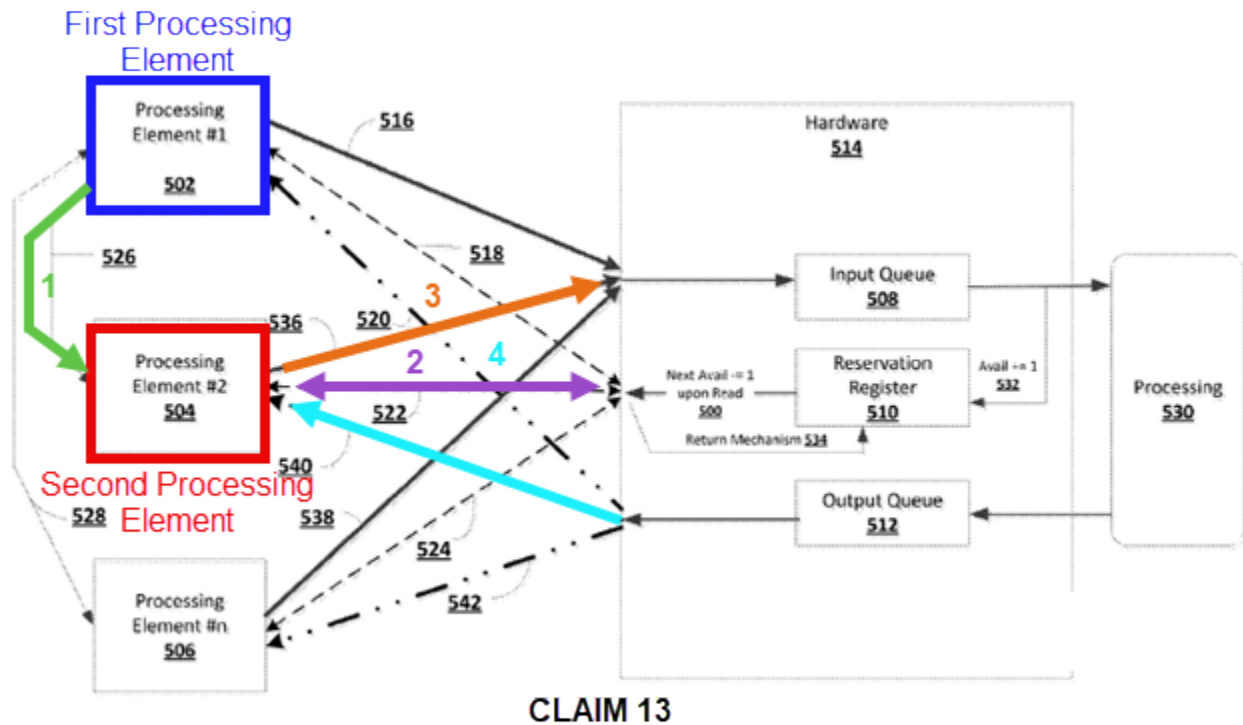
Figure 5 demonstrates the different independent claims. In **claim 1**: First, the 2PE (in red) reads the RR (communication in purple) and determines whether there is sufficient space in the input queue for a particular command. Second, if there is sufficient space, the 2PE notifies the 1PE (in blue) to issue the command (communication in green). Third, the 1PE issues the command to the input queue (communication in orange). Fourth, the output queue returns a response to the 1PE (communication in teal).



In **claim 7**: First, the 2PE (in red) receives a command from the 1PE (in blue) (communication in green). Second, the 2PE reads the RR (communication in purple) and determines whether there is sufficient space available in the input queue for a particular command. Third, if there is sufficient space, the 2PE sends the command to the input queue on behalf of the 1PE (communication in orange). Fourth, the output queue returns a response to the 1PE (communication in teal).



In claim 13: the same first three steps are the same as in claim 7, but with a different fourth step. As shown below, fourth, the output queue returns a response to the **2PE** (communication in teal) instead of the 1PE.



In each of the claims—and unlike in the prior art—the 1PE is in direct communication with the 2PE, the 2PE is the processor responsible for checking the available space in the RR, and the 2PE notifies the 1PE to issue the command or issues it on behalf of the 1PE. *Id.*

B. Overview of Petitioners' Prior Art

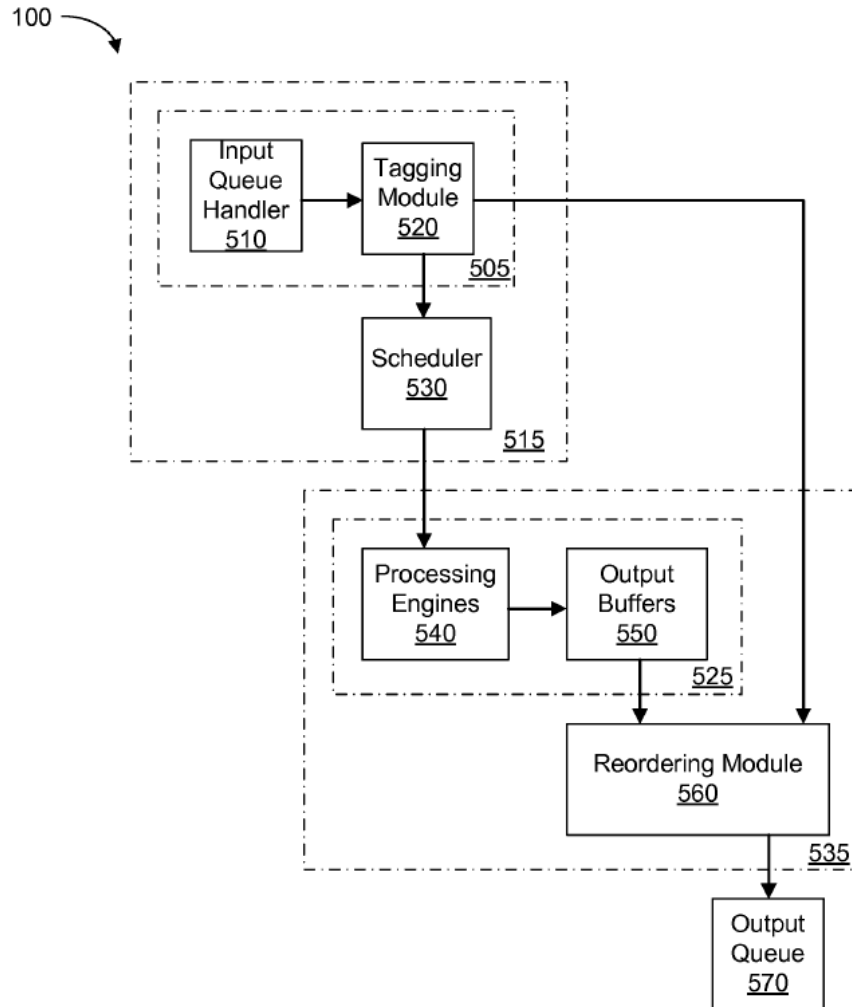
1. U.S. Patent No. 8,559,439 (“Dongare”) (EX1005)

Dongare is U.S. Patent No. 8,559,439 entitled “Method and Apparatus for Queue Ordering in a Multi-Engine Processing System.” Dongare purports to disclose “[a] method and apparatus for queue-ordering commands in multi-engines, multi-queues and/or multi-flows environment.” EX1005, Abstract. Petitioners use

Dongare as the primary reference for Grounds 1 and 2, namely for the “1PE” and “2PE” limitations.

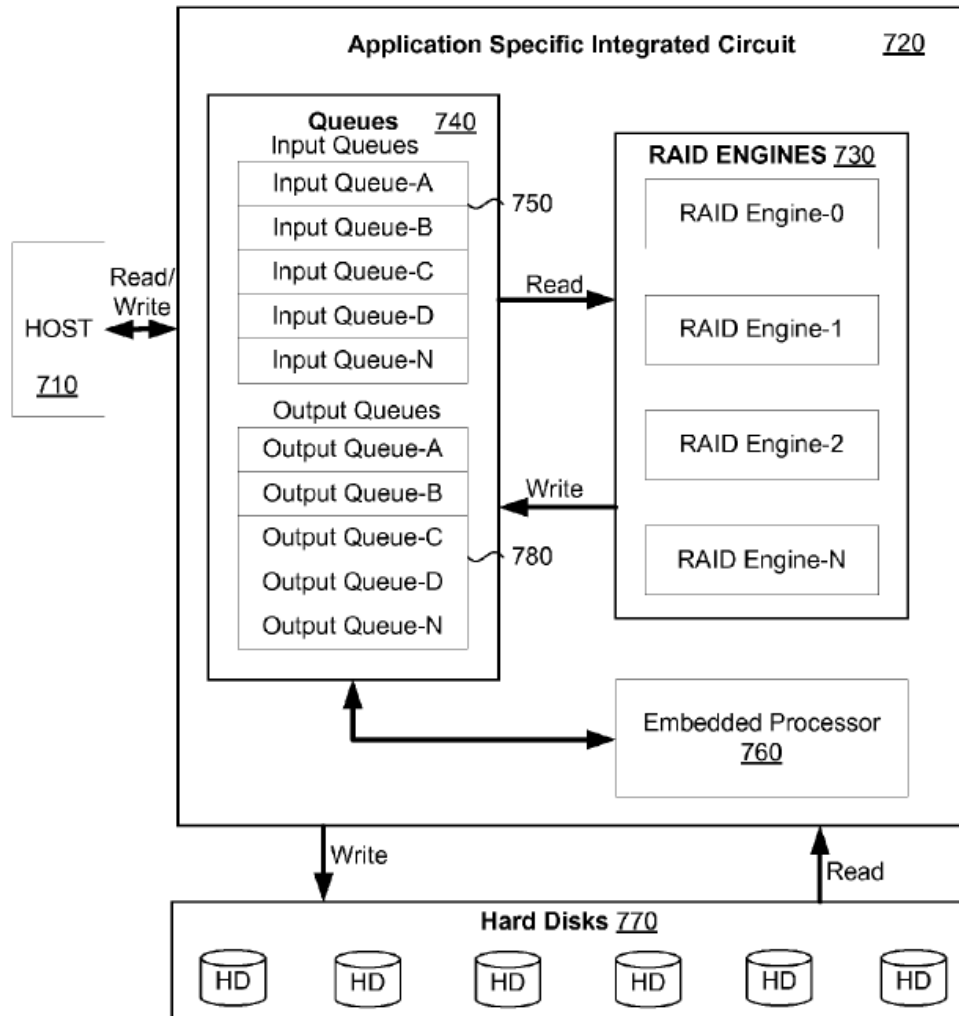
When multiple processors are used to process commands or packet segments from the same input queue, Dongare explains that it is beneficial to process the commands or packet segments out-of-order based on processing engine availability. EX1005, 1:25–32, 5:62–65. But because segments from the same input stream may be dispatched to different engines, the resulting output may be disordered relative to the input. *Id.*, 1:25–32. To address this, the on-chip processing element assigns each command or packet segment an “identity tag” that encodes its stream ID and original sequence position, which is used by a reordering module after execution to reorder the commands into the original sequence. *Id.*, 5:42–53. Dongare thus concerns how multiple processors can be effectively used to process commands while still maintaining the order of those commands—not ensuring that input queues are available and not overfilled, as in the ’596 patent. EX2020, ¶66.

Dongare’s figures demonstrate this limited focus. For example, Figure 5 depicts “System 100”—the entirety of Dongare’s disclosed system—which includes an “input queue handler” that reads commands *after* they have been written into the queue. EX2020, ¶68. This illustrates that Dongare’s system is only concerned with what occurs *after* commands are added to an input queue. EX1005, 10:50–11:13 (showing the pathway after being picked up by the input queue handler).



EX1005, FIG. 5, 10:50–11:13.

Figure 7 of Dongare also underscores the narrow invention. EP 760 (*i.e.*, the on-chip processor responsible for tagging the commands and input segments) sits ***downstream*** from the input queues ***inside*** the ASIC. That is because EP 760—which Petitioners claim is the 2PE—only tags, portions, and reorders commands, and does not determine when they should be issued in the first place. EX2020, ¶69.



EX1005, FIG. 7.

2. U.S. Patent No. 8,051,227 (“Gewirtz”) (EX1006)

Gewirtz is entitled “Programmable Queue Structures for Multiprocessors.” EX1006, Abstract. Petitioners rely on Gewirtz primarily for its teachings regarding a “Queue State Register.” Pet. 19–27. Gewirtz purports to disclose a processing system where a host processor manages input queues using a queue-state register which tracks the available space in an input queue. EX1006, Abstract. Petitioners

use Gewirtz as a secondary reference in Grounds 1 and 2, namely for the “RR” limitation required in each independent claim.

Gewirtz describes a processing system where a credit counter is used for flow control. EX1006, 2:59–62. In particular, the counter tracks the number of free entries and returns a pass/fail result to an agent attempting to reserve space before writing a command or packet segment. *Id.*, 4:61–67. When the agent reads the credit counter, the credit counter decrements the count on a successful reservation and increments when an entry is executed by an engine. *Id.*, 5:3–6. Notably, the credit counter does not provide the actual number of free entries to a processor, instead providing a “pass” or “fail.” *Id.*, 4:61–67.

Petitioners do not rely on Gewirtz to teach or suggest the divided architecture in which it is the 2PE, not the 1PE, that determines when a command should be issued. Pet., 19–24. That makes sense, as Gewirtz teaches a system where the processor which reads the queue-state register is the *same processor* that writes the command into the input queue. EX1006, 4:56–61, 1:39–44.

3. U.S. Patent No. 7,398,528 (“Khawand”) (EX1008)

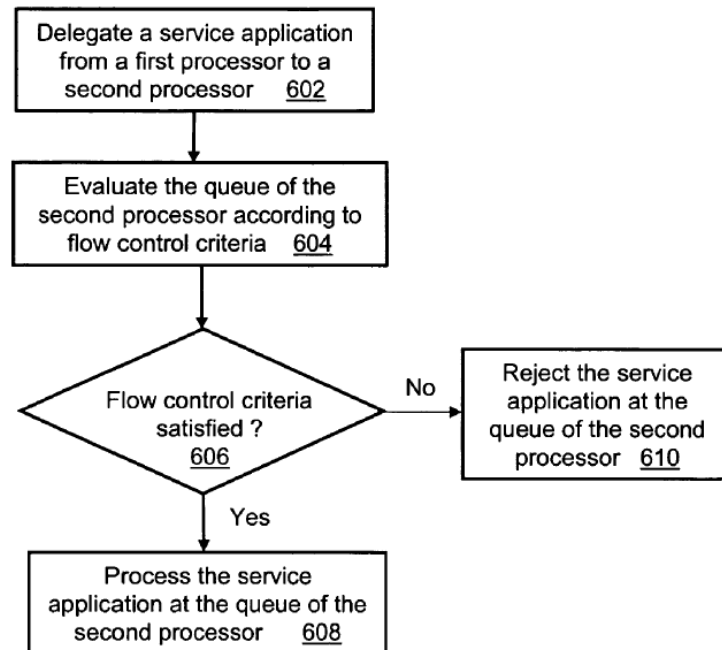
Khawand is entitled “Method And System For Efficient Multiprocessor Processing In A Mobile Wireless Communication Device.” Petitioners contend that Khawand teaches “a multiprocessor architecture that improves efficiency using

delegation, queues, and flow control.” Pet., 44. Khawand is the primary reference for Grounds 3 and 4.

Khawand describes a processing system in which “Processor 1” and “Processor 2” are both parallel processors for executing “service applications.” In Khawand, when Processor 1 is set to process a service application or task (*i.e.*, command) it can instead delegate the entire service application or task to “Processor 2” for execution (*i.e.*, it can write the service application or task to Processor 2’s queue). EX1008, 1:35–48, FIG. 3. Processor 1 initiates this delegation on its own when its processing efficiency begins to suffer. *Id.*, 3:62–4:5. However, if Processor 2’s queue fills up above a “capacity threshold,” the command is rejected and a “suspend command” issued through the “Inter-Processor Manager” to Processor 1 to “suspend” any more delegations. *Id.*, 4:60–67. Processor 2 may then send a “resume” notification through IPM 308 if Processor 2’s queue goes below the “capacity threshold.” *Id.*, 5:18–29.

Khawand’s model for synchronizing multiple processors differs substantially from the ’596 patent, where multiple processing elements jointly participate in issuing commands to a shared hardware queue using a RR. EX2020, ¶79. Khawand’s IPM 308 at most notifies Processor 1 that it may resume delegating service applications again based on whether Processor 2’s queue is generally accepting new

service applications. IPM 308 does not determine if there is enough space in the queue for a particular command, and then notify Processor 1 to issue that command.



Id., FIG. 6.

4. U.S. Patent No. 5,805,930 (“Rosenthal”) (EX1009)

Rosenthal is entitled “System for FIFO Informing the Availability of Stages to Store Commands Which Include Data and Virtual Address Sent Directly From Application Programs.” Petitioners rely on Rosenthal as a secondary reference in Grounds 3 and 4, principally for its teachings regarding its “flow control register 45,” which Petitioners’ map to the RR. Pet. 57-60. Rosenthal was before the Patent Office during prosecution of the ’596 patent.

As relevant here, Rosenthal describes a system that is fundamentally and exclusively single processor (*i.e.*, CPU), where **only** the processor that is writing the

commands (the CPU) can check the register. EX1009, 2:28–33. Rosenthal thus teaches away from the '596 patent (and Petitioners' proposed combination), which requires that the 2PE checks the register while the 1PE issues the command.

5. U.S. Patent No. 6,810,462 (“Matsunami”)

Matsunami is entitled “Storage System And Method Using Interface Control Devices Of Different Types.” Petitioners rely on Matsunami solely for dependent claims 3-4, 9-10, 15-16 in grounds 2 and 5. Pet., 40-44, 82-87.

6. U.S. Patent No. 8,028,284 (“Ohkawa”) (EX1010)

Ohkawa is entitled “Data Processing System And Program For Transferring Data.” Petitioners rely on Ohkawa solely to supply the “output queue implemented in hardware” limitation in Grounds 4 and 5. Pet. 74–79.

C. Alleged Unpatentability Grounds

Grounds	Claims	Statutory Basis	Prior Art
1	1-2, 5-8, 11-14, 17-18	§103	Dongare in view of Gewirtz
2	3-4, 9-10, 15-16	§103	Dongare in view of Gewirtz and Matsunami
3	1-2, 6-8, 12-14, 18	§103	Khawand in view of Rosenthal
4	1-2, 6-8, 12-14, 18	§103	Khawand in view of Rosenthal and Ohkawa
5	3-4, 9-10, 15-16	§103	Khawand in view of Rosenthal, Ohkawa, and Matsunami

III. PROSECUTION HISTORY

Petitioners raise that a European counterpart to the '596 patent, EP14867963, was allegedly rejected because a Supplementary Search Report identified Gewirtz, with the EU Search Opinion noting that “[i]t is common general knowledge of the skilled person [] that an output queue [] is an implementation choice for buffering request results from a storage device.” Pet., 6 (citing EX1004, 000119; EX1003, ¶44).

Petitioners’ cited portion of the prosecution history is not relevant to this proceeding. Patent Owners do not dispute that output queues existed before the priority date of the '596 patent. Instead, as explained below, output queues were not used in the particular configurations claimed by the '596 patent before that date. *See* §§ VIII.B.2.c, C.1.c, C.2.c, C.3.a.

IV. PRIORITY DATE OF THE '596 PATENT

Petitioners claim the '596 patent has a priority date of December 6, 2013. Pet., 8. Patent Owners do not dispute, for purposes of this Response, that all of Petitioners’ prior art pre-dates the priority date of the '596 patent.

V. LEVEL OF ORDINARY SKILL

Petitioners identify their proposed level of ordinary skill. Pet., 8. For purposes of this proceeding, Patent Owners do not dispute Petitioners’ proposed level of ordinary skill, nor do Patent Owners dispute that Petitioners’ expert is a POSITA.

Patent Owners' expert, Dr. Paul Min, is a POSITA because he has a Ph.D., M.S., and B.S. in Electrical Engineering. EX2020, ¶¶9–22 (citing EX2021). Dr. Min is currently a Senior Professor in the Department of Electrical Systems Engineering at Washington University, St. Louis, and has worked in the field of electrical engineering and data processing, streaming architecture, and data networks for more than thirty years. *Id.*

VI. RELATED MATTERS

Patent Owners, CV/XE, presently assert the '596 patent in the following district court action: *Concurrent Ventures, LLC and XtreamEdge, Inc. v. Advanced Micro Devices, Inc. and Pensando Systems, Inc.*, No. 4:25-cv-09567-JST (N.D. Cal. 2025).

Petitioners have two other instituted *inter partes* reviews on two other patents in the district court action: IPR2025-00223 and IPR2025-00486.

VII. CLAIM CONSTRUCTION

The claim construction portion of the Petition identifies one term that is subject to dispute in the district court litigation: “[input/output] queue implemented in hardware,” but only provides a brief discussion on its relevance. Notably absent from this brief discussion is any statement as to what construction (or constructions) Petitioners actually apply in the Petition. Also absent is any discussion of what the plain and ordinary meaning of the term is, much less how Petitioners' district court

construction differs from the plain meaning. This fails to satisfy Petitioners' obligation to "identify . . . [h]ow the challenged claim is to be construed." 37 CFR § 42.104(b)(3).

Regardless, the Petition fails under both Petitioners' and Patent Owners' proposed construction, thus the Board need not construe this term. *See Nidec Motor Corp. v. Zhongshan Broad Ocean Motor Co.*, 868 F.3d 1013, 1017 (Fed. Cir. 2017).

VIII. PETITIONER HAS NOT PROVEN THAT ANY CHALLENGED CLAIM IS UNPATENTABLE

A. Petitioners' Arguments Stem from a Fundamental Misunderstanding of the Problem Solved by the '596 Patent.

The Petition fails to show that any combination of prior art would render obvious the challenged claims. The particular limitations missing are discussed below, but overarchingly, Petitioners fundamentally misunderstand the '596 patent, making the Petition inherently fall short.

The claims of the '596 patent reflect a key insight of the inventors: it is beneficial to offload the decision-making for *when* a first processor should issue a command to a distinct second processor. In the '596 patent, responsibilities are split: the 1PE is responsible for issuing the command, but it is the 2PE that is responsible for deciding when that command should be issued. Accordingly, each claim requires the 2PE to access and read from the hardware RR and, when there is sufficient space, either: (1) signal the 1PE that it may safely write a command into the input queue

(claim 1); or (2) receive the command from the 1PE and writes it into the input queue on behalf of the 1PE (claims 7 and 13). *Id.*, 4:26–27. This novel relationship between the 1PE, 2PE, and RR offloads responsibility from the 1PE, allowing it to allocate its memory to more demanding tasks. EX2020, ¶53.

Critically, the Petition does not even allege that any of its combinations disclose this idea, instead claiming a POSITA would have discovered it themselves because it is beneficial. But the mere fact that the patented invention is beneficial does not make it obvious—otherwise no good invention would be valid. *See Grunenthal GMBH v. Alkem Lab ’ys Ltd.*, 919 F.3d 1333, 1345 (Fed. Cir. 2019) (all non-obvious patents are inherently useful). Petitioners bear the burden of proving that the ’596 patent is unpatentable based on a preponderance of the evidence, and they fail to show that a POSITA would arrive at the ’596 patent through any of their proposed combinations. *See Harmonic Inc. v. Avid Tech., Inc.*, 815 F.3d 1356, 1363 (Fed. Cir. 2016) (burden on Petitioner to show “with particularity” that claims are unpatentable).

B. Petitioners Fail to Prove their Combination of Dongare and Gewirtz (Ground 1 and 2) Discloses or Renders Obvious Claims 1, 2, 5–8, 11–14, 17, and 18.

1. Claim 1

Grounds 1 and 2 fail to render obvious claim 1 for three independent reasons. First, Dongare and Gewirtz at most disclose two processors (Dongare) and a RR

(Gewirtz)—there is *nothing* in Dongare or Gewirtz to suggest that the alleged *2PE* uses the RR to determine when there is space in the input queue. Second, building on the first issue, there is no teaching or suggestion in either Dongare or Gewirtz that it is the alleged 2PE that notifies the 1PE to issue a command. The key innovation of the '596 patent—that different processors are responsible for issuing the command (the 1PE) and for deciding when that command should be issued (the 2PE)—is wholly absent from Petitioners' references. Third, neither reference discloses the "input queue" is implemented in hardware.

a. Dongare and Gewirtz Do Not Render Obvious the Second Processing Element Accessing and Utilizing the Reservation Register, as Required by Limitations 1[e], 1[g], 1[h], and 1[i].

Dongare and Gewirtz do not disclose or suggest (alone or in combination) a "second processing element" that accesses the RR (claim elements 1[e] and 1[g]), reads the stored value in the RR indicative of available space in the input queue (claim element 1[g]), and determines when that value indicates there is available space in the input queue for the 1PE to issue a command (claim element 1[h]) (collectively, the "Input Queue Determination Process"). EX1001, cl. 1. Petitioners' theory is simple (albeit unsupported): (1) Petitioners map Dongare's "EP 760" to the claimed "2PE," Pet., 28, 30–31; (2) Petitioners map Gewirtz's "Queue State Register 106" to the claimed "RR," Pet., 19–20; and (3) Petitioners assert that it would have been obvious for it to be EP 760 that accesses and utilizes Queue State Register 106

to perform the Input Queue Determination Process. Grounds 1 and 2 for claim 1 thus rest on a single, unsupported obviousness assertion: it would be obvious, ***without any suggestion or teaching in the art***, for it to be the 2PE that accesses the RR, reads the stored value, and determines when there is available space in the input queue for the 1PE to issue a command. That assertion should be dismissed as no more than *ipse dixit* and hindsight.

Nothing in Dongare suggests the EP 760 would perform the Input Queue Determination Process. Dongare contains limited discussion on EP 760, referencing it only three times. Dongare's ***entire*** disclosure regarding EP 760 is below:

- “The input queues 750 are read by the **embedded processor 760** and commands are created for RAID engines 730 to process data. Newly generated commands are loaded to input queues 750 dedicated to RAID engines 730.” EX1005, 14:41–46.
- “The **embedded processor 760** monitors output queues 780 for completion of RAID commands in the same order as was loaded in input queues 750.” EX1005, 14:49–51.
- “After completing the Host command, the **embedded processor 760** will signal the Host 710 that the command is complete by writing into the outbound queue 780.” EX1005, 14:52–54.

In the full context of Dongare, a POSITA would understand that EP 760's role is to process commands that have already been written to the input queue by: (1) portioning overly long commands into smaller command segments (if necessary); and (2) appending an identity tag to the command segments that identifies their order in the queue (*i.e.*, their sequence). EX2020, ¶98. After the commands are executed,

EP 760 also (3) reorders the commands back into their original sequence based on their tags and signals the Host via the output queue that the commands have been processed. That is the full extent of EP 760's responsibilities. EX1008, 10:50–11:13; *see also* EX2022, 104:12–16 (Dr. Russ confirming that EP 760 only operates on commands that Host 710 already added to an input queue).

Crucially, EP 760 does not have any role before the command is issued by the 1PE into the input queue. EX2020, 101. EP 760 tags commands that are already in the queue—that can only be done *after they are added to a queue* by Host 710. EX1005, 10:50–11:13, FIG. 3. Dongare simply does not suggest that the EP performs the Input Queue Determination Process.

Recognizing this gap, Petitioners allege that the missing elements would be obvious. In particular, Petitioners make three arguments.¹ First, they argue that it would be obvious for the “reservation register [to be] accessible by both the first and

¹ Petitioners also make a number of arguments about why a POSITA would be motivated to use Gewirtz's Queue Status Register with Dongare generally. *See, e.g.*, Pet., 25. These arguments highlight the issue with the Petition: the claims do not only require a RR, but that a 2PE uses the RR to determine when a 1PE should issue commands. These arguments do not even attempt to show why *that* claimed architecture would be obvious.

second processing elements” because “Gewirtz makes the register accessible by software agents running on ‘multiple processor cores’” and “[a] POSITA would have recognized the value in having this information be visible to multiple processors for synchronization purposes.” Pet., 22. This argument is not supported by Dongare or Gewirtz. EP 760 is only responsible for tagging and reordering commands. EX2020, ¶¶98, 104. There is no reason for it to access the RR, as the commands have already been added to input queues and thus it is too late to check whether there is space for them in the input queue. *Id.*, ¶104. Petitioners cite Gewirtz, but of course it says nothing about why EP 760 would have access to its alleged RR, as Gewirtz does not reference a 2PE. Pet., 22 (citing EX1006, 4:56–61). Petitioners also cite Dr. Russ’s declaration, but it is silent as to why EP 760 would have access to the RR. *Id.* (citing EX1003, ¶113). In any event, Petitioners’ argument at most shows that EP 760 could technically have access to the RR, not that EP 760 would have performed the rest of the Input Queue Determination Process.

Second, Petitioners argue it would have been obvious for EP 760 to perform the Input Queue Determination Process because, allegedly, “Dongare teaches that the embedded processor manages the input queues, e.g., through queue handling/scheduling, but leaves implementation details to the POSITA.” Pet., 19, 28–29 (“One part of queue management is tracking the amount of free space in the queue” (internal citations omitted)), 30–31. Petitioners’ assertion is flat out wrong.

As repeatedly explained, all EP 760 does is tag commands that have *already been issued* by the 1PE and reorder commands that reside within the output buffer. EX2020, ¶107. Dongare does not disclose that EP 760 plays any role regarding how commands are added to the input queue, much less that it manages that process. *Id.* Petitioners’ assertion that EP 760 has anything to with “tracking the amount of free space in the queue” is made up out of whole cloth.

Petitioners repeatedly overstate the role of EP 760 by suggesting it was placed on-chip “for precisely this type of management function,” Pet., 31, but Dongare makes clear that its role is entirely different. EX2020, ¶107, *see supra*. Petitioners’ citations do not show otherwise. Petitioners cite Dongare at 10:50–11:30, but this passage discusses that “[t]he input queue handler 510 is configured *to receive* a plurality of input streams”—not that EP 760 manages the input queues on the front-end before commands are input by the host processor. Pet., 28. Petitioners also cite Dongare at 14:30–55, but this passage likewise discusses that “[t]he *Host 710 writes commands* for the RAID controller *into the input queues 750*” and EP 760 only interacts with input queues that are full of commands. *Id.* Even Petitioners’ expert, Dr. Russ, agreed that this passage in Dongare means that the commands are written into the input queue *by Host 710* before EP 760 even accesses the input queues. EX2022, 100:2–101:7.

Third, Petitioners argue that “[i]t was obvious and natural for Dongare’s embedded processor to perform a coordination role and read the RR due to its proximity to the queues and the register (everything is on-chip, avoiding cross-chip communication delays), in contrast to the host processor.” Pet., 29–31. Petitioners’ baseless assertion fails to show these elements would be obvious.

As an initial matter, even if it would have been beneficial for EP 760 to perform the input queue offloading steps, this does not show it was obvious. The mere fact that the patented invention is beneficial does not make it obvious—otherwise no good invention would be valid. *See Grunenthal GMBH*, 919 F.3d at 1345. The benefits of the ’596 patent’s division of labor between a 1PE and 2PE do not retroactively render that inventive division obvious.

Regardless, Petitioners are wrong that it would have been obviously beneficial for EP 760 to perform the queue-state management. In Dongare, if Host 710 performed the Input Queue Determination Process, there would be two off-chip communications: reading the RR and, when appropriate, writing the command to the input queue. EX2020, ¶114. If EP 760 performed the Input Queue Determination Process, there would still be two off-chip communications *and also* one additional on-chip communication: reading the RR (on-chip); when appropriate, notifying Host 710 to issue the command (off-chip); and Host 710 writing the command to the input queue (off-chip). *Id.* Thus, using EP 760 to perform the Input Queue Determination

Process in the context of Dongare would involve more communications (one on-chip communication and two off-chip communications versus just two off-chip communications) and be slower. *Id.*

In fact, this hypothetical highlights the '596 patent's non-obviousness. For example, applying the '596 patent to Dongare would require *more* communications than prior art systems. In the context of Dongare, the obvious option is for the *IPE* to determine when to issue a command, since it is the processor that is writing that command. EX2020, ¶115. By adding a 2PE that has to determine when to issue a command and then also notify the Host to write that command, the '596 patent adds additional steps to the process, which inherently takes more time. *Id.* But the inventors of the '596 patent realized there were unexpected benefits in reducing latency that made this sacrifice worthwhile. *Id.* This unexpected benefit demonstrates that non-obvious nature of the '596 patent's solution.

In sum, Petitioners' arguments do nothing to show that it would have been obvious to have Dongare's EP 760 perform the Input Queue Determination Process. Petitioners argue it would have been obvious to transform Dongare and its EP 760 into a completely different type of processor—based, apparently, only on the fact that that is what the '596 patent claims, Pet., 31–34—but Courts consistently reject such hindsight arguments dressed up as obviousness. *See Cheese Sys. v. Tetra Pak Cheese & Powder Sys.*, 725 F.3d 1341, 1352 (Fed. Cir. 2013) (“Obviousness ‘cannot

be based on the hindsight combination of components selectively culled from the prior art to fit the parameters of the patented invention.””) (quoting *ATD Corp. v. Lydall, Inc.*, 159 F.3d 534, 546 (Fed. Cir. 1998)); *In re Fritch*, 972 F.2d 1260, 1266 (Fed. Cir. 1992) (“It is impermissible to use the claimed invention as an instruction manual or ‘template’ to piece together the teachings of the prior art so that the claimed invention is rendered obvious.”).

Moreover, Petitioners’ request for the Board to find these missing elements obvious based on their (unsupported) reasoning and expert opinion flies in the face of 37 C.F.R. § 42.104(b)(4). As the Office has recognized, 37 C.F.R. § 42.104(b)(4) and binding Federal Circuit caselaw states that “applicant admitted prior art (AAPA), expert testimony, common sense, and other evidence that is not ‘prior art consisting of patents or printed publications’ (collectively, ‘general knowledge’) may not be used to supply a missing claim limitation.” *See* EX2025 (July 31, 2025 Memorandum)² (citing 37 C.F.R. § 42.104(b)(4) and *Qualcomm Inc. v. Apple Inc.*, 134 F.4th 1355, 1364-65 (Fed. Cir. 2025)). Ground 1 fails this threshold, black letter

² Although the July 31, 2025 memorandum itself technically applies to petitions filed on or after September 1, 2025, it reflects the Office’s interpretation of the plain text of 37 C.F.R. § 42.104(b)(4) and precedential case law and those interpretations should apply here.

requirement that each element of each challenged claim must be present in a prior art patent or publication. The Petition relies on the wholesale creation of a limitation—the 2PE performing the Input Queue Determination Process—that is neither taught nor suggested by either of the references relied upon, plainly violating 37 C.F.R. § 42.104(b)(4) and Federal Circuit precedent.

b. Dongare and Gewirtz Do Not Render Obvious a Second Processing Element Sending Notifications to a First Processing Element to Issue Commands, as Required by Limitations 1[i] and 1[j].

Petitioners do not show that Dongare and Gewirtz render obvious a system where the “first processing element receives notification from said at least one second processing element regarding issuing the command.” EX1001, cl. 1. The Petition’s arguments regarding limitations 1[i] and 1[j] suffer from the same flaws as their argument for limitations 1[e], 1[g], 1[h], and 1[i], and those arguments are incorporated here. *See* § VIII.B.1.a. However, limitations 1[i] and 1[j] also suffer from an additional (albeit similar) flaw: neither reference even hints at using separate processors to: (1) issue commands (the 1PE); and (2) notify the 1PE *when* commands should be issued (the 2PE).

As explained above, EP 760: (1) portions long commands into smaller command segments; (2) tags commands and command segments for reordering after processing; and (3) reorders the commands after they have been processed. EX2020, ¶119. It does not read input queues to determine if space is available, schedule

commands to be issued into the input queue, or notify the host processor that space is available. *Id.* Dongare does not disclose a 2PE that **notifies** a 1PE whether there is space in the input queue to issue a command. *Id.*, ¶120. If anything, Dongare discloses that the Host 710 makes this decision on its own. *Id.*; EX1005, 14:36-42 (“The Host 710 and RAID controller 720 communicate to each other by using a set of queues 740 on the RAID control . . . The Host 710 writes commands for the RAID controller into the input queues 750.”).

The disconnect between Dongare and Gewirtz on the one hand, and the '596 patent on the other, is readily apparent when elements 1[i] and 1[j] are viewed in combination. Element 1[j] requires that the 1PE issues the command **after** it receives the notification from the 2PE. But in Dongare, Host 710 (the alleged 1PE) issues the command **before** EP 760 does *anything*. EX1005, FIG. 3. Thus, not only is there no suggestion that the 2PE notifies the 1PE when to issue a command, adding that role for EP 760 makes no sense. EP 760 has *no role before* the command is issued, while the claims require the 2PE to notify the 1PE to issue the command **before** the 1PE does so.

Moreover, Dongare is incompatible with elements 1[i] and 1[j] because there is no mechanism in Dongare for EP 760 to notify the 1PE to issue a command. EX2020, ¶123. While Host 710 connects to the ASIC via a PCIe bus, the **only** way that Dongare discloses EP 760 communicating with Host 710 is by “signal[ing] the

Host 710 that the command is complete by writing into the outbound queue 780. *Id.* In other words, EP 760 can only communicate with Host 710 via the output queue. *Id.*, ¶124. As a technical matter, EP 760 is incapable of notifying the Host to write new commands, because it can only communicate with the Host via the output queue where commands are stored *after* they have already been added to the input queue and processed.

Given that Dongare does not disclose EP 760 performing these claim elements, Petitioners resort to combining it with Gewirtz. Pet., 32. Petitioners first repeat their previous argument, alleging that “Dongare’s embedded processor [has a] queue-management and scheduling role.” Pet., 33, n.9 (explicitly referring back to previous arguments). Petitioners next allege it would have been obvious for “Dongare’s embedded processor [to] notif[y] the host over the PCIe bus or by generating an interrupt to the host” when the queue dips below a certain threshold because the host is off-chip and must communicate “over PCIe.” *Id.*, 33. Again, Petitioners’ argument fails for the same reason discussed above: EP 760 has a ***completely different role*** than the claimed 2PE. EX2020, ¶122. And, as discussed above, it would not have been obvious or even obviously beneficial for EP 760 to have taken on the role of the 2PE.

Petitioners arguable make one new argument for these elements in particular:

(1) Gewirtz discloses that the input queue can generate “interrupts” to a

programmable target to either say the queue is full or that it has fallen below a predetermined threshold of available space and (2) in light of this, it would have been obvious for the input queue to send the interrupt to EP 760, which would then notify Host 710. Pet., 32–33. This argument makes no sense. As discussed above, there is no disclosure in the references that EP 760 can communicate with the Host except through the *output* queues when the commands have been executed. Even if EP 760 receives an interrupt from the input queue, there is no way for it to notify the Host. EX2020, ¶¶124–25. And, since the EP’s only role is portioning, tagging, and reordering commands, EP 760 would have no use for Gewirtz’s queue status interrupts. *Id.* Petitioners assert that “Dongare’s embedded processor notifies the host over the PCIe bus or by generating an interrupt to the host,” Pet., 33, but there is no support for that assertion. First, there is no disclosure or suggestion that EP 760 can communicate directly with the Host over the PCIe bus and, second, the only “interrupt” it can generate to the host is through the output queue to tell the Host that the commands have been executed. EX2020, ¶¶124–25.

Further, even if Defendants’ proposed configuration were obvious (it is not), it does not actually meet the claim language. Petitioners allege that “when an input queue is full . . . the embedded processor notifies the host after the input queue drains below a predetermined threshold. The host processor can then issue commands to the input queue.” Pet., 33 (citations and footnotes omitted). But the claims require

that the 2PE notify the 1PE when there is “available space in said input queue for said first processing element to issue a command” and then “notif[ying] said first processing element to issue *said command* to said input queue.” EX1001, cl. 1. Simply alerting the 1PE that there is some “predetermined” amount of space in the input queue—without knowing what “said command” is, without determining whether there is room available for that particular command, and without notifying the Host to issue that particular command—does not meet claim 1.

Finally, Petitioners generically allege that “[t]he prior art provides ample motivation for one processor to notify another when space is available,” but none of their evidence supports that it would be obvious for EP 760 to notify the Host to issue a command. Pet., 33. For example, Petitioners cite unasserted prior art, U.S. Patent No. 5,204,954, Pet., 33 (citing EX1011, 10:3–26, 11:52–54), but that does not provide any motivation to reconfigure mismatched processors for entirely new roles, as Petitioners attempt.

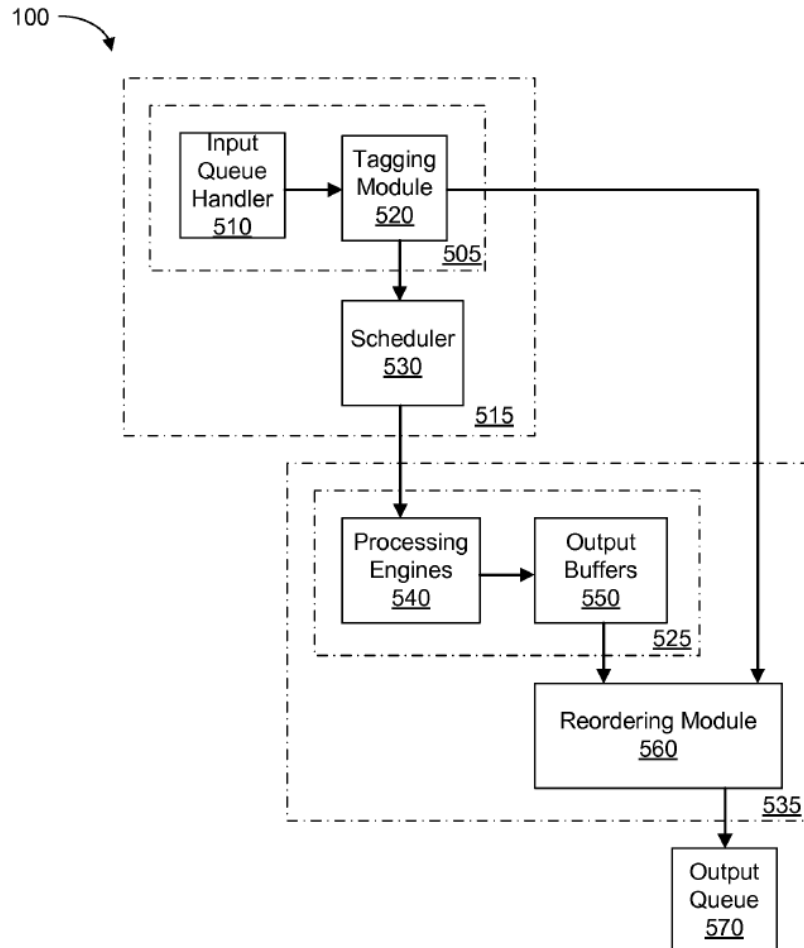
Petitioners’ argument that EP 760 notifying Host 710 to issue commands can be read into the art because it would be obvious plainly violates 37 C.F.R. § 42.104(b)(4) and Federal Circuit precedent and thus fails.

c. **Dongare Does Not Render Obvious an “Input Queue Implemented in Hardware” As Required By Limitation 1[a].**

Petitioners have not shown that Dongare discloses “an input queue implemented in hardware.” EX1001, cl. 1, 7 and 13. Petitioners identify a single sentence in Dongare to support their argument: “[i]t is noted that the elements of the multi-engine processing system 100 may be implemented as a software routine, hardware circuit, firmware, or a combination thereof.” Pet., 15–16 (citing EX1005, 11:14–16). The remainder of their evidence only shows that Dongare uses input queues generally, without specifying whether they are in software or hardware. *See, e.g.*, Pet., 14–15 (citing Ex-1005, 5:3-36, 14:39-42; Ex-1003, ¶¶84-87.). Petitioners’ single sentence fails to show that Dongare discloses “an input queue implemented in hardware.”

Dongare is silent on whether its input queues are in hardware because none of the evidence cited by Petitioners says anything about *input queues*. Dongare provides that “elements of the *multi-engine processing system 100* may be implemented as a software routine, hardware circuit, firmware, or a combination thereof,” but *Dongare’s multi-engine processing system 100 does not include the input queues*. Pet., 15–16 (citing EX1005, 11:14–16) (emphasis added). Instead, as shown in Figure 5 below, “[m]ulti-engine processing system 100” only comprises “an input queue handler 510; a tagging module 520; a scheduler 530; a plurality of

processing engines 540; a plurality of output buffers 550; a reordering module 560; and an output queue 570.” EX1005, 10:51–56.



Id., FIG. 5. Each of these hardware components is separate from the input queues flowing *into* the system—Input Queue Handler 510 and Tagging Module 520 are only responsible for tagging packets *after* they are placed in an input queue. EX1005, 10:56–62.

Dr. Min explained that a POSITA would assume these modules act on commands from input queues implemented in software, not hardware. EX2020,

¶132. A POSITA would assume that Dongare’s queues are software queues because it does not discuss the available space in the queues. *Id.* Software queues essentially do not have fixed sizes as the size can increase or decrease using the head and tail pointers, while hardware queues have fixed sizes. *Id.* Because the reference does not disclose any information about the size of its queues, a POSITA would not understand that Dongare’s queues are hardware. *Id.*

Petitioners cannot use boilerplate language referring to *a different part* of the system and import it as evidence for this argument. The mere fact that a system includes hardware circuits elsewhere does not mean that it would be obvious for its queues to be implemented in hardware. Pet., 15–16 (citing EX1005, 11:14–16, 2:55–58; EX1003, ¶¶88, 90–91).³

³ The Board considered this issue as part of the Institution Decision (“ID,” Paper 11), but it did not have the benefit of the expert testimony now in the record. The Board stated that Dongare likely discloses input queues implemented in hardware because its ASIC connects to the Host computer via a PCIe protocol bus. ID (Paper 11), 19. But while the input queues are on a module in an ASIC, this *does not* mean that the input queues are necessary in hardware. EX2020, ¶133. And the read/write arrow between Host 710 and the Queues 740 indicates the host can issue read/write commands to the input queues, not that the queues are in hardware. *Id.*

2. Independent Claims 7 and 13

Claims 7 and 13 are similar to Claim 1, but each have a key difference that cause Petitioners' arguments to fail for additional reasons. For claims 7 and 13, while 1PE still serves as the originator of commands and recipient of responses (as in claim 1), 2PE now issues the commands into the hardware input queue "on behalf of" 1PE. *Id.*, cl. 7. For claim 13, 2PE is also given access to the output queues and 2PE (instead of 1PE) receives the response from the output queue. *Id.*, cl. 13.

a. **Limitations 7[a], 7[e], 7[h]–7[i] and 13[a], 13[e], 13[h]–13[i] Fail for the same reasons as equivalent in claim 1.**

As shown above, Dongare and Gewirtz do not anticipate or render obvious limitations 1[a], 1[e], 1[g]–[j]. For the equivalent limitations in claims 7 and 13, Petitioners simply cite back to the portions of the Petition concerning claim 1 without any additional evidence, analysis, or argument. Pet., 37–38. Thus, Petitioners fail to show that elements 7[a], 7[e], 7[h]–7[i] and 13[a], 13[e], 13[h]–13[i] would be obvious for the same reasons discussed above.

b. **Dongare and Gewirtz Did Not Render Obvious a Second Processing Element Receiving a Command from the First Processing Element and Issuing a Command to the Input Queue on Behalf of the First Processing Element, as Required by Limitations 7[g]/[j] and 13[g]/[j].**

Claims 7 and 13 requires that 1PE sends the command to 2PE, which then issues the command "on behalf of" 1PE once 2PE determines that space is available in the input queue. Petitioners propose a hypothetical configuration in which EP 760

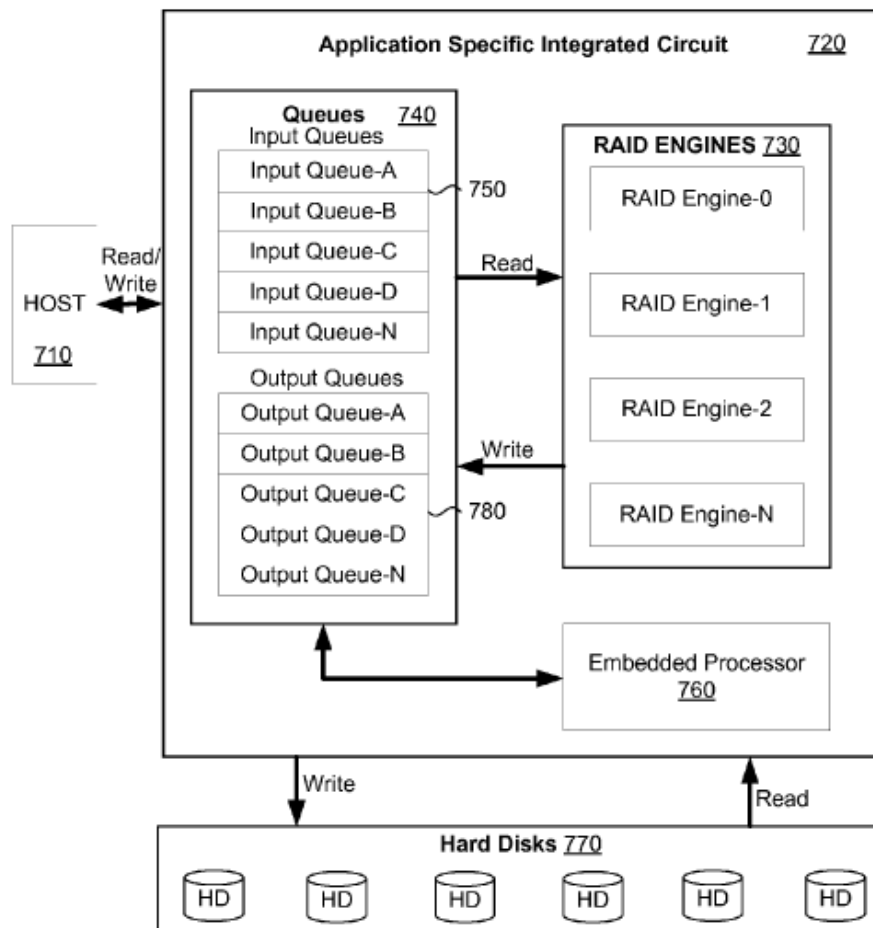
receives a command from the 1PE and then issues that command on behalf of the 1PE. Pet., 38–39 (citing EX1005, 14:41–44, FIG. 7). In particular, Petitioners propose the following⁴: (1) Host 710 issues a command to the input queue; (2) EP 760 reads those commands from the input queue, thereby allegedly “receiving” the commands from Host 710; (3) EP 760 uses Gewirtz’s RR to determine whether there is room in the input queue for the commands *which are already in the input queue*; and (4) EP 760 tags and portions out those commands, thereby allegedly “issuing” them on behalf of Host 710. *Id.* Petitioners’ hypothetical configuration mischaracterizes Dongare, lacks any support in the record, and flat out makes no sense.

First, the 2PE does not “receive[] a command from said first processing element.” Petitioners argue that Host 710 sends the command to EP 760 by writing the commands into the input queue. Pet., 38. That makes no sense. The claims require that 2PE receives a command from 1PE so that 2PE can later issue that command to the input queue. It makes no sense for 1PE to send a command to 2PE *by issuing the command to the input queue itself* only for 2PE to later reissue that

⁴ Petitioners never set forth their full proposed combination in a single location, likely because it makes little sense when viewed in its entirety.

command. Nor would a POSITA understand that a 2PE reading a command already in the queue to be receiving a command.

Nothing in Dongare or Gewirtz suggests the 2PE “receives a command from said first processing element.” Dongare discloses Host 710 writes commands to the input queue and there is no disclosure or suggestion that it can send them to EP 760. EX1005, 14:41–42; EX2020, ¶137. In fact, as discussed above, Host 710 and EP 760 can only communicate via the output queue and only for EP 760 to tell Host 710 when the commands have been executed. *See* § VIII.B.1.b. There is no way for Host 710 to send a command to EP 760.



EX1005, FIG. 7.

Second, EP 760 does not “issue[] said command to said input queue on behalf of said first processing element.” EX1001, cl. 7, 13. Petitioners rely on the following disclosure from Dongare: “The input queues 750 are read by the embedded processor 760 and commands are created for RAID engines 730 to process data. Newly generated commands are loaded to input queues 750 dedicated to RAID engines 730.” Pet., 38. But Petitioners’ own expert⁵ testified that this passage only discloses EP 760 tagging and potentially dividing the “existing commands” not creating and issuing new ones:

Q. Does the embedded processor create commands?

A. The words on the page are commands are created for the RAID engines, and so I acknowledge the words on the page.

I think read in context, *it’s clear that [the Embedded Processor] is simply tagging, and potentially dividing the existing command, because the alternative would be really, really weird for a disk controller.*”

⁵ Dr. Russ also conceded that Dongare provides little detail as to how its disclosed system works. *See, e.g.*, EX2022, 105:9–16 (“I don’t know. I mean, Dongare is silent on the subject [of how IPM 308 could write commands into an input queue if Host 710 already wrote commands into the input queue].”); 106:4–7.

EX2022, 108:5-13 (emphasis added). When asked what this passage means, Dr.

Russ testified that in Dongare:

[T]he Host is writing a command, and it gets put into the input queue, and the embedded processor performs processing on that command. As we described earlier, for example, it performs a tagging operation, which is important for making sure commands are issued in the correct order. And that processor can also divide a command into portions. *And so what the words on the page say is ‘newly generated commands,’ and of course it can’t mean something like completely different.* That just wouldn’t make any sense if you’re writing files to a hard drive. *What it actually means is it’s the commands that have been divided into portions and tagged.*

Id., 104:5–105:8 (emphasis added). Petitioners even admit that “[t]he input queues are *read* by the embedded processor 760,” which means the commands *have already been written to the input queue* by the 1PE. Pet. 38–39 (emphasis added). In other words, Dongare discloses the complete opposite of claims 7 and 13: Host 710 (not EP 760) writes the command into the input queue.

Moreover, Petitioners’ proposed configuration for this limitation is again nonsensical. In Petitioners’ hypothetical, EP 760 “receives” commands that are already in the input queue. Thus, Petitioners’ obviousness argument requires EP 760 to “determine” if there is to available space in the input queue for commands that have *already been issued* to that input queue. That is absurd: there inherently must

be sufficient space for the commands in the input queue because they are already in the input queue. EX2020, ¶141.

c. Dongare and Gewirtz Did Not Disclose or Render Obvious Limitation 13[k].

Petitioners also do not show that EP 760 receives a response corresponding to Host 710's command from the RAID engines. EX1001, cl. 13. Dongare discloses that EP 760 continually *monitors* output buffers—it does not receive a response from them. EX1005, 14:49–51. In particular, Dongare discloses that EP 760 is not notified by the output queues when a task is finished, but rather each tagged output from the RAID engines is stored “in a temporary write out buffer . . . as it is identified,” *id.*, 8:4–8, and EP 760 “*repeatedly scan[s]* the plurality of dedicated output buffers to identify tagged output segments according to their relative order within each input stream to generate the plurality of output streams,” *id.*, 7:65–8:3 (emphasis added). After the commands are processed, they “*wait* for write out until [] selected by the sequencer.” *Id.*, 12:30–33 (emphasis added).

Petitioners do not present any evidence that EP 760 receives a response from an output queue, nor do they contend that it would be obvious for it to include this functionality. Pet., 40. Dr. Russ does not add anything to support Petitioners' assertion. In contrast, Dr. Min explains that a POSITA would not expect EP 760 to receive a response corresponding to the command because it did not add the

command to the input queues on behalf of another processing element. EX2020, ¶147.

3. Dependant claims

a. **Dongare and Gewirtz Did Not Disclose or Render Obvious Claims 5, 11, and 17.**

Petitioners provide almost no argument for claims 5, 11, and 17, instead simply alleging that Gewirtz discloses these claims because it teaches that “[t]he available reservation count is maintained by hardware: it is decremented when a successful reservation read request is processed.” Pet., 36 (citing EX1006, 4:56–5:6).⁶ But Gewirtz does not teach that “*the stored value*” is returned to the 2PE. *Id.*

Gewirtz does not return a numerical value of available space in the queue to the 2PE—it only teaches a system where the RR returns a pass/fail value to the host processor. EX2020, ¶150 (citing EX1004, 4:59–67; 4:26–29 (“Non-blocking reads may return a fail value to the requesting agent, whereas blocking reads may be queued in the queue element storage space (e.g., memory 103) in the order that they are received.”)). This evidence is uncontroverted. Petitioners only contend that Gewirtz teaches that the space counted is stored in the Queue State Register and

⁶ Petitioners do not rely on Dongare for the new limitations recited in these dependent claims. Nor do Petitioners argue that it would be obvious in the art to meet the “returning said stored value” limitation.

decremented upon a successful read request. That value is not sent anywhere. Pet., 36; EX1003, ¶154–57 (only discussing value, not that it is sent to another processor).

Gewirtz thus explicitly discloses an alternative path—only returning a pass/fail—and teaches away from returning the value itself. In light of that, it would not be obvious to a POSITA for Gewirtz to return the claimed value. In the '596 patent, it is beneficial for the 2PE to obtain the stored value upon a read request because: (1) this is how the 2PE determines there is enough space to add a particular command to the queue; and (2) it allows the 2PE to make policy decisions and conduct the “peek” operation described in the specification. EX2020, ¶153; EX1001, 3:49–64. If the RR only provided a pass/fail result in response to a reservation request, the 2PE would not be able to “reserve however much space is available” or “peek” at the available space. *Id.*, 3:52–64. Gewirtz does not recognize these benefits or propose the claimed solution, and thus does not render it obvious.

b. Dongare and Gewirtz Did Not Render Obvious Claims 6, 12, and 18.

Finally, Dongare and Gewirtz do not render obvious claims 6, 12, and 18. Claims 6, 12, and 18 expressly require that it is the *third processing* element that “notifies said output queue to issue said response” *to the first processing element*. *Id.*, cl. 6, 12, and 18; 5:56–59. Petitioners allege that the RAID engines in Dongare execute the commands and “after processing them (e.g., read/write operations from Hard Disks 770), writes the processed commands back to output queues 780.” Pet.,

37 (citing EX1005, 14:46–49). But while the RAID engines (*i.e.*, the third processing elements) process the commands, Dongare explains it is EP 760 (*i.e.*, the 2PE)—not the Raid Engines—that notifies the output queue to issue said response to 2PE. EX1005, 14:52–54.

In particular, EP 760 “monitors output queues” from the RAID engines and “will signal the Host 710 that the command is complete by writing into the outbound queue.” EX1005, 14:49–54. Thus, the RAID engines write processed commands into output buffers, which are only accessible to EP 760, and then EP 760 notifies the 1PE of the completion via an output queue. *Id.*, Abstract.

Petitioners do not address this distinction. They provide no explanation for the differing relationships, nor do they explain why a POSITA would see these different relationships and find the ’596 patent’s configuration obvious. In fact, Dr. Russ admitted that it was EP 760—not the third processing element—which notifies the host. EX2022, 121:16–21.

Dongare’s use of the output queue is emblematic of the different purpose of the invention and the 2PE. Dongare teaches a system where EP 760 monitors the output queues because the invention is only a tagging system used to optimize downstream processing. EX1005, 11:9–11. EP 760 notifies the host processor via an output queue because its purpose is to reorder the completed commands after processing before they reach their next destination. *Id.* 11:9–11. It would not make

sense for the RAID engines (*i.e.*, the third processing element) to notify Host 710 via an output queue because the processing is not complete until after EP 760 conducts the reordering. EX2020, 163.

4. Ground 2 Does Not Address The Problems In Ground 1.

In Ground 2, Petitioners rely on Matsunami only for the limitations recited in claims 3–4, 9–10, 15–16. Petitioners do not rely on Matsunami to teach or suggest any of the limitations challenged above. Matsunami thus does not address the deficiencies in Ground 2.

C. Petitioners Fail to Prove their Combination of Khawand, Rosenthal, Ohkawa, and Matsunami (Grounds 3, 4, and 5) Discloses or Renders Obvious Claims 1–4, 6–10, 12–16, 18.

Having failed to establish unpatentability through the combination of Dongare and Gewirtz, Petitioners turn to alternative grounds based on Khawand and Rosenthal. These grounds also fail, both because the references teach away from Petitioners' proposed combinations and because, even if Khawand and Rosenthal were combined, the combination still fails to render obvious multiple claim limitations.

1. Claim 1

a. The References Teach Away From Petitioners' Proposed Combination.

i. Rosenthal Teaches Away From Petitioners' Proposed Combination for Limitations 1[e]–1[j].

Petitioners rely on the combination of Khawand *and* Rosenthal to argue that limitations 1[e]–1[j] would have been obvious. In particular, Petitioners rely on Rosenthal’s “flow control register” to map to the claimed reservation register (“RR”) and Khawand’s IPM 308 to map to the claimed second processing element (“2PE”). Pet., 44–47. Accordingly, Petitioners contend it would have been obvious for Khawand’s IPM 308 to access Rosenthal’s alleged flow control register, read from it the value indicative of available space in the input queue, and determine when that value indicates there is available space in the input queue for the command. *Id.*, 53–60. But Rosenthal explicitly teaches away from Petitioners’ proposed combination because it states that processors like IPM 308 *cannot* access Rosenthal’s register. Thus, Petitioners have failed to show that a POSITA would have found limitations 1[e]–1[j] obvious.

Rosenthal describes a system in which a CPU (a type of processor) can run multiple different applications programs, each of which has an associated input buffer. EX1009, 2:18–33. Rosenthal explains that the CPU running the application program can access a “flow control register” to determine whether there is sufficient space in the input buffer for new commands. *Id.*, 16:33–49. If there is, the application program running on the CPU can then write new commands to the input buffer. *Id.*, Abstract, 2:21–26.

Crucially, Rosenthal states that access to input/output devices is limited to the operating system of the CPU, and that *only* the CPU can access and read from the input/output devices, including the “flow control register.” EX1009, 1:10–28; *see, e.g., id.*, 16:44–46 (“This it does by having the *central processing unit* read the value in the flow control register 45 associated with the FIFO buffer 39 being utilized.” (emphasis added)). Rosenthal repeatedly states that the “*flow control register [] must be read by the processing unit running the application register.*” *Id.*, Abstract, 2:21–26 (emphasis added). In other words, Rosenthal requires that the processor writing commands (the CPU) is the only processor that can read from the flow control register. EX2020, ¶¶170–71. Thus, Rosenthal is only applicable to systems with a single processor (the CPU), in contrast to the ’596 patent, which requires multiple processors. *See, e.g.,* EX1009, 4:6–11.

Rosenthal’s single-processor architecture is fundamentally inconsistent with Petitioners’ proposed combination. Petitioners propose modifying Khawand so that IPM 308 (the alleged 2PE) could access and use Rosenthal’s flow control register to determine whether there is space in the input queue for Khawand’s Processor 1 (the alleged 1PE) to write a new command. Pet., 62–63. But that contradicts Rosenthal’s disclosure that only the processor writing commands (Khawand’s Processor 1) can access its flow control register. Based on this, a POSITA would not be motivated to combine Rosenthal with Khawand for Petitioners’ proposed combination. *See*

Adidas AG v. Nike, Inc., 963 F.3d 1355, 1359 (Fed. Cir. 2020) (“Fundamental differences between the references are central to this motivation to combine inquiry.”); *Henny Penny Corp. v. Frymaster LLC*, 938 F.3d 1324, 1329 (Fed. Cir. 2019).

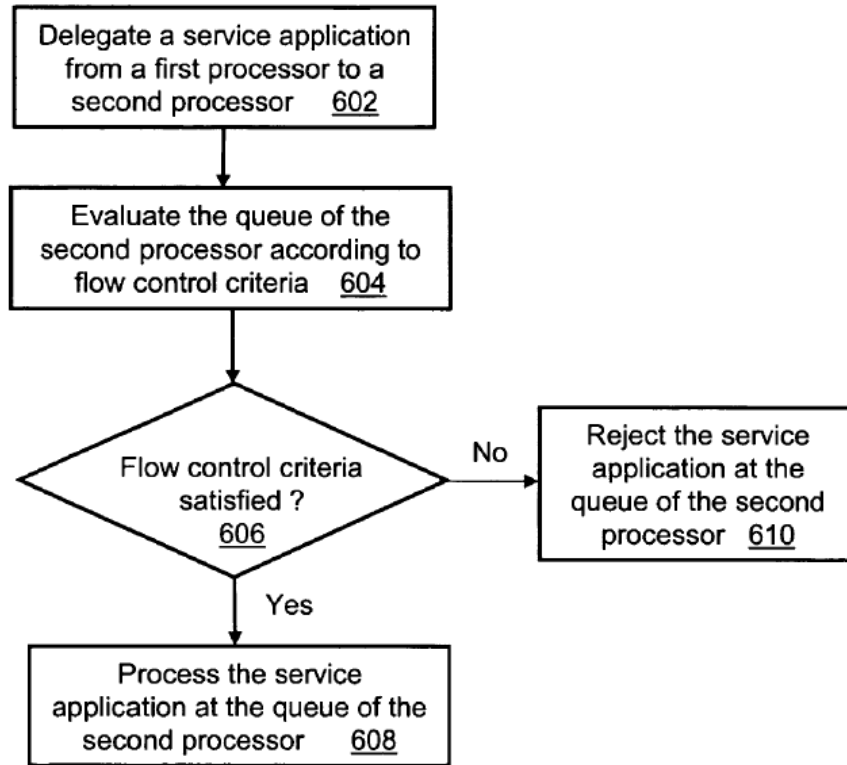
Without this combination, Petitioners have no argument left for why the ’596 patent’s requirement for a processor that uses the register to determine whether there is space in the input queue for a *different* processor to write a new command would have been obvious. Petitioners thus cannot prove claim 1 unpatentable under Ground 3.

ii. Khawand Teaches Away From Petitioners’ Proposed Combination For Limitation 1[h].

Even if a POSITA would have been motivated to combine Khawand and Rosenthal in general, a POSITA would not have been motivated to combine them as Petitioners propose for limitation 1[h] because Khawand teaches away from the combination. “There is no suggestion to combine . . . if a reference teaches away from its combination with another source.” *Tec Air, Inc. v. Denso Mfg. Michigan Inc.*, 192 F.3d 1353, 1360 (Fed. Cir. 1999). “A reference may be said to teach away when a person of ordinary skill, upon reading the reference . . . would be led in a direction divergent from the path that was taken by the applicant . . .” *Tec Air, Inc. v. Denso Mfg. Michigan Inc.*, 192 F.3d 1353, 1360 (Fed. Cir. 1999) (quoting *In re Gurley*, 27 F.3d 551, 553, 31 USPQ2d 1130, 1131 (Fed.Cir.1994)).

Here, Petitioners argue that it would have been obvious to modify Khawand so that its IPM 308 “determines if there is enough space in the input queue for the command as Rosenthal teaches.” Pet., 62–63. But Khawand discloses its own, “divergent” approach that is different than that described in Rosenthal and claimed in the ’596 patent.

As shown in Figure 6 below, Khawand’s first step is Step 602, in which the “first processor” (the alleged 1PE) “delegate[s] a service application from a first processor to a second processor [the alleged 3PE].” *Id.*, Fig. 6, 3:65–4:2. As explained by Petitioners’ expert, this means that “the first processor writ[es] [the application] to the queue of the second processor.” EX2022, 139:11–21. After Khawand’s Processor 1 writes the application to the queue of Khawand’s Processor 2, IPM 308 evaluates Processor 2’s queue and determines whether the flow control criteria is satisfied (in Steps 604 and 606).



EX1008, FIG. 6.

Khawand's "flow control criteria" is based on the "capacity threshold," which "indicates the capacity of the queue 404 for processing service applications." EX1008, 4:15-18. When the queue is over the "capacity threshold," the service application is rejected and a "suspend command" is issued so that no more service applications can be put into the queue; when the queue falls below the "capacity threshold," a "resume command" is issued, and more service applications can be

placed in the queue.⁷ *Id.*, 4:18–67. Khawand discloses that the capacity threshold can be static or variable and describes a complex method for determining a variable threshold. *Id.*, 4:28–59. But for both static and dynamic capacity thresholds, IPM 308 simply determines whether the queue is above or below the current capacity threshold. Thus, Khawand affirmatively discloses a “determination” that is based solely on: (1) how full the queue is; and (2) what the capacity threshold is. Notably, this method of “determining” does **not** account for any particular command and thus does **not** determine if there is enough space available in the queue for that particular command, as required by Petitioners’ combination and the ’596 patent.

Khawand’s approach is fundamentally different than that taken by the ’596 patent. In Khawand, IPM 308 only determines whether there is sufficient space *in general* to re-open the queue (based on the capacity threshold). Indeed, much of Khawand is concerned with how best to calculate a capacity threshold. Notably, that calculation is only necessary because of Khawand’s alternative approach. For example, there is no need for a “capacity threshold” in the ’596 patent because either

⁷ Although Khawand refers to “suspend **commands**” and “resume **commands**,” Petitioners do not map these concepts to the claimed “commands” issued to the input queue. To avoid confusion, this Response always refers to these terms with quotation marks: “suspend command” and “resume command.”

there is enough room in the queue for a particular command or there is not. EX2020, ¶¶179–80. But in Khawand, the IPM 308 is “determining” whether there is enough room available in the queue to safely reopen the queue for new applications in general, without reference to any particular command. Thus, Khawand is concerned with calculating how much availability a queue must have in general for it to safely resume allowing new service applications (the “capacity threshold”). This alternative determining is not a peripheral teaching; it is central to Khawand and is even explicitly required by each claim. *See, e.g.*, EX1008, cl. 1.

Khawand’s “capacity threshold” determination represents “a direction divergent from the path that was taken by the applicant” that teaches away from the ’596 patent. *See Tec Air, Inc.*, 192 F.3d at 1360. It is improper to combine references to create a new hypothetical system where the references teach away from that hypothetical system. *See In re Grasselli*, 713 F.2d 731, 743 (Fed. Cir. 1983); *Arctic Cat Inc. v. Bombardier Recreational Prods. Inc.*, 876 F.3d 1350, 1360 (Fed. Cir. 2017) (“[P]rior art need not explicitly ‘teach away’ to be relevant to the obviousness determination.”). Thus, a POSITA would not have been motivated to substitute Rosenthal’s flow control method of Khawand’s, as Petitioners suggest.

b. In Any Event, Petitioners’ Proposed Combination Does Not Render Obvious Claim 1.

Even if a POSITA would make Petitioners’ proposed combination, it does not render obvious claim 1. First, neither reference teaches or suggests a second

processing element notifying a first processing element to issue a command. Second, neither reference teaches or suggests that the first processing element accesses an “output queue” and “receives a response corresponding to said command from said output queue.”

i. Limitations 1[i]–1[j] Are Not Obvious in Light of Khawand and Rosenthal.

Khawand and Rosenthal do not teach or suggest a second processing element notifying a first processing element to issue a command, and Petitioners have provided no reason a POSITA would have plucked it out of thin air. *See Univ. of Strathclyde v. Clear-Vu Lighting LLC*, 17 F.4th 155, 161 (Fed. Cir. 2021) (“Given neither [reference] teaches or suggests [the limitation], we fail to see why a skilled artisan would opt to [add the limitation] when combining these references.”).

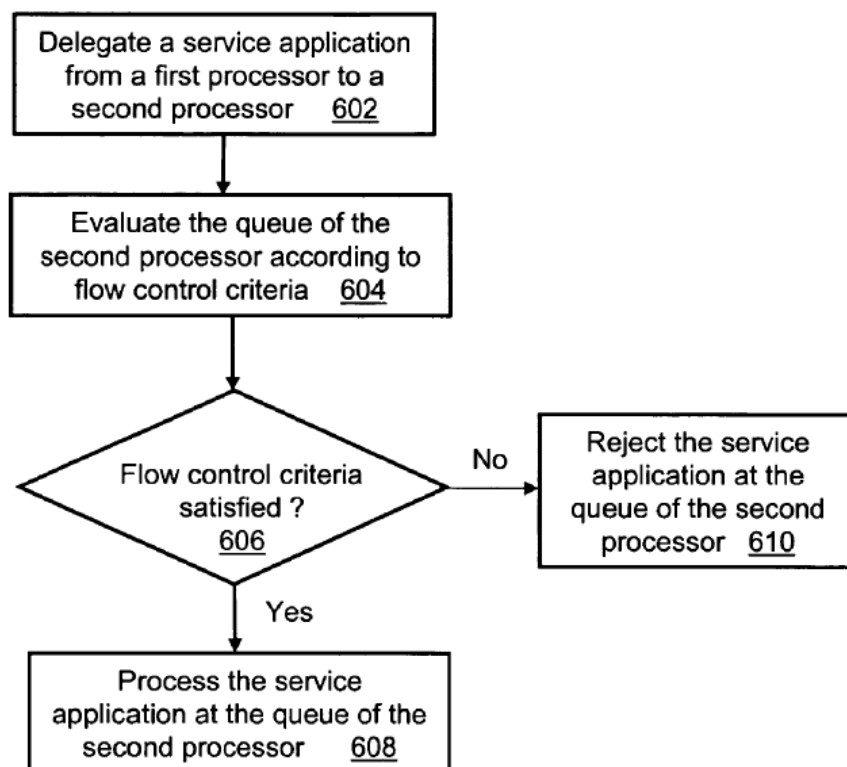
Petitioners allege that “Khawand teaches, or at least suggests, that [IPM 308] notifies the first processor to issue the command to the input queue.” Pet., 64. In particular, Petitioners argue that “Khawand’s Processor 2 [the claimed “third processing element”] sends a suspend command to [IPM 308] [the claimed “second processing element”] when its queue is full, indicating that the associated queue is not available” and “[w]hen the input queue becomes available again, [IPM 308] notifies the first processor of the availability through a resume command.” *Id.*, 63. Petitioners contend that this “resume command” is what “notifies” the 1PE to issue the claimed “command.”

Petitioners' theory stumbles out of the gate: a "resume command" does not "notif[y] said first processing element to issue said command to said input queue" and Khawand's Processor 1 does not issue "said command." As discussed above, IPM 308 does not notify Processor 1 to issue a particular command, but instead issues a "resume command" that is a traffic signal that *allows* Processor 1 to resume writing to the input queue. The differences between a "resume command" and the claimed "notif[ying]" are manifest. For example, Khawand does not disclose that IPM 308 has any knowledge of a particular command and thus IPM 308's "resume command" *cannot* notify 1PE to issue "said command." EX2020, ¶188. For another example, Processor 1 might have no commands pending, in which case it will not issue any commands after the "resume command." *Id.* The fact that Processor 1 may not issue *any* commands after the "resume command" further demonstrates that IPM 308's "resume command" does not "notif[y]" Processor 1 to issue a command. These issues reflect a fundamental disconnect between Khawand and the '596 patent: IPM 308's "resume command" only informs Processor 1 that it is *allowed* to issue any commands it wants, it does not "notif[y] said first processing element to issue" a command. *Id.* Accordingly, Processor 1 does not "issue[] said command," as there is no "said command."

In sum, nothing in either reference suggests that IPM 308 notifies Processor 1 to issue "said command." That itself defeats Ground 3. *See* 37 C.F.R.

§ 42.104(b)(4); *Qualcomm Inc.*, 134 F.4th at 1364-65; *see also See Univ. of Strathclyde*, 17 F.4th at 161.

What is worse, IPM 308 is **incapable** of notifying Processor 1 to issue “said command” because it only evaluates the queue **after** Processor 1 has already written the command. EX2022, 139:13–16 (delegation occurs by the act of writing a service application to a queue). As shown in Figure 6, Khawand’s Processor 1 writes the application to the queue of Khawand’s Processor 2 as the first step of the process (Step 602)—before the IPM 308 has done anything, and certainly without first receiving approval from IPM 308 that there is space available in the queue.



EX1008, FIG. 6.

It is only *after* Khawand's Processor 1 writes the application to the queue of Khawand's Processor 2 that IPM 308 "evaluates the queue" and determines whether the flow control is satisfied. That means that, at most, IPM 308 can drop a command from an input queue. EX2020, ¶190. That is vastly different than determining whether to notify Processor 1 to issue the command in the first place. *Id.*

In addition to failing to disclose this limitation, Khawand's alternative approach undermines a key objective of the '596 patent. The '596 patent ensures that input queues never overflow with commands exceeding the available space. EX2020, ¶191–94. Without this safeguard, such a command could be dropped or corrupt the queue. *Id.* But Khawand only operates at a level of generalized queue availability. *Id.* IPM 308 evaluates whether its queue is above or below a certain ***threshold*** and then issues a suspend or resume signal to IPM 308. *Id.*; EX1008, 4:15–59. This suspend or resume signal is not based on the size requirement of the specific command that Processor 1 seeks to delegate—the signal only conveys whether the queue is above or below its defined threshold at a given time. EX2020, ¶¶191–94. There is nothing to prevent the next service applications from immediately exceeding the capacity if it is larger than IPM 308 expects. *Id.* In fact, Khawand expressly tolerates this imprecision by allowing IPM 308 to reject a service application after it is written to the queue, confirming that its design is retrospective. *Id.* When asked at his deposition, Dr. Russ did not know what it meant for a service

application to be “rejected,” EX2022, 144:4–5, but a POSITA would understand it to mean the command is dropped, which the ’596 patent is designed to avoid, *id.*, ¶194.

ii. Limitations 1[c] and 1[j] Are Not Obvious in Light of Khawand and Rosenthal.

Petitioners also have not shown that Khawand and Rosenthal teach or suggest that the 1PE accesses an “output queue” and “receives a response corresponding to said command from said output queue.” EX1001, cl. 1. Khawand discusses that Processor 1 may delegate a service application to Processor 2. However, Processor 2 does not send this output *back* to the 1PE. When Processor 1 delegates a service application, it cedes all ownership and responsibility for it.

It is undisputed that Khawand does not discuss output queues. EX2020, ¶199. Recognizing this, Petitioners contend that it would nonetheless be obvious that Processor 1 receives a response corresponding to the service application it delegated. Pet., 65. In support, Dr. Russ provides a conclusory opinion that “the purpose of delegation is for another processor to process a service application, [so] it would have been obvious to include an output queue to return processed data from that service application” and “output queues had been used for that purpose since at least the 1990s.” EX1003, ¶293.

Petitioners misstate “the purpose of delegation.” Pet., 65. It would not be obvious for Processor 1 to receive a response corresponding to the delegated service

application exactly for the reason that Dr. Russ opined: the purpose of delegation is for *another processor* to process a service application. EX2020, ¶200. Khawand teaches a system for load balancing across functional processors, so when Processor 1 delegates a service application, it cedes all responsibility for it and leaves any processing and output up to Processor 2. *Id.* The purpose of an output queue is to deliver results or acknowledgements back to the processing element where the command originated so that the processing element may either confirm the command was processed or conduct further processing with the results of the command. *Id.*, ¶201. But in Khawand, Processor 1 is a functional processor tasked with executing the service application *on behalf of another processor*. *Id.* Service applications do not originate at Processor 1, thus Processor 1 has no use for the results of the service application and would not require a response before processing other service applications. *Id.*

2. Claims 7 and 13

a. Limitations 7[e]–7[i] and 7[k], and 13[e]–13[i] Fail for the same reasons as equivalent in claim 1.

As shown above, Khawand and Rosenthal do not anticipate or render obvious limitations 1[e]–[j]. For the equivalent limitations in claims 7 and 13, Petitioners simply cite back to the portions of the Petition concerning claim 1 without any additional evidence, analysis, or argument. Pet., 37–38. Thus, Petitioners fail to

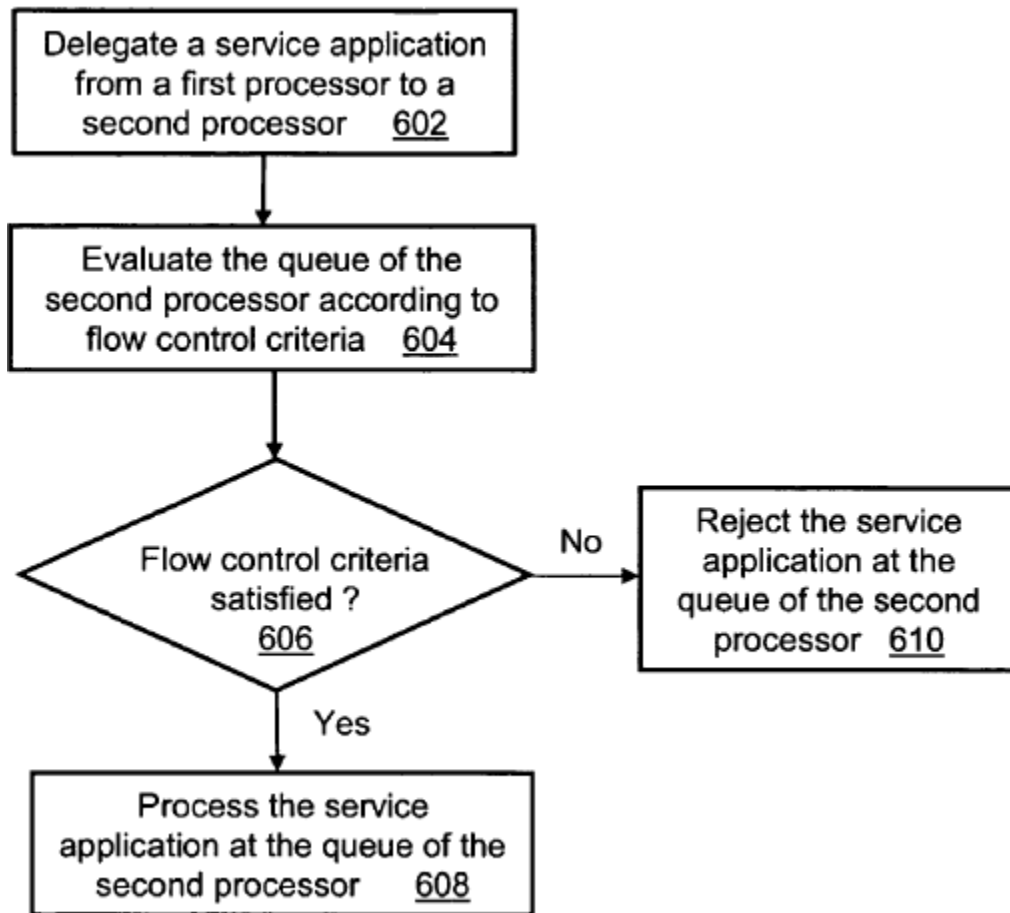
show that elements 7[e]–7[i] and 7[k], and 13[e]–13[i] would be obvious for the same reasons discussed above.

b. Khawand and Rosenthal Did Not Disclose or Render Obvious Limitations 7[g]/[j] and 13[g]/[j].

Petitioners do not demonstrate that Khawand and Rosenthal disclosed or rendered obvious that a 2PE “receives a command from said first processing element” and “issues said command to said input queue *on behalf of* said 1PE.” EX1001, cl. 7, 13 (emphasis added). There is simply no evidence that IPM 308 can accept or submit commands on behalf of the 1PE, or that it would be obvious to modify Khawand to do so.

Petitioners argue that “[b]efore Khawand’s Inter-Processor Manager submits the command from the first processor, it must first receive the command from the first processor.” Pet., 68. However, Khawand does not disclose IPM 308 “receiving the command” *at all*. EX2020, ¶207. IPM 308 only acts as a flow control device that routes “resume/suspend commands” between execution engines. EX1008, 4:60–67; EX2020, ¶208. Petitioners’ expert, Dr. Russ, acknowledges that issuing a command “on behalf of” another processor requires that the issuing processor receive the command from a different processor. EX2022, 69:1–20. Yet Dr. Russ also admitted that Processor 1 issues commands directly to the queue of Processor 2—not to or through IPM 308. EX2022, 139:11–21; EX1008, 3:65–4:2.

Petitioners cite Figure 6 as evidence that IPM 308 purportedly issues the command from Processor 1 to the input queue of Processor 2, Pet., 69, but Figure 6 shows the opposite: it is Processor 1 that delegates the service application (which Petitioners allege is writing the command)—not IPM 308:



Id., FIG. 6.

Petitioners' arguments on limitations 7[g]/[j] and 13[g]/[j] are made up from whole cloth. Dr. Russ also does not provide any unique opinion on the limitations, instead reciting the same unsupported argument from the Petition. Given that these limitations are plainly missing from Khawand and Rosenthal, and Petitioners do not

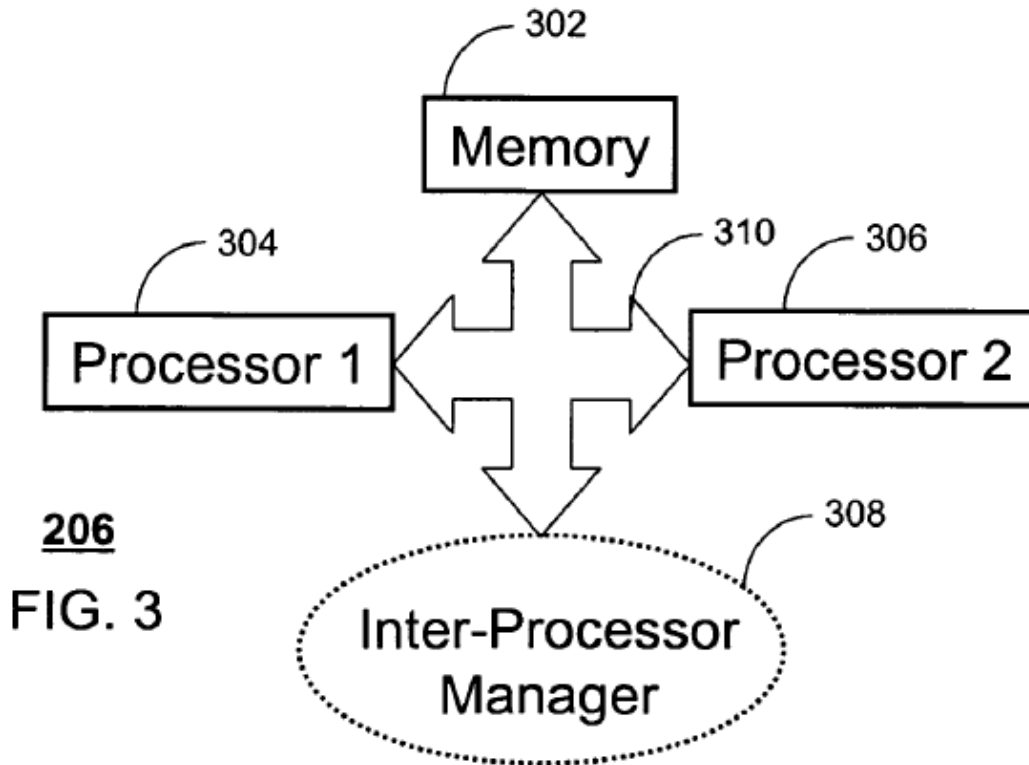
otherwise contented that these limitations would be obvious, Petitioners do not meet their burden of proving unpatentability. And again, Petitioners' request that the Board add in limitations that are missing wholesale from their references is inconsistent with both the plain language of PTAB regulations and binding Federal Circuit caselaw. *See* EX2023; 37 C.F.R. § 42.104(b)(4); *Qualcomm Inc. v. Apple Inc.*, 134 F.4th 1355, 1364-65 (Fed. Cir. 2025)).

c. Khawand and Rosenthal Did Not Disclose or Render Obvious Limitation 13[k].

Petitioners do not demonstrate that Khawand and Rosenthal disclose or rendered obvious a 2PE that “receives a response corresponding to said command from said output queue.” EX1001, cl. 13. IPM 308's only purpose is to signal processing engines whether to suspend or resume commands. EX2020, ¶213. IPM 308 does not offload commands on behalf of another processing element—thus there is *no return data* for IPM 308 to receive. *Id.* As discussed above, it would it not obvious for Processor 1 (which wrote the command) to receive a response regarding that command. *See* § VIII.C.2.b. That is all the truer for IPM 308, which plays no role in the execution of the command at all.

Petitioners contend that “it was obvious to include an output queue for return data from delegated service applications” because IPM 308 “plays a centralized role in the delegation, synchronization, and coordination of service applications.” Pet.,

72. But Khawand is explicit that IPM 308 only plays a centralized role in load-balancing across service applications:



206
FIG. 3

EX1008, FIG. 3.

Petitioners next contend that Khawand suggests IPM 308 “would also be able to access the corresponding output queues” because it “directly requests updates from each input queue.” Pet., 72. But this ignores that IPM 308 only acts as a flow-control mediator between the processing elements and has no role in the system once the tasks have been executed. EX1008, 4:6–14, 5:57–63. Khawand describes that IPM 308 may act as a centralized hub for the processors within the system to track whether each other processor has capacity to load-balance a task. *Id.*

Petitioners effectively argue that it would be obvious for IPM 308 to take on the new, undisclosed role of receiving updates from output queues, which it would then pass via different output queues back to the host processor. Pet., 72. Even if it were technically possible to redesign Khawand to give IPM 308 access to any undisclosed output queues, there is no teaching or suggestion that a POSITA would do this because it would only harm the efficiency of the disclosed system by adding new, inconsistent roles to IPM 308. EX2020, ¶217.

In fact, Petitioners undercut their argument in their discussion of claims 6, 12, and 18. Each of those claims requires that “at least one third processing element executes said command and notifies said output queue to issue said response.” EX1001, cl. 6, 12, 18. Petitioners contend that when the third processing element executes the commands, it provides its results *back to the first processor* (i.e., “the point of delegation”)—not to IPM 308. Pet., 66. Even more damaging to their theory, Petitioners go on to allege that it would have been obvious to a POSITA that the output queue notifies the *first processor* of the result through a “ready” or “queue not empty” flag. Pet., 66 (“Accordingly, it was obvious for Processor 2 to provide the results from the delegated command to the output queue, and then notify the output queue *to provide the response back to the first processor*”) (emphasis added). Thus, Petitioners appear to agree that it would not be obvious for IPM 308 to receive the results back from the output queue. Pet., 66.

3. Dependant claims

a. Khawand and Rosenthal Did Not Disclose or Render Obvious Claims 6 and 12.

Petitioners contend that Khawand discloses that “at least one third processing element executes said command and notifies said output queue to issue said response” because Processor 2 purportedly “provides its results back” to Processor 1 after executing the service application. Pet., 66. Petitioners do not provide additional support for this argument beyond what they discussed for claim 1[j]. Pet., 66 (citing argument on limitation 1[j]). Thus, their argument fails for the same reasons as above: Khawand does not disclose that Processor 2 sends results back to Processor 1 because Processor 1 ceded all responsibility when it delegated to Processor 2. EX1008, 4:2–5, 6:5–13; EX2020, ¶220. Because Khawand does not teach disclose or suggest any output queues sending a response to Processor 1, it also does not teach that this response comes from the “third processing element.”

b. Petitioners Do Not Even Allege Claim 18 Would Be Obvious Over Khawand and Rosenthal.

Petitioners address claim 18 in combination with claims 6 and 12, but in doing so fail to actually allege claim 18 is unpatentable. In particular, Petitioners argue that “[a]ccordingly, it was obvious for Processor 2 to provide the results from the delegated command to the output queue, and then notify the output queue to provide the response back to the first processor.” Pet., 65–66. As explained above, that is incorrect. But even if it were correct, that does not address claim 18, which requires

that the response be sent back to *IPM 308* (the alleged 2PE)—not the “IPM 308. Petitioners fail to provide any argument, analysis or evidence—or to even assert—that claim 18 would be obvious. *Id.* The Petition thus fails to prove claim 18 is unpatentable.

4. Ground 4 Does Not Address The Problems In Ground 3.

Petitioners present an alternative theory using the Ohkawa reference to demonstrate that hardware output queues were obvious by 2013. Pet., 72. Petitioners rely on Ohkawa simply to disclose “output queues,” and do not rely on Ohkawa to teach or suggest any of the limitations challenged above. Ohkawa thus does not address the deficiencies in Ground 3.

5. Ground 5 Does Not Address The Problems In Ground 3.

In ground 5, Petitioners rely on Matsunami only for the limitations recited in claims 3–4, 9–10, 15–16. Petitioners do not rely on Matsunami to teach or suggest any of the limitations challenged above. Matsunami thus does not address the deficiencies in Ground 3.

IX. CONCLUSION

For the reasons stated above, the Board should find that Petitioners failed to carry their burden of proving unpatentability as to all grounds in the Petition.

Case IPR2025-00478

Patent No. 8,924,596

Dated: January 5, 2026

By: /s/ Michael Matulewicz-Crowley

Michael Matulewicz-Crowley

PTO Reg. No. 72,220

Counsel for Patent Owner

CERTIFICATE OF COMPLIANCE

The undersigned hereby certifies that Patent Owners' Response (the "POR") consists of 13,805 words, and 18 words in images, excluding the table of contents, certificate of service, this certificate of compliance, or table of exhibits. The POR complies with the type-volume limitation of 14,000 words as mandated in 37 C.F.R. § 42.24. In preparing this certificate, counsel has relied on the word count of the word-processing system used to prepare this paper (Microsoft Word).

Dated: January 5, 2026

/s/ Michael Matulewicz-Crowley
Michael Matulewicz-Crowley
Counsel for Patent Owner

CERTIFICATE OF SERVICE

Pursuant to 37 C.F.R. § 42.6, the undersigned certifies that on January 5, 2026,
a copy of the foregoing document was served by email upon the following:

Harper Batts (Reg. No. 56,160)
hbatts@sheppardmullin.com
legaltm-amd-concurrent@sheppardmullin.com
SHEPPARD MULLIN LLP
1540 El Camino Real, Suite 120
Menlo Park, CA 94025
T 650-815-2673 / F 650-815-4668

Christopher Ponder (Reg. No. 77,167)
cponder@sheppardmullin.com
legaltm-amd-concurrent@sheppardmullin.com
SHEPPARD MULLIN LLP
1540 El Camino Real, Suite 120
Menlo Park, CA 94025
T 650-815-2676 / F 650-815-4664

Jeffrey Liang (Reg. No. 69,043)
jliang@sheppardmullin.com
legaltm-amd-concurrent@sheppardmullin.com
SHEPPARD MULLIN LLP
1540 El Camino Real, Suite 120
Menlo Park, CA 94025
T 650-815-2685 / F 650-815-4666

/s/ Michael Matulewicz-Crowley
Michael Matulewicz-Crowley
Counsel for Patent Owner

CLAIM APPENDIX

The language of independent claims 1, 7, and 13 of the '596 patent is reproduced below. For convenience, Patent Owners have included the limitation labels used by Petitioners.

Claim 1:

1[pre] A system for dividing and synchronizing a processing task across a plurality of processing elements comprising:

[1a] an input queue implemented in hardware;

[1b] an output queue implemented in hardware;

[1c] a first processing element having access to said input queue and said output queue;

[1d] at least one second processing element in communication with said first processing element;

[1e] a reservation register implemented in hardware storing a value indicative of available space in said input queue, said reservation register accessible by both said first processing element and said at least one second processing element;

[1f] computer storage storing instructions, which when executed by said at least one second processing element:

[1g] accesses said reservation register and reads said stored value;

[1h] determines when said read value indicates available space in said input queue for said first processing element to issue a command;

[1i] notifies said first processing element to issue said command to said input queue; and

[1j] wherein said first processing element receives notification from said at least one second processing element regarding issuing the command, issues said command to said input queue, and receives a response corresponding to said command from said output queue.

EX1001, cl. 1; *see also* Pet., 92–93 (claim appendix).

Claim 7:

[7pre] A system for dividing and synchronizing a processing task across a plurality of processing elements comprising:

[7a] an input queue implemented in hardware;

[7b] an output queue implemented in hardware;

[7c] a first processing element having access to said input queue and said output queue;

[7d] at least one second processing element in communication with said input queue;

[7e] a reservation register implemented in hardware storing a value indicative of available space in said input queue, said reservation register accessible by both said first processing element and said at least one second processing element;

[7f] computer storage storing instructions, which when executed by said at least one second processing element:

[7g] receives a command from said first processing element;

[7h] accesses said reservation register and reads said stored value;

[7i] determines when said read value indicates available space in said input queue to issue said command;

[7j] issues said command to said input queue on behalf of said first processing element; and

[7k] wherein said first processing element receives a response corresponding to said command from said output queue.

EX1001, cl. 7; *see also* Pet., 93–95 (claim appendix).

Claim 13:

13[pre] A system for dividing and synchronizing a processing task across a plurality of processing elements comprising:

[13a] an input queue implemented in hardware;

[13b] an output queue implemented in hardware;

[13c] a first processing element having access to said input queue and said output queue;

[13d] at least one second processing element in communication with said input queue and said output queue;

[13e] a reservation register implemented in hardware storing a value indicative of available space in said input queue, said reservation register accessible by both said first processing element and said at least one second processing element;

[13f] computer storage storing instructions, which when executed by said at least one second processing element:

[13g] receives a command from said first processing element;

[13h] accesses said reservation register and reads said stored value;

[13i] determines when said read value indicates available space in said input queue to issue said command;

[13j] issues said command to said input queue on behalf of said first processing element; and

[13k] receives a response corresponding to said command from said output queue.

EX1001, cl. 13; *see also* Pet., 95–96 (claim appendix).

Petitioners also challenged the dependent claims of the '596 patent. The language of dependent claims 5 and 6 are reproduced below:

Claim 5:

The system of claim 1, wherein said stored value is decremented by a predefined value prior to returning said stored value to said at least one second processing element.

EX1001, cl. 5; *see also* Pet., 93 (claim appendix).

Claim 6:

The system of claim 1, wherein at least one third processing element executes said command and notifies said output queue to issue said response.

EX1001, cl. 5; *see also* Pet., 93 (claim appendix)