

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent of: Bismuth et al.
U.S. Patent No.: 11,886,750 Attorney Docket No. 54788-0024IP1
Issue Date: January 30, 2024
Appl. Serial No.: 18/075,576
Filing Date: December 6, 2022
Title: ANALYTICS, ALGORITHM ARCHITECTURE, AND DATA
PROCESSING SYSTEM AND METHOD

Mail Stop Patent Board

Patent Trial and Appeal Board
U.S. Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

**PETITION FOR *INTER PARTES* REVIEW OF UNITED STATES PATENT
NO. 11,886,750 PURSUANT TO 35 U.S.C. §§ 311–319, 37 C.F.R. § 42**

TABLE OF CONTENTS

I.	REQUIREMENTS FOR IPR	1
A.	Grounds for Standing	1
B.	Challenge and Relief Requested	1
II.	THE '750 PATENT	2
A.	Overview	2
III.	LEVEL OF ORDINARY SKILL	4
IV.	CLAIM CONSTRUCTION	5
V.	THE CHALLENGED CLAIMS ARE UNPATENTABLE.....	5
A.	GROUND 1A: Dalal-Ng Renders Obvious Claims 1–2, 5–9.....	5
1.	Dalal	5
2.	Ng	8
3.	Dalal-Ng Combination	10
4.	Analysis	15
B.	GROUND 1B: Dalal-Ng-Raghava Renders Obvious Claims 3–4	53
1.	Raghava	53
2.	Dalal-Ng-Raghava Combination.....	54
3.	Analysis	56
C.	GROUND 2A: Heil-Jun Renders Obvious Claims 1–3, 5–8.....	59
1.	Heil	59
2.	Jun.....	67
3.	Heil-Jun Combination	68
4.	Analysis	72
D.	GROUND 2B: Heil-Jun-Raghava Renders Obvious Claims 4, 9.....	103
1.	Heil-Jun-Raghava Combination.....	103
2.	Analysis	107
VI.	PAYMENT OF FEES	108
VII.	MANDATORY NOTICES—37 C.F.R. § 42.8(a)(1)	108
A.	Real Party-In-Interest—37 C.F.R. § 42.8(b)(1)	108
B.	Related Matters—37 C.F.R. § 42.8(b)(2)	108
C.	Lead And Backup Counsel—37 C.F.R. § 42.8(b)(3).....	108
D.	Service Information	109

EXHIBITS

AMD1001	U.S. Patent No. 11,886,750 to Bismuth et al. (“the ’750 patent”)
AMD1002	Excerpts from the Prosecution History of the ’750 Patent (“the Prosecution History”)
AMD1003	Declaration of Dr. Donald Alpert
AMD1004	U.S. Publication No. 2019/0109793 (“Dalal”)
AMD1005	U.S. Publication No. 2019/0114538 (“Ng”)
AMD1006	U.S. Publication No. 2019/0108145 (“Raghava”)
AMD1007	[RESERVED]
AMD1008	U.S. Patent No. 9,043,526 (“Freking”)
AMD1009	[RESERVED]
AMD1010	U.S. Publication No. 2016/0306668 (“Heil”)
AMD1011	Jun et al. “BlueDBM: An Appliance for Big Data Analytics” ACM (June 13, 2015) (“Jun”)
AMD1012	Liu et al. “BlueFlash : a reconfigurable flash controller for BlueDBM” Massachusetts Institute of Technology (2014) (“Liu”)
AMD1013	Declaration of June Ann Munford
AMD1014	Putnam et al. “A Reconfigurable Fabric for Accelerating Large- Scale Datacenter Services” Microsoft, (2014)

AMD1015	Vipin et al. “FPGA Dynamic and Partial Reconfiguration: A Survey of Architectures, Methods, and Applications” ACM Comput. Surv. Vol. 51, No. 4, Article 72 (July 2018)
AMD1016	[RESERVED]
AMD1017	[RESERVED]
AMD1018	Verma et al. “PCIe BUS: A State-of-the-Art-Review” (2017)
AMD1019	[RESERVED]
AMD1020	U.S. Patent No. 11,169,727 (“Doucette”)
AMD1021	U.S. Patent No. 10,802,995 (“Singh”)
AMD1022	Park et al. “PFFS: A Scalable Flash Memory File System for the Hybrid Architecture of Phase-change RAM and NAND Flash” (2008)
AMD1023	Prasad et al. “Design and Verification of Phy Interface for PCIe Gen 3.0 and USB Gen 3.1 Using UVM Methodology” (2017)
AMD1024	U.S. Patent No. 11,055,252 (“Constantinides”)
AMD1025	“AXI Central Direct Memory Access v4.1: LogiCORE IP Product Guide” (April 4, 2018)
AMD1026	“PCI Express Base Specification Revision 1.0” (April 29, 2002)
AMD1027	IEEE “IEEE Standard for In-System Configuration of Programmable Devices” IEEE Std 1532™ (2002)
AMD1028	U.S. Patent No. 10,831,376 (“Jung”)

AMD1029	Jung et al. “HIOS: A Host Interface I/O Scheduler for Solid State Disks” (2014)
AMD1030	IEEE “The Authoritative dictionary of IEEE Standards Terms, Seventh Edition” (2000)
AMD1031	U.S. Patent No. 9,152,592 (“Galles”)
AMD1032	McCarthy et al. “Visualizing the Five-dimensional Torus Network of the IBM Blue Gene/Q” (2014)
AMD1033	“Serial ATA: High Speed Serialized AT Attachment” Revision 1.0a (January 7, 2003)
AMD1034	U.S. Patent No. 6,711,701 (“Roohparvar”)
AMD1035	U.S. Publication No. 2013/0318285 (“Pignatelli”)
AMD1036	U.S. Patent No. 9,678,842 (“Zhang”)
AMD1037	Ousterhout et al. “The Case for RAMClouds: Scalable High-Performance Storage Entirely in DRAM” SIGOPS Operating Systems Review, Vol. 43, No. 4, December 2009, pp. 92-105.
AMD1038	Kang et al. “Performance Trade-Offs in Using NVRAM Write Buffer for Flash Memory-Based Storage Devices” IEEE Transactions on Computers, Vol. 58, No. 6, June 2009, pp. 744-758.
AMD1039	Zazo et al. “A PCIe DMA engine to support the virtualization of 40 Gbps FPGA-accelerated network appliances” 2015 International Conference on ReConFigurable Computing and FPGAs (ReConFig) (2015).
AMD1040	[RESERVED]
AMD1041	[RESERVED]

AMD1042	[RESERVED]
AMD1043	U.S. Publication No. 2020/0233588 (“Li”)
AMD1044	Voluntary Search Disclosure Declaration of Kenneth Darby
AMD1045	[RESERVED]
AMD1046	U.S. Patent No. 8,527,949 (“Pleis”)
AMD1047	Toulson et al. “Fast and Effective Embedded Systems Design: Applying the ARM mbed. 2nd ed.” Newnes (2017)
AMD1048	[RESERVED]

LISTING OF CLAIMS

Claim 1	
[1pre]	A data processing system operative in cooperation with a host compute system, the data processing system comprising:
[1a]	a router module comprising a host interface and a node interface, wherein the router module is communicatively coupled to the host compute system via the host interface; and
[1b]	multiple parallel first tier compute nodes coupled to one or more additional compute nodes in series, each compute node comprising:
[1c]	a communications link, wherein each first tier compute node is communicatively coupled to the node interface at the router module via the communications link, at different successive tiers, each compute node in a series being communicatively coupled, via its communications link to one or more additional compute nodes connected at different successive tiers;
[1d]	a data store comprising records associated with a data processing operation;
[1e]	a programmable logic component to execute the data processing operation in cooperation with the data store;
[1f]	a node memory comprising data and instructions to support operation of the programmable logic component in connection with the data processing operation and to facilitate data communications via the communications link;
[1g]	a data mover component to facilitate data communications between the programmable logic component and the node memory; and
[1h]	a storage interface component to facilitate data communications between the programmable logic component and the data store;
[1i]	wherein the storage interface component utilizes a plurality of communications channels to transfer data between the programmable logic component and the data store.

Claim 2	
[2]	The data processing system of claim 1 wherein the communications link employs a serial communications protocol.
Claim 3	
[3]	The data processing system of claim 1 wherein the data store comprises an Electronically Erasable Programmable Read Only Memory device.
Claim 4	
[4]	The data processing system of claim 3 wherein the storage interface component employs [an] Open NAND Flash Interface protocol.
Claim 5	
[5]	The data processing system of claim 1 wherein the programmable logic component comprises a field programmable gate array.
Claim 6	
[6]	The data processing system of claim 1 further comprising a processor to manage operation of the data mover component.
Claim 7	
[7]	The data processing system of claim 6 wherein the data mover component comprises a device controller.
Claim 8	
[8]	The data processing system of claim 1 wherein the host compute system executes instructions to influence operation of the compute node.
Claim 9	
[9]	The data processing system of claim 1 wherein the router module and the compute node are integrated into an interface card communicatively coupled to the host compute system.

CONVENTIONS OF THE PETITION

- All emphases are added unless noted otherwise;
- Bold-italic emphases correlate to claim language;
- Quotations are generally from exhibits, not claim language; and
- The phrase “as discussed” and equivalent phrases incorporate fully the analysis of the cross-cited portion of the Petition.

Advanced Micro Devices, Inc. (“Petitioner”) petitions for IPR of claims 1–9 (“Challenged Claims”) of U.S. Patent No. 11,886,750 (“the ’750 patent”) based on compelling evidence of obviousness presented throughout this Petition. AMD1003, ¶29.

I. REQUIREMENTS FOR IPR

A. Grounds for Standing

Petitioner certifies that the ’750 patent is available for IPR, and that Petitioner is not barred or estopped from requesting IPR. *See* §VII.B.

B. Challenge and Relief Requested

Petitioner requests IPR based on the grounds noted below.

ID	Claims	Prior Art Basis Under §103
1A	1–2, 5–9	Dalal-Ng
1B	3–4	Dalal-Ng-Raghava
2A	1–3, 5–8	Heil-Jun
2B	4, 9	Heil-Jun-Raghava

Petitioner does not concede that the ’750 patent is entitled to the benefit of its priority claim. Petitioner has used February 6, 2019, as the earliest possible priority date (“Critical Date”) of the Challenged Claims. Each reference applied in the Grounds listed above qualifies as prior art under 35 U.S.C. §§102 or 103.

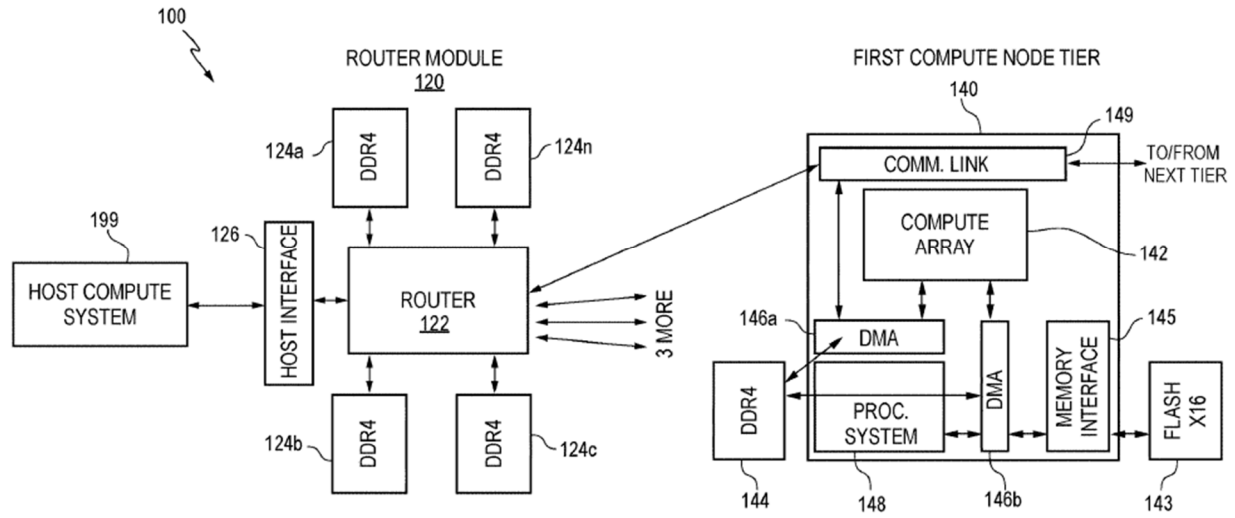
Reference	Filing Date	Publication Date
Dalal- (AMD1004)	2018-09-12	2019-04-11
Ng- (AMD1005)	2017-10-17	2019-04-18
Raghava- (AMD1006)	2018-12-06	2019-04-11
Heil- (AMD1010)	2015-05-20	2016-10-20
Jun- (AMD1011)	—	2015-06

II. THE '750 PATENT

A. Overview

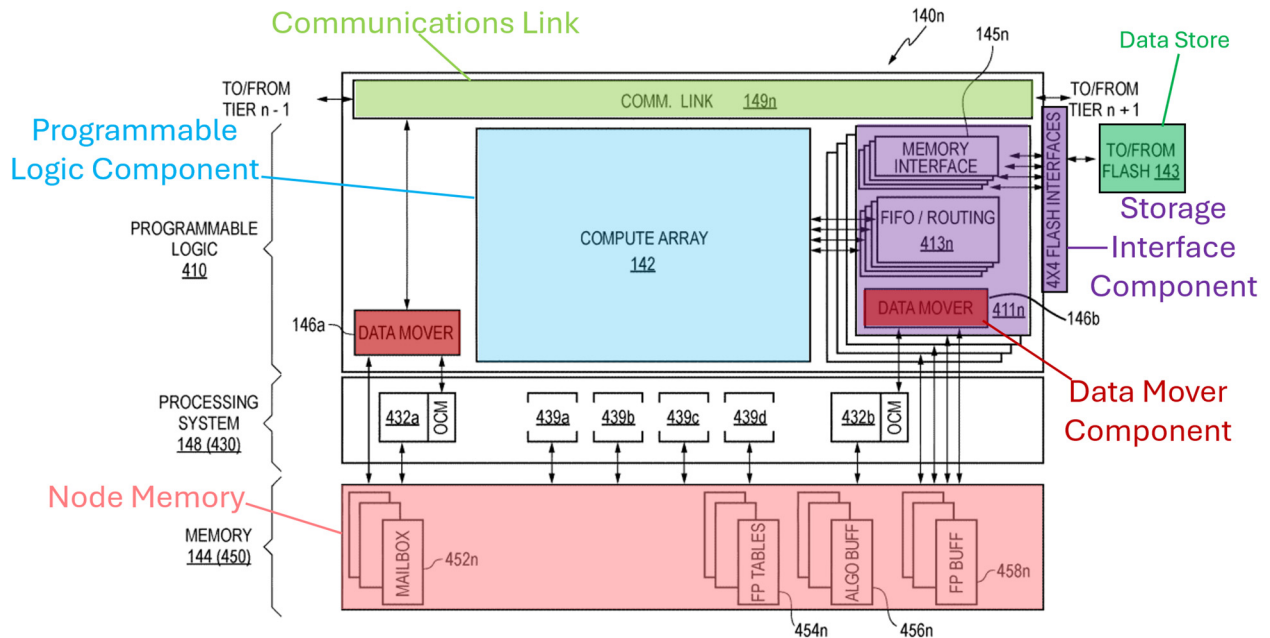
The '750 patent describes a “data processing system” that includes multiple nodes performing operations. AMD1001, Title, 7:4–22, FIG. 1. A first tier compute node receives input data, executes an operation, and passes the resulting data to the node at a successive tier. *Id.*, 43:46–44:46, FIGS. 1, 4, 22; AMD1003, ¶¶22–25. The architecture can include four “parallel lanes” each with multiple compute nodes “connected in series (i.e., at different successive tiers).” *Id.*, 9:61–10:3, FIGS. 1, 22.

FIG. 1



AMD1001, FIG. 1.

A node (*e.g.*, 140n shown in Figure 4 below) in the pipeline includes programmable logic, data store, storage interface, data mover, node memory, and communications link. AMD1001, 9:54–10:58, 54:21–57, FIGS. 1, 4; AMD1003, ¶23.



AMD1001, FIG. 4.¹

III. LEVEL OF ORDINARY SKILL

A person of ordinary skill in the art (“POSITA”) relating to the subject matter of the ’750 patent as of the Critical Date would have had a bachelor’s degree in Computer Engineering, Electrical Engineering, Computer Science, or a comparable field and at least two years of experience related to computer architecture. AMD1003, ¶¶27–28. Additional experience could substitute for a formal degree or formal training (and *vice versa*). *Id.*

¹ Annotations to the figures throughout this petition are shown in color.

IV. CLAIM CONSTRUCTION

The Challenged Claims are obvious under any reasonable interpretation advanced pursuant to the *Phillips* standard. Although the Board need not resolve any express constructions to adjudicate this Petition, Petitioner nevertheless identifies the claim term “programmable logic component” as a “logic component programmable by reconfiguring its internal hardware connections, such as an FPGA.” AMD1003, ¶17. This interpretation is supported by the intrinsic record and consistent with the ordinary meaning. AMD1001, 15:3–13, 39:5–9, 19:55–62.

V. THE CHALLENGED CLAIMS ARE UNPATENTABLE

A. GROUND 1A: Dalal-Ng Renders Obvious Claims 1–2, 5–9

1. Dalal

Dalal² describes a system including servers connected in a distributed computing environment. AMD1004, [0026]–[0048], FIGS. 1–2C; AMD1003, ¶30. Figure 1 (below) shows a server in Dalal’s system that includes hardware accelerator modules (hwa modules) that include computing elements that process data. *Id.* Each

² General descriptions provided for this and other references and combinations thereof are incorporated into each subsection and mapping of the claims that includes citations to these references.

server includes “server switches” coupled to the hwa modules and a host processor.

Id.

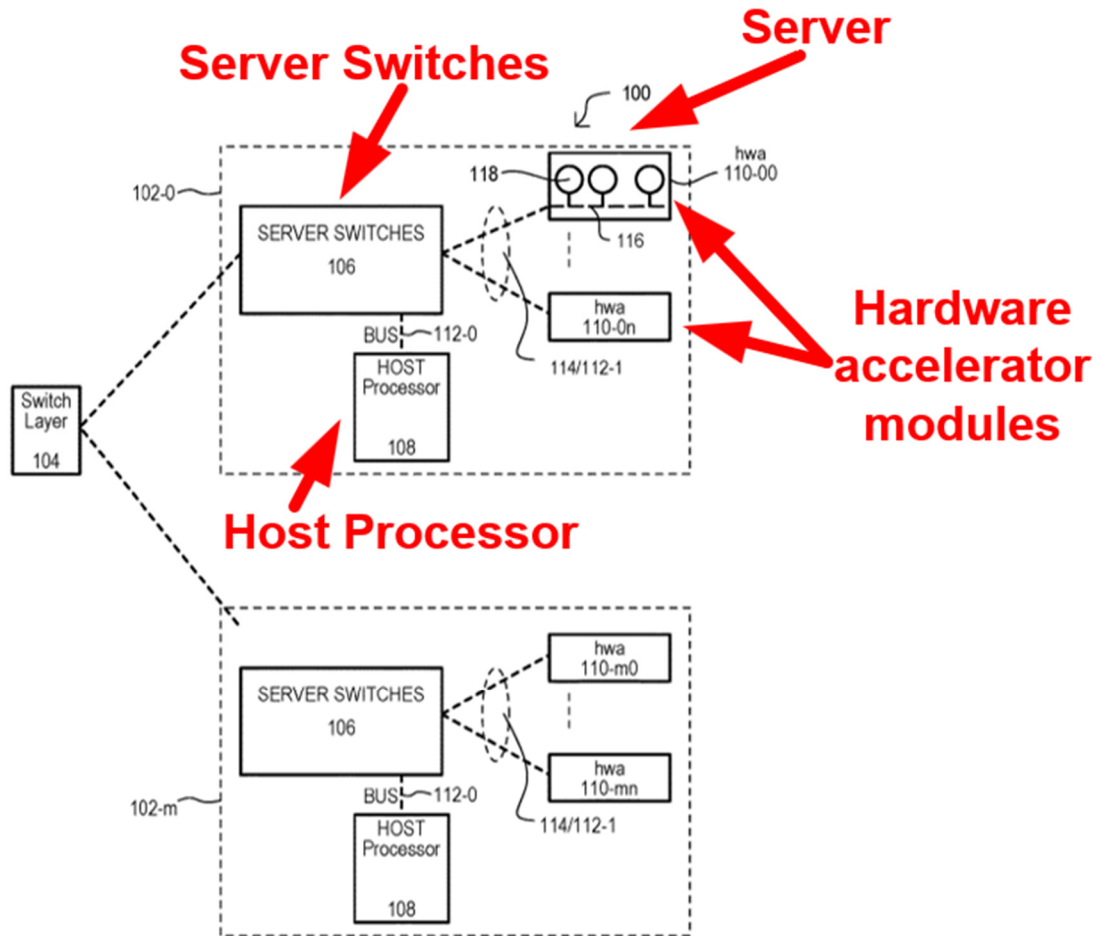


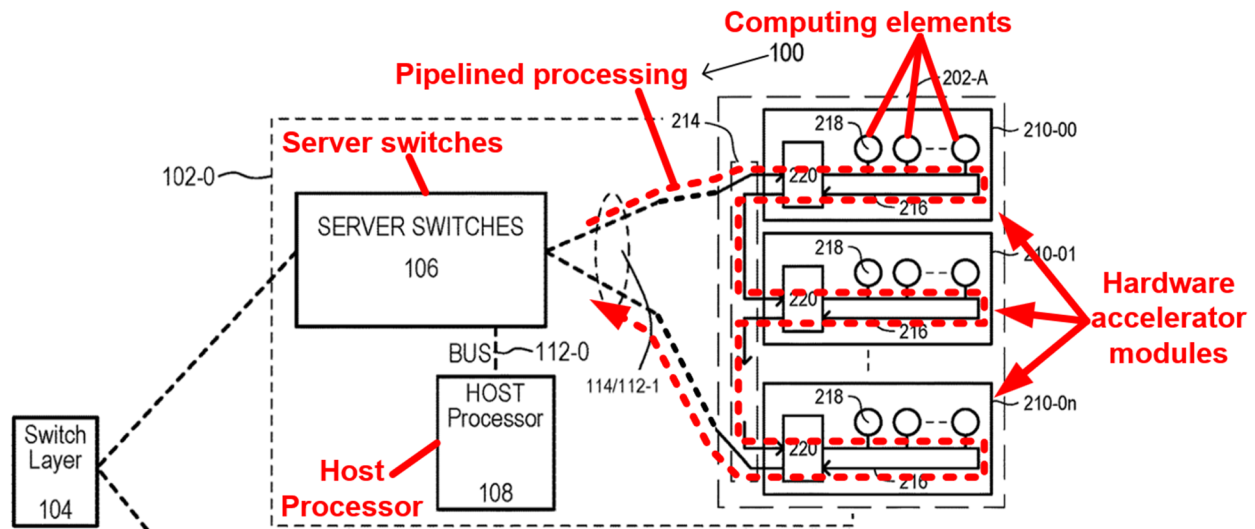
FIG. 1

AMD1004, FIG. 1.

Dalal’s “FIG. 2A to 2C show various systems, which...can be particular versions of system 100 of FIG. 1.” AMD1004, [0039]. The below drawing is a combined figure that integrates the system of Figure 1 with the “particular version” of Figure 2B. AMD1003, ¶31. In the combined figure, the hwa modules 210 in box

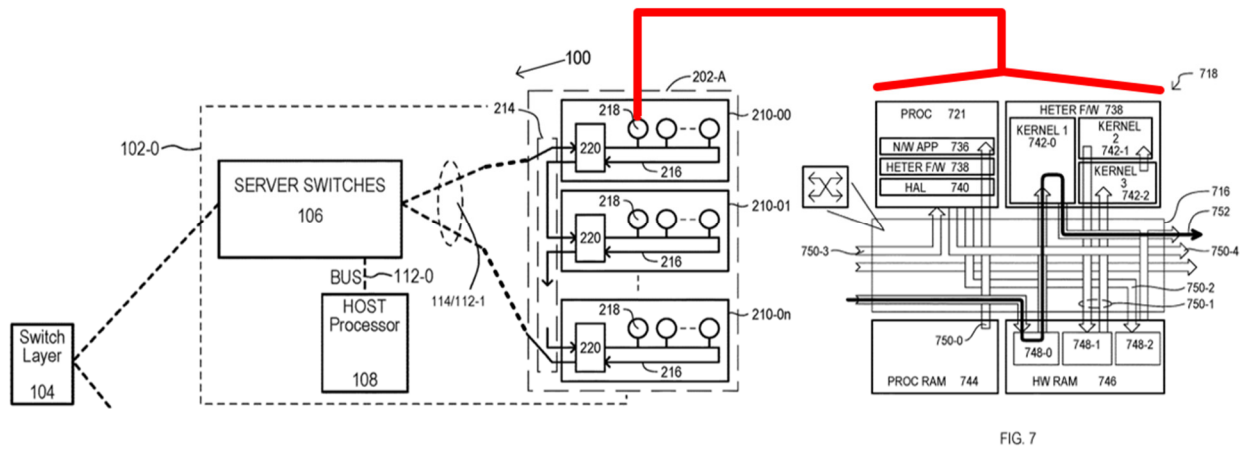
202-A from Figure 2B replace the hwa modules 110 shown in Dalal's original Figure

1. In the combined figure (below), each server includes hwas that are serially coupled to one another so that the modules process data sequentially. AMD1004, [0035]–[0048], FIGS. 2A–2C; AMD1003, ¶32.



AMD1004, FIGS. 1, 2B Combination Figure.

The computing elements located in Dalal’s hwa modules include components used for processing data according to a data processing operation (e.g., machine learning), including processing circuits (*e.g.*, programmable logic), a first memory, and a second memory. AMD1004, [0075]–[0088], FIG. 7; AMD1003, ¶33.



AMD1004, FIGS. 1, 2B Combination Figure; AMD1004, FIG. 7.

2. Ng

Ng describes a host computer system that communicates with a hardware accelerator to perform complex data processing operations. AMD1005, [0025]–[0042], FIGS. 1–4; AMD1003, ¶34. The hardware accelerator includes a programmable integrated circuit (IC) (*e.g.*, a field programmable gate array (FPGA)) that communicates with memory and a host computer system. AMD1005, [0032]–[0039], FIGS. 2–4.

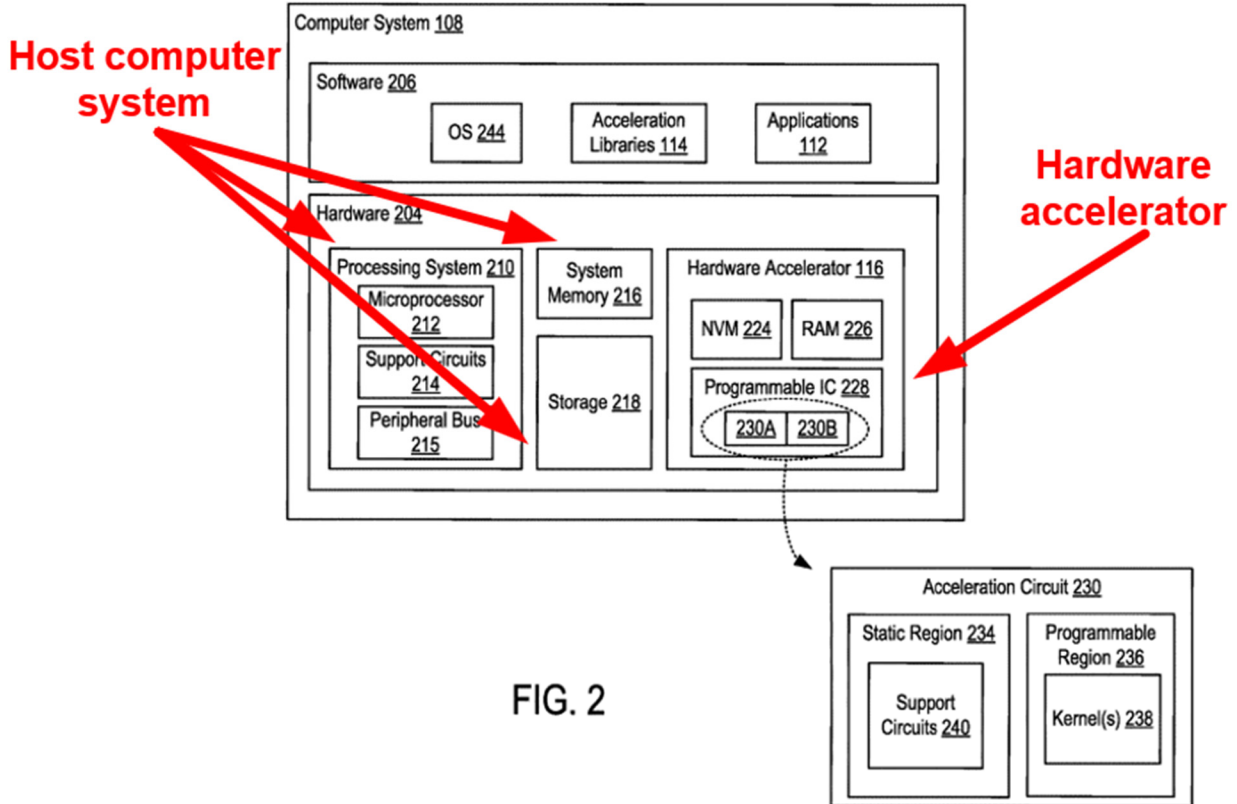
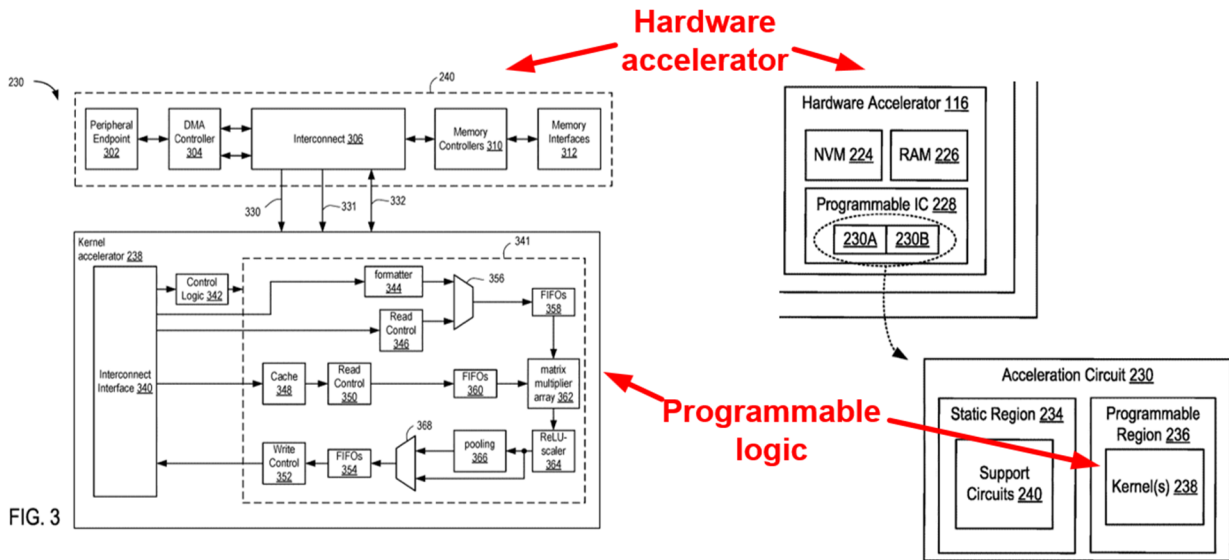


FIG. 2

AMD1005, FIG. 2.

The hardware accelerator performs complex data processing operations to execute multiple layers of a neural network. AMD1005, [0007]–[0008], [0021]–[0046], FIGS. 1–4; AMD1003, ¶35.



AMD1005, FIGS. 2, 3.

3. Dalal-Ng Combination

Dalal and Ng are analogous art to the '750 patent. AMD1003, ¶36. The '750 patent is directed to “a distributed hardware architecture” that “facilitate[s] high throughput data processing and analytics solutions” such as “machine learning and artificial intelligence.” AMD1001, Abstract, 1:35–53, 2:1–14, 47:46–48:7, 49:4–22. Dalal’s “network architecture” includes “hardware accelerator (hwa) modules” arranged to perform complex operations from a “machine learning library, such as TensorFlow,” which falls within the same field of endeavor as the '750 patent. AMD1004, [0026], [0032]–[0036], [0040]–[0050], [0078], FIGS. 1, 2B–2C; *supra* §V.A.1. Ng involves a host computer system that provides tailored instructions to a single hardware accelerator for executing every layer of a neural network and therefore (1) is within the same field of endeavor as the '750 patent and (2) describes

solutions relevant to one or more problems addressed by the '750 patent. AMD1005, [0021]–[0022], [0024], [0039]–[0042], [0044]–[0046], FIGS. 2–4; AMD1003, ¶36.

A POSITA would have been motivated to modify Dalal by incorporating Ng's:

- (1) host computer system;
- (2) programmable neural network accelerator;
- (3) memory interfaces; and
- (4) memory storing records such as matrices and work requests.

AMD1005, [0025]–[0051]; AMD1003, ¶37. An example illustration of how a POSITA would have combined Dalal and Ng is provided below. AMD1003, ¶37. In the Combination, Dalal's host processors are implemented with the host computer system, and each one of Dalal's hwa modules are implemented with Ng's hardware accelerator. AMD1003, ¶37. Doing so preserves Dalal's pipelined architecture and supplements Dalal's servers with aspects of the host-accelerator cooperation described in Ng. *Id.*

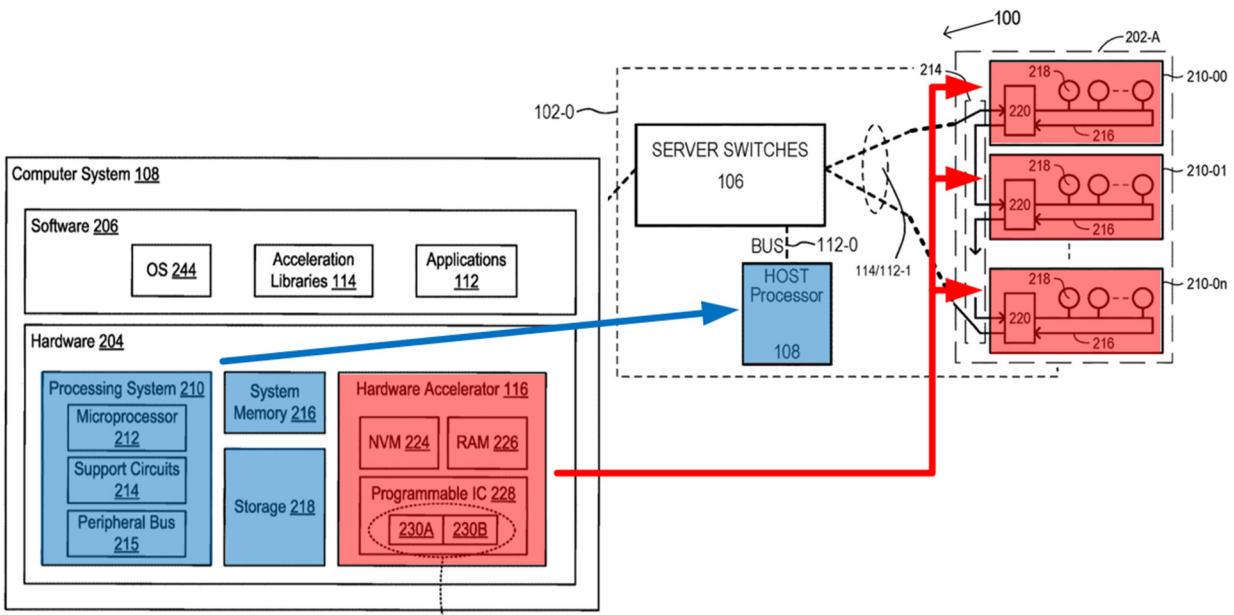


Illustration of a Portion of the Dalal-Ng Combination.

The following reasons support the Dalal-Ng Combination (DNC).
AMD1003, ¶38.

First, a POSITA would have implemented Dalal's host processor with Ng's host computer system that coordinates with a hardware accelerator to manage complex machine learning operations. AMD1003, ¶39. For example, Dalal's hardware accelerators use "pipelined processing" to sequentially process packet data. AMD1004, [0035]–[0036], [0042]; AMD1003, ¶39. Ng describes an effective division of labor between the host and hardware accelerator, whereby the accelerator performs computations for implementing each layer of a neural network using a hardware accelerator and the host "need only provide the control parameters to the neural network accelerator." AMD1005, [0007], [0023]; AMD1003, ¶39. Dalal

acknowledges that “processing circuits” of the hwa modules “consume less power than a host processor,” and Ng recognizes that concentrating “as much processing as possible in the neural network accelerator” will “improve system performance” by reducing an amount of host-accelerator communication. AMD1004, [0063], [0077]; AMD1005, [0023]; AMD1003, ¶39.

Second, a POSITA would have understood that implementing Dalal’s system with Ng’s design tools, such as a user interface for programming the hardware accelerators, would have allowed a developer to customize Dalal’s pipelined processing to achieve specialized goals. AMD1005, [0027]; AMD1003, ¶40. Because hardware accelerators include programmable logic, these accelerators would have been programmable to perform custom operations more efficiently than a general processor (*e.g.*, a CPU or GPU) could have performed these tasks, while consuming less power in the process. AMD1005, [0006]; AMD1004, [0063], [0077]; AMD1003, ¶40.

Third, configuring programmable logic of Dalal’s hwa modules to perform computations such as “matrix multiplication” and “convolution,” as described by Ng, would have provided an improved method for executing “machine learning applications,” which Dalal cites as an example operation performed by the hwa modules. AMD1005, [0021], [0039], [0041]–[0042]; AMD1004, [0078]; AMD1003, ¶41. Configuring Dalal’s hwa modules to perform convolution and

matrix multiplication would have positioned these modules to efficiently execute layers of a neural network and therefore implement machine learning applications. *Id.*

Fourth, memory interface circuits manage communication between programmable logic and memory. AMD1003, ¶42. Implementing Dalal's hwa modules with a memory interface circuit that operates according to a protocol of a coupled memory would have beneficially allowed the modules to ensure that outgoing data is arranged to be received by the memory and received data is identified and processed quickly. *Id.*

Fifth, it would have been obvious for a POSITA to implement Dalal's hardware accelerators with a cooperative host, as described by Ng, because doing so represents one of a finite number of predictable solutions for governing a data processing operation performed by accelerators. AMD1003, ¶43. A POSITA would have appreciated, for example, that "programmable" logic would have been programmed by a developer, and it would have been predictable to do so via a host computer system. *Id.*

Implementing Dalal's hwa modules with a cooperative host and interfaces for communicating with memory merely incorporates well known features that yield predictable results. AMD1003, ¶44. Supplementing Dalal's hwa modules in this

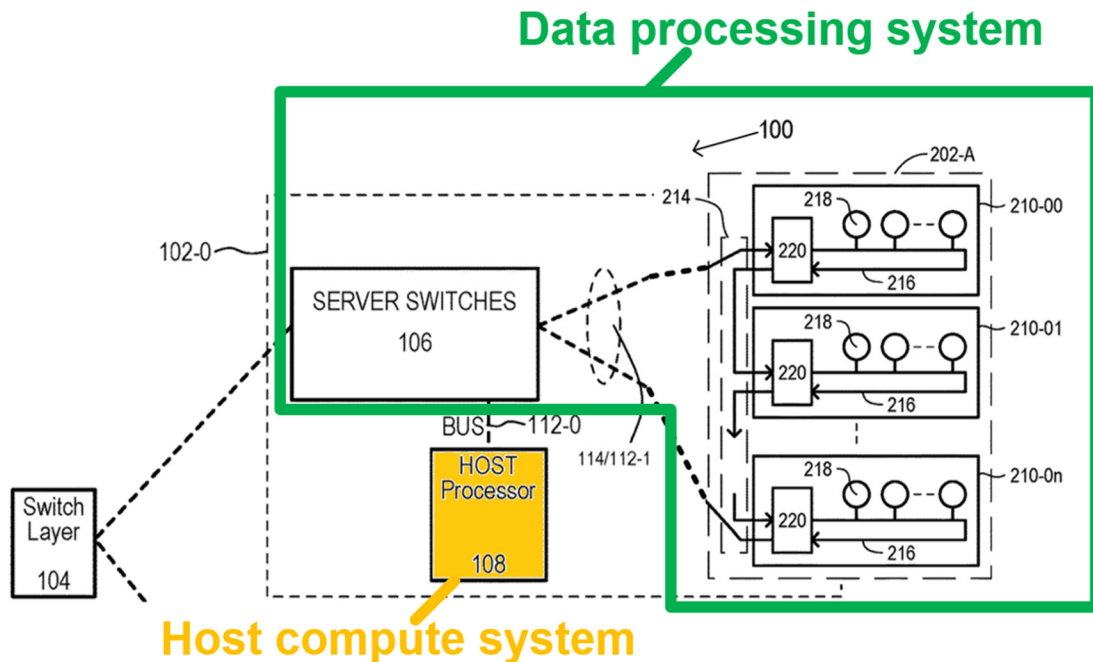
way would have enhanced the system's ability to handle multi-step operations in a systematic manner. *Id.*

Furthermore, combining the teachings of Dalal and Ng would have been obvious and a POSITA would have reasonably expected success, because Dalal's hwa modules process data sequentially just as Ng's hardware accelerator executes each layer of a neural network in order. AMD1004, [0042]–[0047]; AMD1005, [0047]; AMD1003, ¶45. A POSITA would have reasonably expected that using a pipeline of hwa modules to execute a similar multilayer neural network, as described by Ng, would have resulted in a predictable and foreseeable chance of success because both approaches merely involve using the results of one layer as inputs to the next layer. *Id.*

4. Analysis

Element_[1pre]

Dalal's **system** includes one or more servers that each include “one or more **host** processors” (**host compute system**) configured to process data. AMD1004, [0032], [0031]–[0047], [0063], [0069]–[0070], [0073], [0077], [0099], [0101]–[0103], [0123]–[0125], [0170]–[0172], FIGS. 1–2C, 5–6, 12, 18, cl.1; AMD1003, ¶46. As shown below, each server includes server switches 106 coupled to a host processor 108 and hwa modules 210 via buses 112. AMD1004, [0031]–[0038], FIG. 1.



AMD1004, FIGS. 1, 2B Combination Figure³.

Dalal's hwa modules "execute operations on received *data*" from the server switches, indicating that these components are part of a *data processing system*. AMD1004, [0032]–[0036], FIG. 1. The host processor is connected to the server switches via the bus, indicating that Dalal's *data processing system* is *operative in cooperation* with Dalal's *host compute system*. *Id.*; AMD1003, ¶47.

As shown in the figure below, in DNC, Ng's host computer system (*host compute system*) including processing system 210, system memory 216, and storage

³ Recall from §V.A.1 that Dalal explains that Figure 2B embodies a version of Figure 1. AMD1004, [0039]. Thus, Figures 1 and 2B would have been combined in the manner illustrated here consistent with Dalal's teachings. AMD1003, ¶47.

218, are incorporated with Dalal's host processor 108. AMD1005, [0042], [0029], [0043], [0047], [0049], FIGS. 2, 4; AMD1003, ¶¶48–49; *see* §V.A.3. DNC's *host compute system* is *operative in cooperation* with a *data processing system* at least because the “host instructs the neural network accelerator” of the hardware accelerator to perform operations of neural network layers. AMD1005, [0024], [0030]–[0032], [0040]–[0053], FIGS. 2–5; AMD1004, [0032]–[0038], FIG. 1; *supra* §V.A.3.

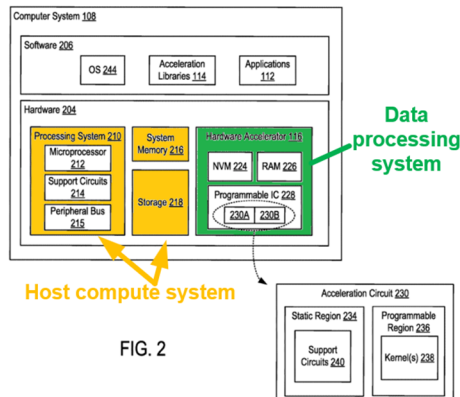


FIG. 2

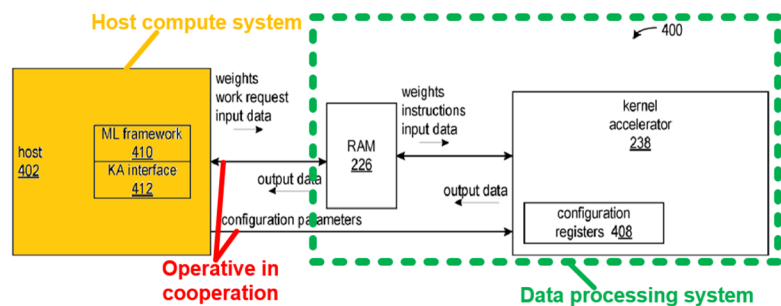


FIG. 4

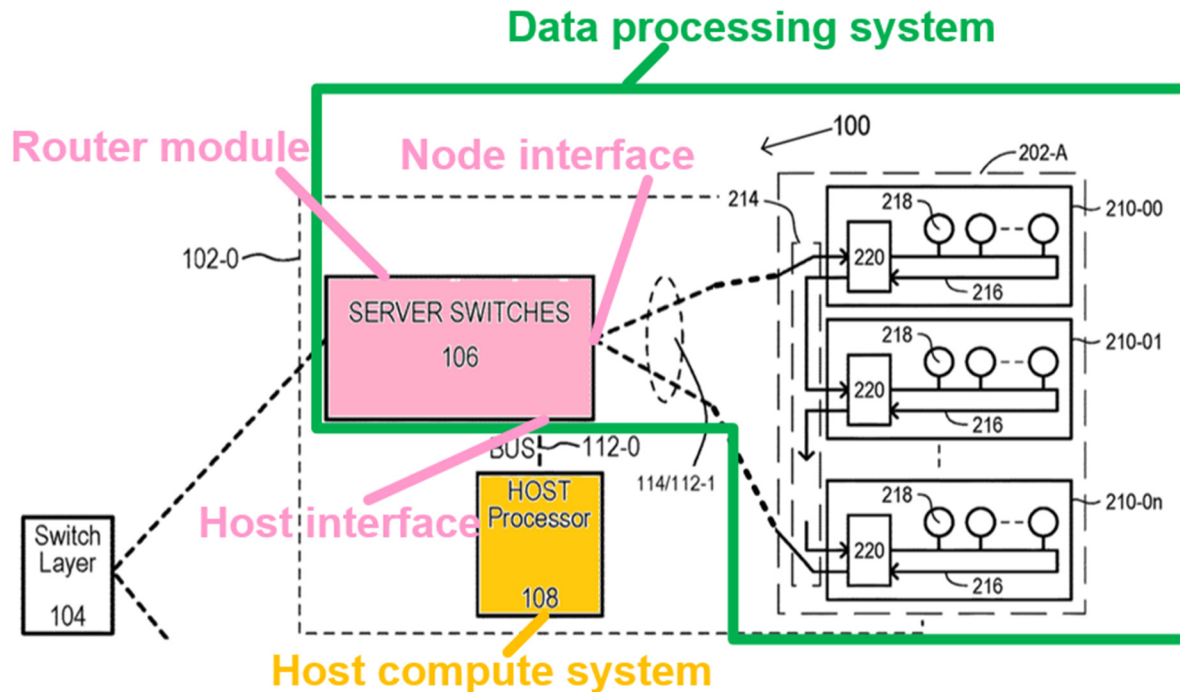
AMD1005, FIGS. 2, 4.

Element [1a]

Dalal's *data processing system* includes “server switches” (*router module*) that perform *routing* operations such as “forward[ing] packet data” to an hwa module “based on information in the packet.” AMD1004, [0032]–[0036], [0123], [0170], FIGS. 1, 12, 18; AMD1003, ¶50. As described above, Dalal's Figure 2B is a “particular version” of the “system...of FIG. 1,” and the hwa modules 210 of Figure 2B are therefore a particular version of the hwa modules 110 of Figure 1.

AMD1004, [0039]; *supra* §V.A.1. The combined figure below demonstrates that Dalal's server switches (*router module*) are connected (*communicatively coupled*) to the host processor (*host compute system*) via a peripheral component interconnect express (PCIe) bus. AMD1004, [0032]–[0036], [0123], [0170]; *supra* Element_[1pre]; AMD1003, ¶51.

A POSITA would have understood that a “*host interface*” would have been used to connect the server switches to the host processor, and a “*node interface*” would have been used to connect the server switches to the various compute nodes. AMD1003, ¶52 (*citing* AMD1030, 3 (“interface” is “[a] shared electrical boundary between parts of a computer system”); AMD1029, p.1). Indeed, such an understanding is consistent with Dalal's method of using interfaces to connect various elements in its system. AMD1004, [0040] (“switching circuit 220 can include a...wired...network *interface*”), [0060] (“network *interfaces*...part of a hwa module”), [0069] (“[n]etwork *interface*...[is] connected to a host processor...and hwa module...by one or more buses,”), [0094]–[0100]; AMD1003, ¶52.



AMD1004, FIGS. 1, 2B Combination Figure.

Moreover, Dalal teaches that its server switches include hardware, such as a bus connection or bus interface (*host interface*), that couples to the PCIe bus and is compatible with the PCIe protocol. AMD1004, [0032]–[0036]; AMD1003, ¶53 (*citing* AMD1031, 5:53–6:4 (describing a “host PCIe interface” as connecting to a PCIe bus)). This is similar to the ’750 patent which discloses that the router’s *host interface* is “operative in accordance with...PCIe...hardware and associated communications protocols.” AMD1001, 9:38–53, 11:8–21; AMD1003, ¶54.

Further, including a hardware interface such as a port on a switch for connecting to a PCIe link was well known by the time of the '750 patent. AMD1026, 33; AMD1036, 7:19–23 (describing a “PCIE switch device” that

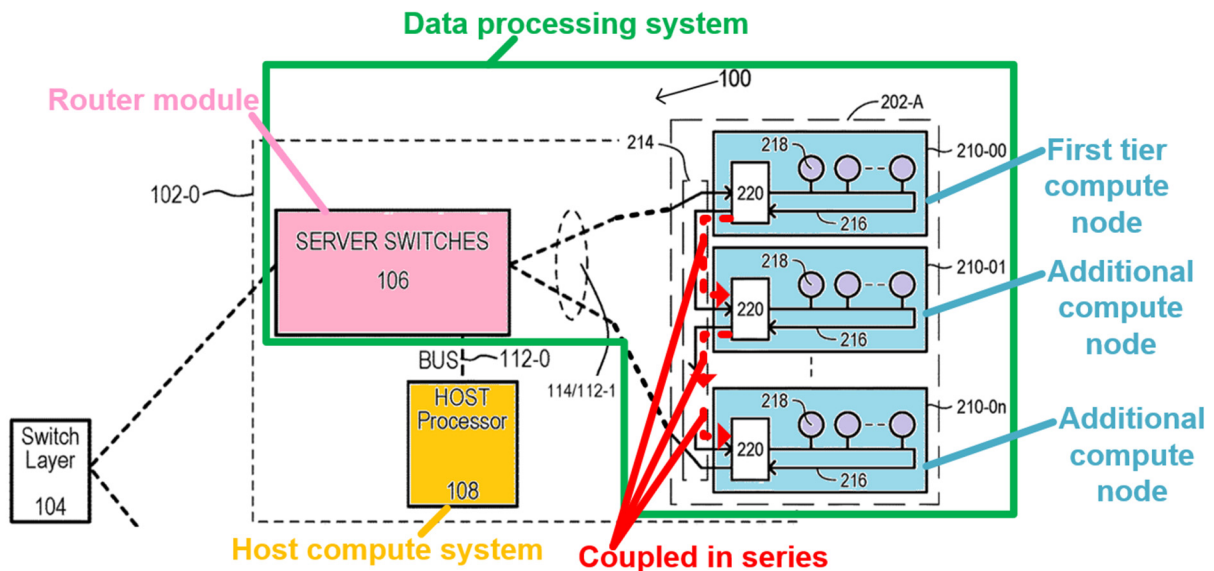
includes a “PCIe port...to connect to the HOST or the PCIe Endpoint”); AMD1003, ¶55.

Similarly, Dalal’s server switches (*router module*) would also have included hardware interfaces (e.g., a bus connection or bus interface) (*node interface*) to be connected to one or more hwa modules via the PCIe bus. AMD1004, [0032]–[0036], [0123], [0170]. Indeed, the ’750 patent acknowledges that including a *node interface* on a *router module* was “generally known in the art.” AMD1001, 42:4–30; AMD1003, ¶56.

Element [1b]

Dalal’s server includes hwa modules (*compute nodes*) that “execute operations on received data.” AMD1004, [0026]–[0028], [0032]–[0037], [0040]–[0046], [0089]–[0092], [0096]–[0098], [0123]–[0125], [0170]–[0172], FIGS. 1, 2B, 8, 12, 18; AMD1003, ¶57. Each hwa module includes one or more “computing elements,” and therefore represents a *compute node* at least because it performs computation operations on received data. AMD1004, [0026], [0033], [0075], [0089]. Dalal also explains that a plurality of hardware acceleration modules within the same server can be interconnected to perform “pipelined processing” that involves “processing by hwa modules *in series* and/or in parallel.” AMD1004, [0042], [0045]–[0049], [0035], [0036], FIGS. 1, 2B–2C; AMD1003, ¶57.

Dalal explains that a server “200-B” includes a first hwa module “210-00” (*first tier compute node*) that initially receives “packet data” before any other hwa module in the server 200-B. AMD1004, [0042], [0045]–[0047], FIG. 2B; AMD1003, ¶58. This packet data is “forwarded to a next hwa module 210-01” (*additional compute node*), where it can be further processed by elements in the next hwa module” and “[s]uch processing can continue through multiple hwa modules (210-00 to -0n) of a same server.” *Id.* Because Dalal explains that the “hwa module 210-00” (*first tier compute node*) processes packet data initially and then passes the packet data to hwa modules 210-01 to -n (*one or more additional compute nodes*) for processing in sequence, the hwa module 210-00 is therefore *coupled to* the hwa modules (210-01 to -0n) *in series*. *Id.* Dalal also explains that the hwa modules can be *coupled in series* to one another via a PCIe bus. AMD1004, [0032], [0042], [0045], [0047], FIGS. 1, 2B–2C; AMD1003, ¶59.

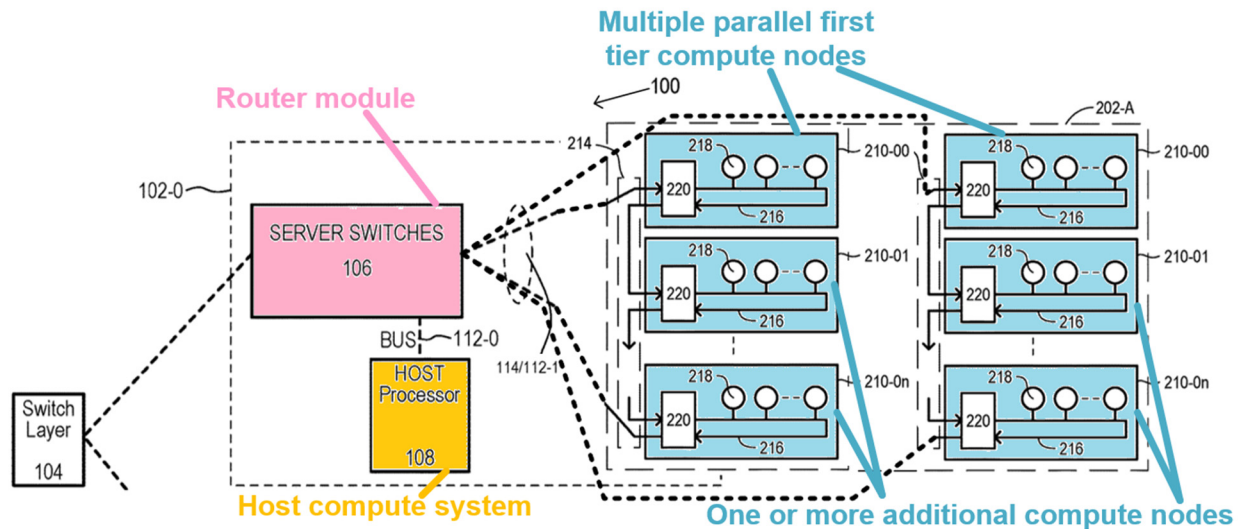


AMD1004, FIGS. 1, 2B.

Dalal explains that one server can have more than one “processing pipeline” each having a *first tier compute node* and one or more *additional compute nodes*. AMD1004, [0040]–[0048], FIGS. 1, 2A–2C; AMD1003, ¶60. Dalal expressly discloses that processing “can be deepened by a serial flow through multiple” hwa modules, and that “scaling up can occur in *parallel*, as well, by sending packet data to” hardware accelerator modules “in *parallel*.” AMD1004, [0045], [0047] (processing by hwa modules *in series* and/or in *parallel*), [0050], FIGS. 1, 2B–2C. Dalal therefore discloses that in the same server, there can be *multiple parallel first tier compute nodes* each *coupled to one or more additional compute nodes* in *series*, as depicted below in a modified version of FIGS. 1 and 2B depicting Dalal’s

embodiment where hwa modules are scaled up both in parallel and in series.

AMD1003, ¶61.



AMD1004, FIGS. 1, 2B Modified Combination Figure.

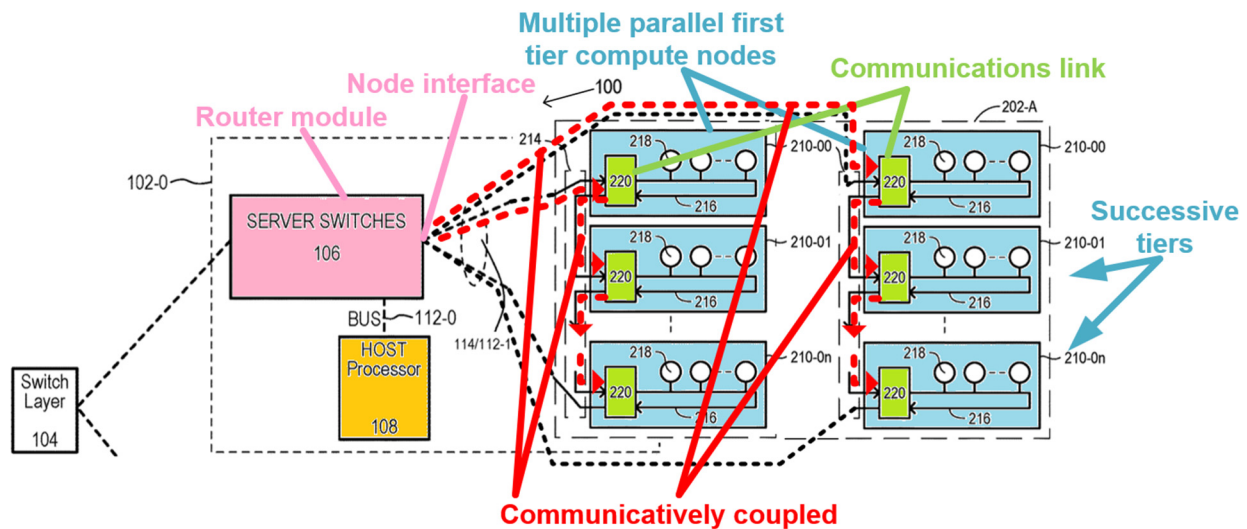
Element [1c]

Each of Dalal’s hwa modules (*compute nodes*) includes a switching circuit 220 (*communications link*) that receives and outputs data. AMD1004, [0040]–[0048], FIGS. 2A–2C; AMD1003, ¶62. The switching circuit is a *communications link* at least because it “can be any suitable structure for translating received packet data.” AMD1004, [0040]. This *communications link* includes the switching circuit and any buses, fabrics, links, and interfaces connected thereto (e.g., data path 214). AMD1004, [0043]. The first hwa module 210-00 (*first tier compute node*) in each *series* of hwa modules is *communicatively coupled* to the *node interface* of the

server switches (*router module*) via the switching circuit (*communications link*).

AMD1004, [0035], [0040], [0042]; *supra* Element_[1b]; AMD1003, ¶63.

In each *series* of *compute nodes* within a server, every hwa module (e.g., 210-01 through 210-0n) that follows the first hwa module 210-00 (*first tier compute node*) represents an *additional compute node* located at a *different successive tier* following the *first tier*. AMD1004, [0032]–[0036], [0042]–[0047], FIGS 1, 2B; AMD1003, ¶64. The *first tier compute node*, and every *additional compute node* located at a *different successive tier* following the *first tier*, includes a switching circuit (*communications link*) that is *communicatively coupled* to at least one other hwa module (*additional compute node*) located at a *different successive tier*. AMD1004, [0032]–[0036], [0042]–[0047], FIGS 1, 2B.



AMD1004, FIGS. 1, 2B Modified Combination Figure.

Element [1d]

Each hwa module includes “computing elements” that execute a ***data processing operation***. AMD1004, [0033]–[0042], [0046], [0075]–[0087], [0026]–[0029], FIGS. 1–2C, 7; *supra* Element [1a]; AMD1003, ¶65. Dalal’s Figure 7 (reproduced below) depicts an example block diagram of a single computing/computation element 718 within a hardware acceleration module. AMD1004, [0075]; *supra* §V.A.3. “A computation element...can include one or more processing circuits 721” that “run a program 736 for a networked application,” thereby confirming that the hwa module executes a ***data processing operation***. AMD1004, [0076].

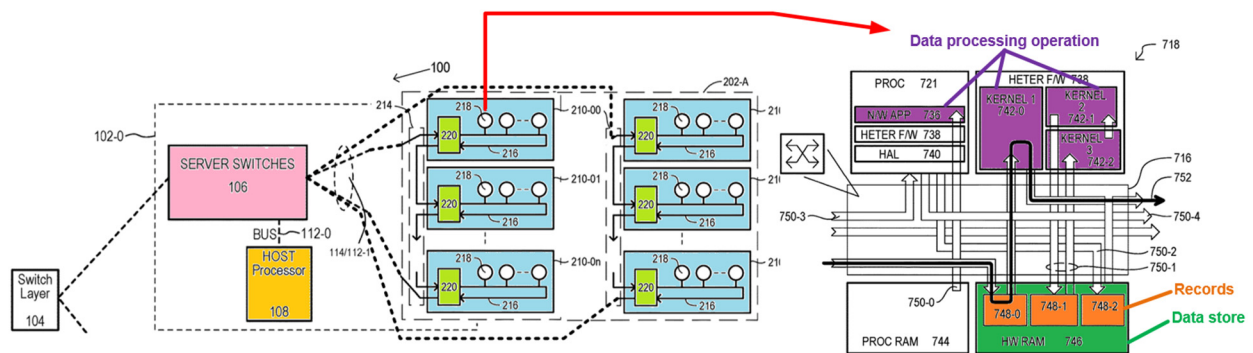


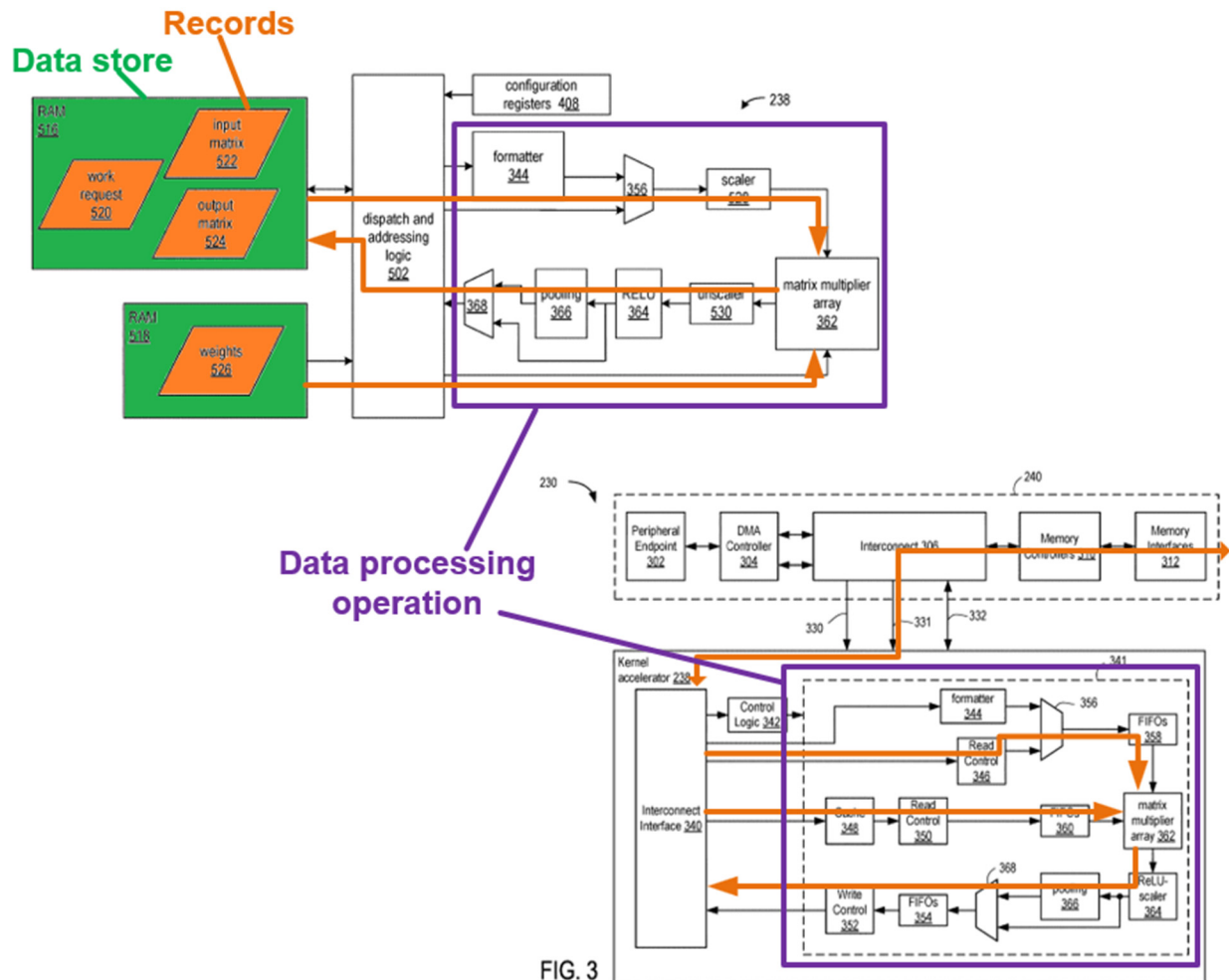
FIG. 7

AMD1004, FIGS. 1, 2B Modified Combination Figure; AMD1004, FIG. 7.

To run the program, Dalal’s processing circuits access ***records*** stored in predetermined memory spaces of a second memory 746 (***data store***) of each computing element. AMD1004, [0081], cls.1–2, [0075], [0080]–[0082], [0085], [0094]. Each memory space provides a functional unit of memory for storing

“packet data” (*record*). *Id.*; AMD1003, ¶¶66–70 (*citing* AMD1030, 4 (a record represents “[a] set of related data items treated as a unit”)). Collectively, the memory spaces and the second memory are a *data store* comprising *records* associated with a *data processing operation*. *Id.*

Moreover, DNC incorporates Ng’s RAM (*data store*) comprising *records* that include matrices and work requests. AMD1005, [0041], [0044], [0007]–[0008], [0024], [0032], [0034], [0036]–[0038], FIGS. 3–5; *see* §V.A.3; AMD1003, ¶71. As depicted below in Ng’s FIGS. 3 and 5, a programmable IC uses these *records* to execute a *data processing operation*, such as performing matrix multiplication for each layer of a neural network using input and weight matrices stored in the RAM. AMD1005, [0041], [0052], FIGS. 3–5.



AMD1005, FIGS. 3, 5.

In DNC, Dalal's hwa modules would have implemented Ng's *data processing operation* through storing *records* (e.g., matrices, work requests) in the memory spaces of Dalal's second memory. AMD1004, [0080]–[0081], FIG. 2B; AMD1005, [0024], [0047], [0094]; AMD1003, ¶72; *supra* §V.A.3. In DNC, Ng's records would have been transmitted as packet data, as in Dalal. AMD1003, ¶¶72–74.

Element [1e]

The computing elements in Dalal's hwa module each include processing circuits 721, 738 that collectively represent a *programmable logic component*. AMD1004, [0075]–[0079], FIG. 7. Example processing circuits “include an ARM type processor” that is “integrated with other circuits” such as “*programmable logic*.” AMD1004, [0077], [0085] (“processing circuits...can be...ARM processors embedded with *programmable [logic]*”); AMD1003, ¶75.

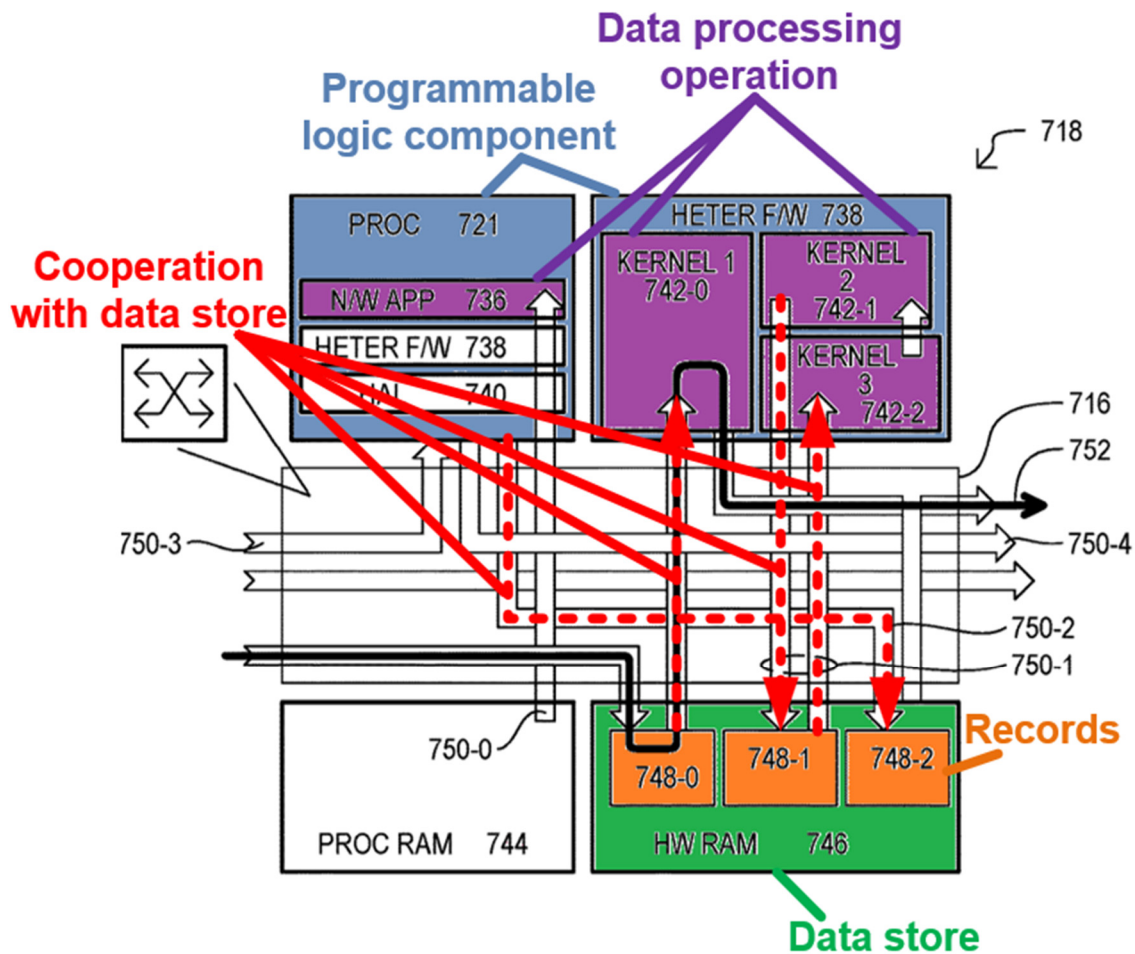


FIG. 7

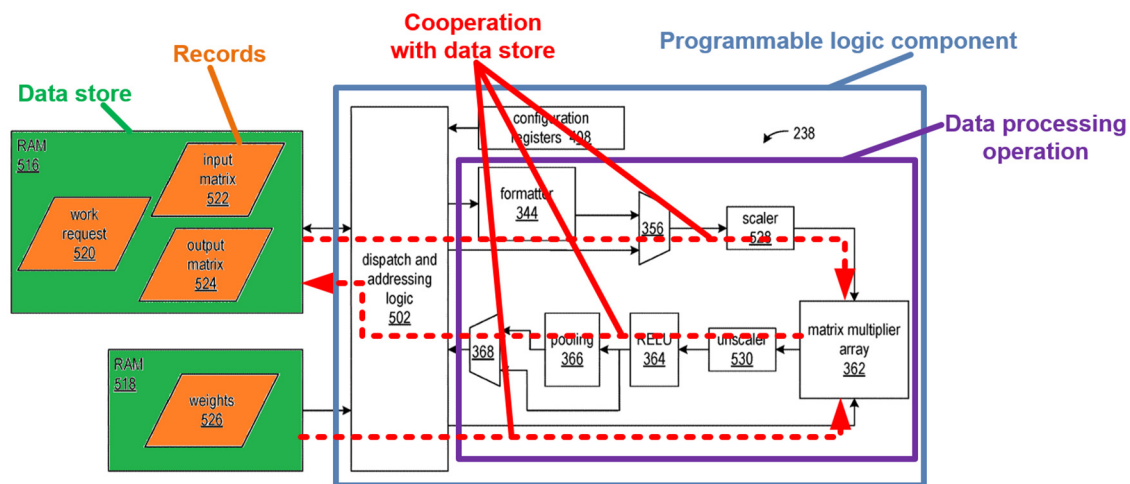
AMD1004, FIG. 7.

Internal hardware connections of Dalal's processing circuits (*programmable logic component*) are configurable (or reconfigurable) to "run a program for a networked application." AMD1004, [0076] ("[p]rocessing circuits...configured to run a program"), [0077]–[0079], FIG. 7; AMD1003, ¶76. This program is "written on a framework" that "enable[s] various execution processes...to perform computation actions." AMD1004, [0078], FIG. 7; AMD1003, ¶77. Execution processes retrieve *records* ("packet data") from the *data store* and therefore *execute* the *data processing operation* in *cooperation* with the *data store*. AMD1004, [0078], [0081]; *supra* Element_[1d].

DNC also renders Element_[1e] obvious via Ng's supplemental teaching of a hardware accelerator including a *programmable logic component*. AMD1005, [0025]–[0039], [0042]–[0044], [0047]–[0051], FIGS. 1–5; AMD1003, ¶78. For example, Ng's hardware accelerator includes "*programmable* integrated circuits (ICs)" that include "field *programmable* gate arrays (FPGAs)." AMD1005, [0026], [0032]–[0035], FIGS. 2–3.

Ng explains that internal hardware connections of this *programmable* IC are configured (or reconfigured) to perform a specialized task, which would have motivated a POSITA to implement Dalal's hwa modules with Ng's *programmable logic component*. AMD1005, [0027] ("programming the hardware accelerator" using "configuration bitstreams for FPGAs"), [0034], [0035], [0080]

(“***programmable logic***...includes ***logic*** cells...configured to implement general logic functions”); *supra* Element_[1d]; *supra* §V.A.3; AMD1003, ¶79. As depicted below in Ng’s Figure 5, this programmable IC performs a ***data processing operation*** by retrieving ***records*** (e.g., “input” and “weight” matrices) from a ***data store*** for processing and delivering processed ***records*** (e.g., “output” matrice) to the ***data store***. AMD1005, [0041], [0050], [0053], FIGS. 3–5.



AMD1005, FIG. 5.

A POSITA would have understood that a *programmable logic component*, such as Dalal-Ng’s FPGA, would have included internal hardware connections that are reprogrammable by configuration data (*e.g.*, bitstreams). AMD1003, ¶80 (*citing* AMD1027, 31, AMD1006, [0023]–[0030]).

Element [1f]

As described above in Elements_[1d]–[1e], DNC’s *programmable logic component* runs programs to perform a *data processing operation*. AMD1004,

[0075]–[0082], FIG. 7; AMD1005, [0025]–[0051]; AMD1003, ¶81. These programs are “loaded into first *memory*” 744 of each computing element in an hwa module, and the first memories collectively represent a *node memory*. AMD1004, [0079], [0075], [0082], FIG. 7. As depicted below in Dalal’s Figure 7, the programs are executed by processing circuits to perform execution processes and computation actions and therefore support *operation of the programmable logic component in connection with the data processing operation*. AMD1004, [0076]–[0079], FIG. 7; *supra* Element_[1d].

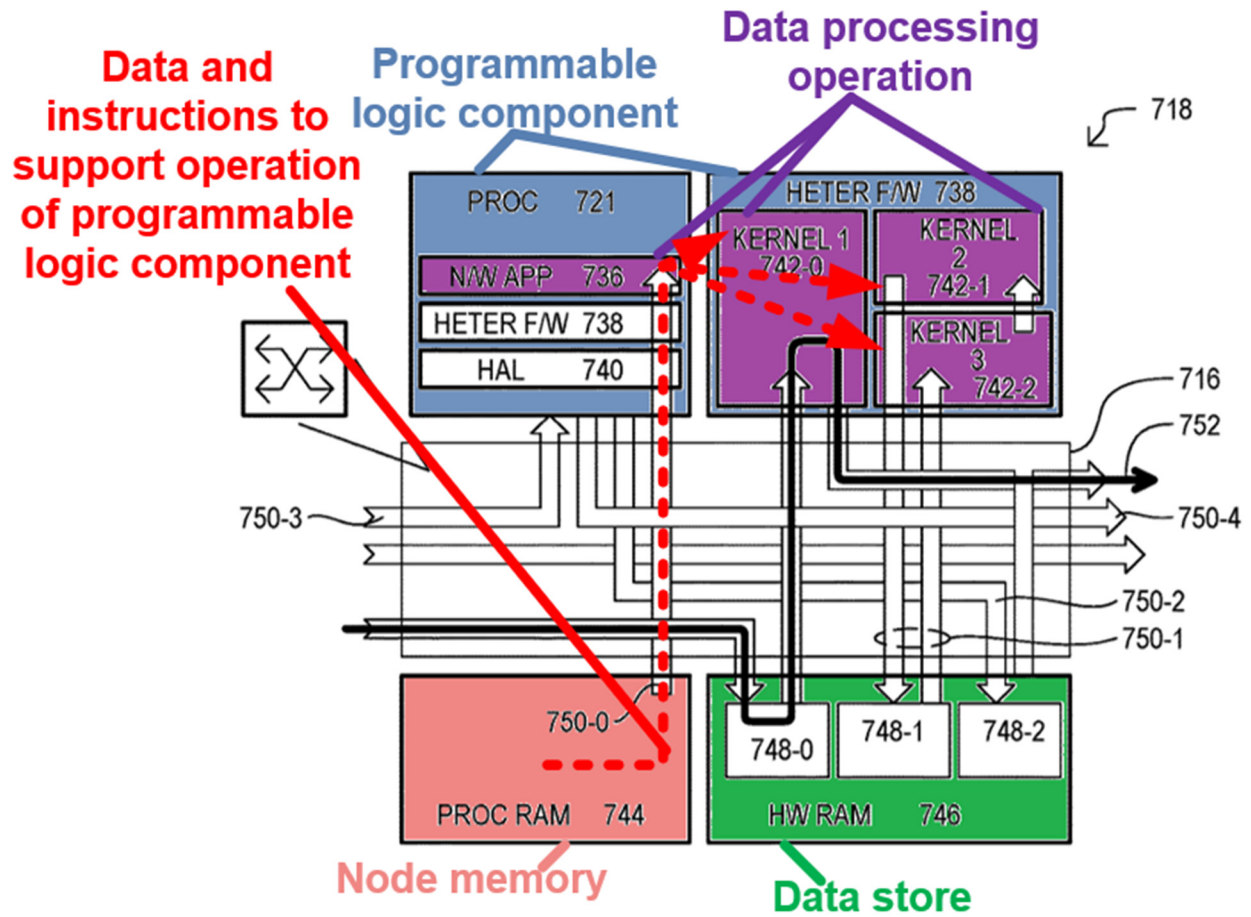


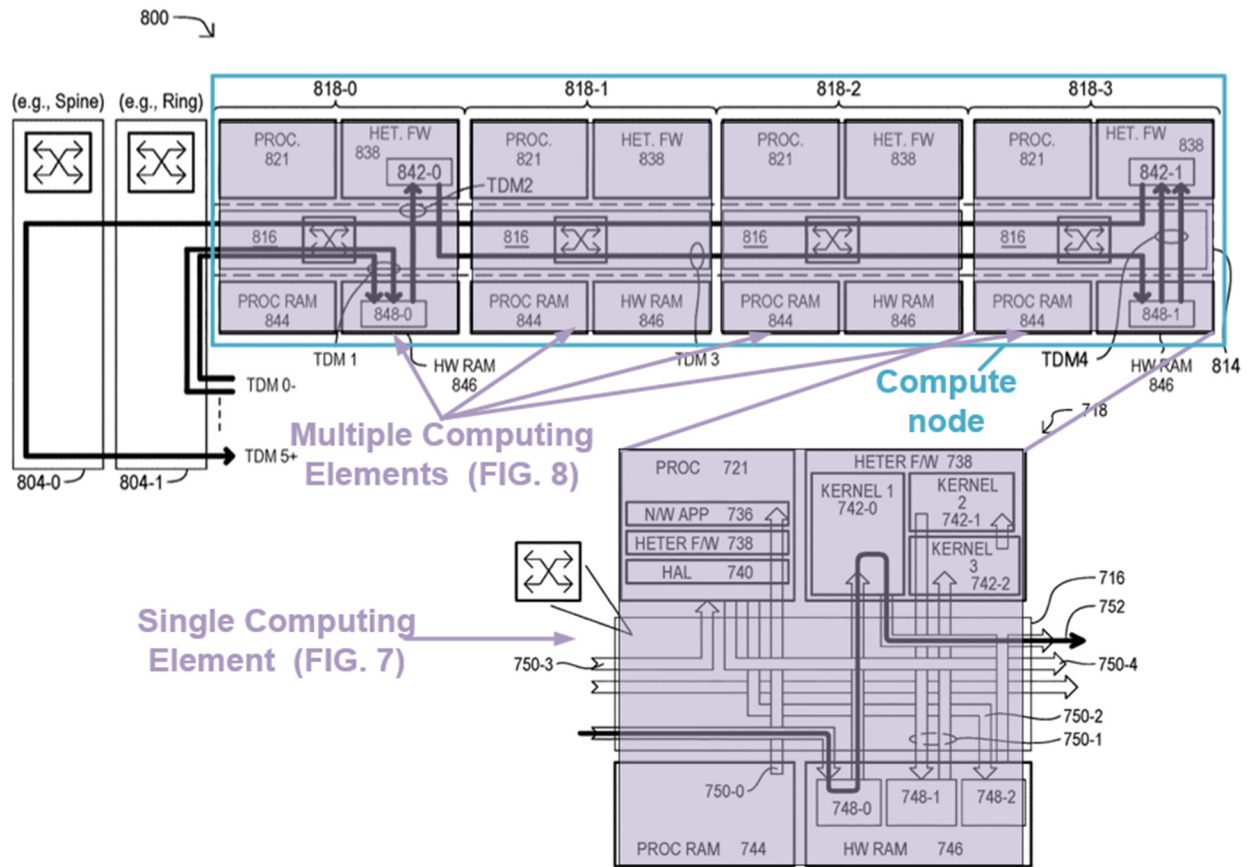
FIG. 7

AMD1004, FIG. 7.

Dalal's first memory 744 (*node memory*) "store[s] *instructions* for execution by the processing circuits" (*supporting operation* of the *programmable logic component*). AMD1004, cl.11. Furthermore, a "*data* transfer fabric 716" provides "*data* transfer paths between components" including connections between the processing circuits and first memory. AMD1004, [0081]–[0083], FIG. 7. Path 750-0 in the fabric transfers *data* and *instructions* from the first memory (*node memory*) to the processing circuits (*programmable logic component*). AMD1004, [0079];

AMD1003, ¶82. Recall also that “[f]irst memory...[is] accessed by processing circuits,” “programs can be loaded into first memory...for execution by processing circuits,” and a “framework can enable various execution processes...to perform computation actions called by a program.” AMD1004, [0078], [0079]. At least because the data and instructions stored in the first memory are used by the processing circuits (*programmable logic component*) to execute operations, DNC’s system renders obvious *a node memory comprising data and instructions to support operation of the programmable logic component in connection with the data processing operation.* AMD1003, ¶83.

The analysis will now turn to the *inter-node communications* aspect of Element_[1f]. As described above, Dalal’s Figure 7 depicts a single computing element located within an hwa module that includes one or more computing elements. AMD1004, [0075], [0033]; *supra* Element_[1b], Element_[1d], FIG. 7; AMD1003, ¶84. Figure 8 then depicts an hwa module having multiple computing elements 818-0 to -3 that each “take the form” of the single computing element in Figure 7. AMD1004, [0089], [0090], FIGS. 7–8.



AMD1004, FIGS. 7–8.

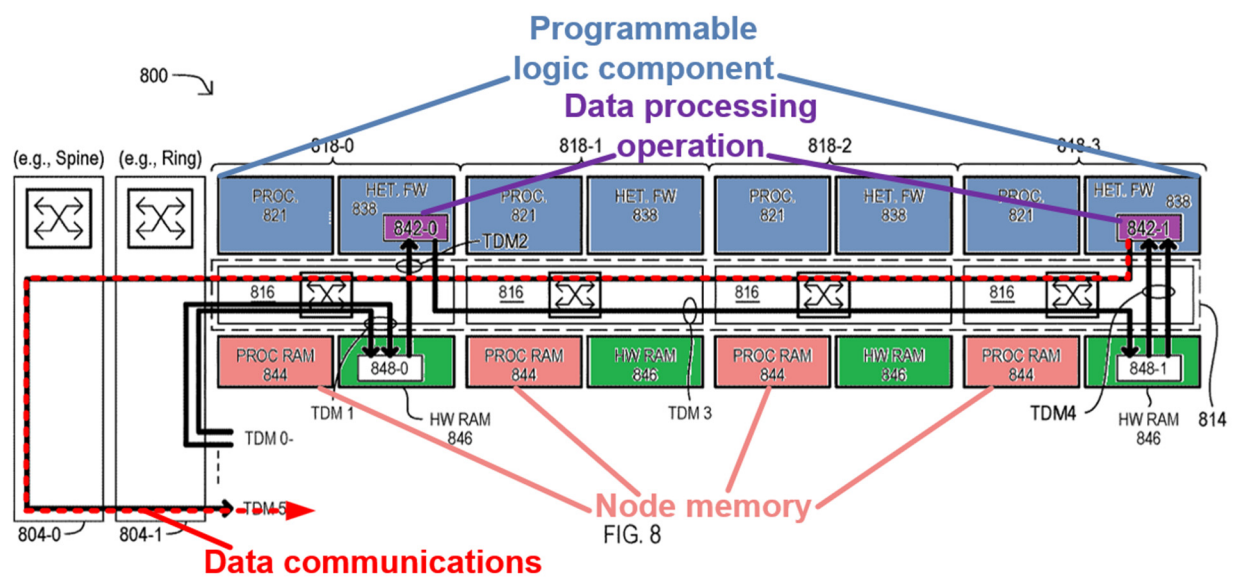
Figure 8 depicts the understanding a POSITA would have had of a computing element 718, as shown in Figure 7, in connection with other similar computing elements, which is consistent with the connection of computing elements shown in Dalal’s Figure 2B. *See* §V.A.4, Element_[1a]; AMD1003, ¶85. In particular, Dalal explains that the computing elements 818-0 to -3 of the hardware acceleration module are connected to other parts of a system over a first connection layer 804-0 that couples hwa modules of the same server. AMD1004, [0091]. The first connection layer 804-0 “include[s] a bus...that can connect hwa modules to one

another” and “can include inter-module connections...including direct inter-module connections.” AMD1004, [0092]. This supports other parts of Dalal’s disclosure (e.g., Figure 2B) confirming that *data communications* occur between *compute nodes* in the same server. AMD1004, [0042], FIG. 2B; *supra* Elements_[1b]–[1c]; AMD1003, ¶85. These *data communications* occur via the switching circuit (*communications link*) in each hwa module (*compute node*). AMD1004, [0040]–[0045], FIGS. 1, 2B; *supra* Element_[1c].

According to Dalal, a result of execution process 842-1 is sent via switching layer 804-0 to another location (e.g., another appliance with computing elements connected by an inter-module fabric). Execution process 842-1 “operate[s] according to a same network application” corresponding to the “programs” of Dalal’s Figure 7 which are run by the *programmable logic component* and stored in the *node memory*. AMD1004, [0095], [0076]–[0088], FIGS. 7–8. A POSITA would have understood that execution process 842-1, like the execution processes 742-0 to -2 of Figure 7, “perform computation actions called by...[a] program” which is “performed by processing circuits.” AMD1004, [0078]; AMD1003, ¶86. Just as programs are “loaded into” the first memory 744 “for execution by processing circuits 721,” a POSITA would have understood that programs are loaded into the first memory (PROC RAM 844) of Figure 8 for execution by processing circuits (PROC 821). AMD1004, [0079]; AMD1003, ¶87. These

programs define a network application that governs both what data is generated in the data processing operation performed by the hwa module (*compute node*) and that such data must be forwarded to the next *compute node* in the *execution pipeline*. AMD1004, [0076]–[0078], [0095]. Because Dalal explains that the first connection layer 804-0 includes “direct inter-module connections” to hwa modules “of a same server,” a POSITA would have understood that the *data* and *instructions* stored in Dalal’s *node memory* facilitate *data communications* by causing execution process 842-1 to forward a result to another hwa module (*compute node*) in the same server.

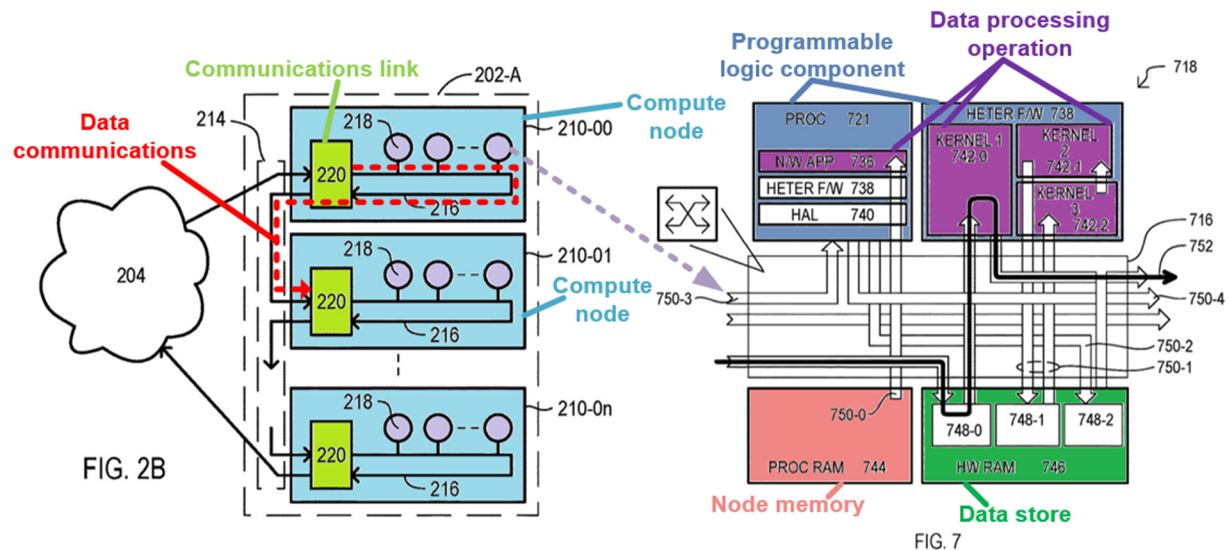
Id.



AMD1004, FIG. 8.

Furthermore, because running the *data and instructions* stored in Dalal’s first memory (*node memory*) cause, upon processing, packet data to flow to a next hwa

module, a POSITA would have understood that the data and instructions *facilitate data communications* via the switching circuit (*communications link*). AMD1004, [0035]–[0036], [0042], [0045], [0075]–[0078], FIGS. 1, 2B–2C, 7; AMD1003, ¶88.



AMD1004, FIGS. 2B, 7.

Element [1g]

In Dalal, a data transfer fabric 716 provides data transfer paths. AMD1004, [0082], [0033]–[0036], [0040]–[0043], [0075], [0079], [0081]–[0089], FIGS. 1–2C, 7–8; *supra* Element [1f]. Dalal’s processing circuits (*programmable logic component*) access the first memory (*node memory*) through the data transfer fabric via the path 750-0 (*data mover component*) such that path 750-0 facilitates *data communications* between these components. AMD1004, [0079], [0034], [0075]–[0088], cl.1, FIG. 7; AMD1003, ¶89.

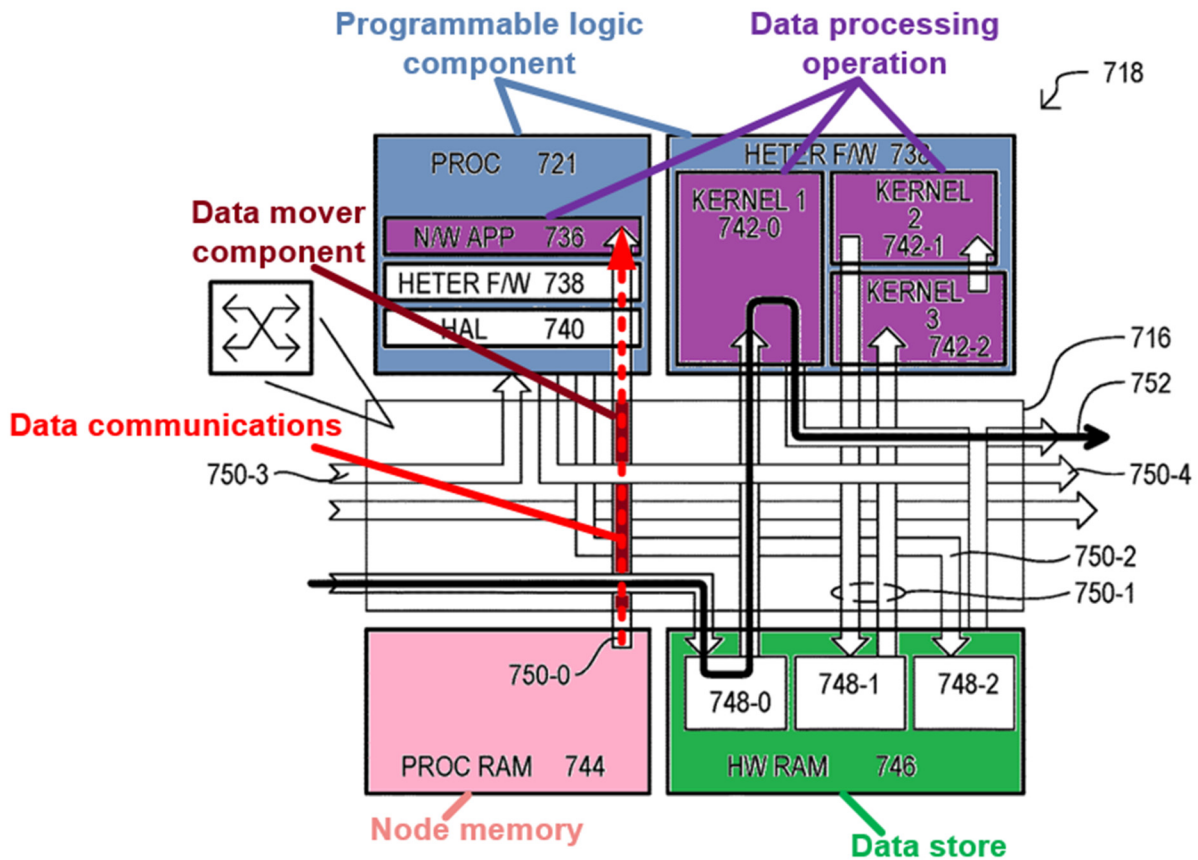


FIG. 7

AMD1004, FIG. 7.

Dalal explains that the data transfer fabric is “programmable” and “formed by buses on a programmable logic device, such as an FPGA,” much like the *data mover component* of the ’750 patent. AMD1004, [0084]–[0087], FIG. 7; AMD1001, 15:3–30, 15:58–16:7; AMD1003, ¶90 (*citing* AMD1025, p.32).

Element [1h]

DNC renders Element [1h] obvious (1) based on Dalal’s teachings alone, and (2) based on a combination of Dalal-Ng. AMD1003, ¶91.

First, Dalal's second memory 746 (*data store*) is accessed by various computing circuits including the processing circuits, 721, 738 (*programmable logic component*) indicating that *data communications* occur between the *programmable logic component* and the *data store* of the same *compute node*. AMD1004, [0078]–[0082], FIG. 7; AMD1003, ¶92. To *facilitate* these *data communications*, the computing element includes data transfer paths (*storage interface component*) through Dalal's data transfer fabric 716 that connect the *programmable logic component* to the *data store*. AMD1004, [0082], [0081] (“memory spaces...serve to provide connections between...components”), [0078]–[0080], FIG. 7; AMD1003, ¶93.

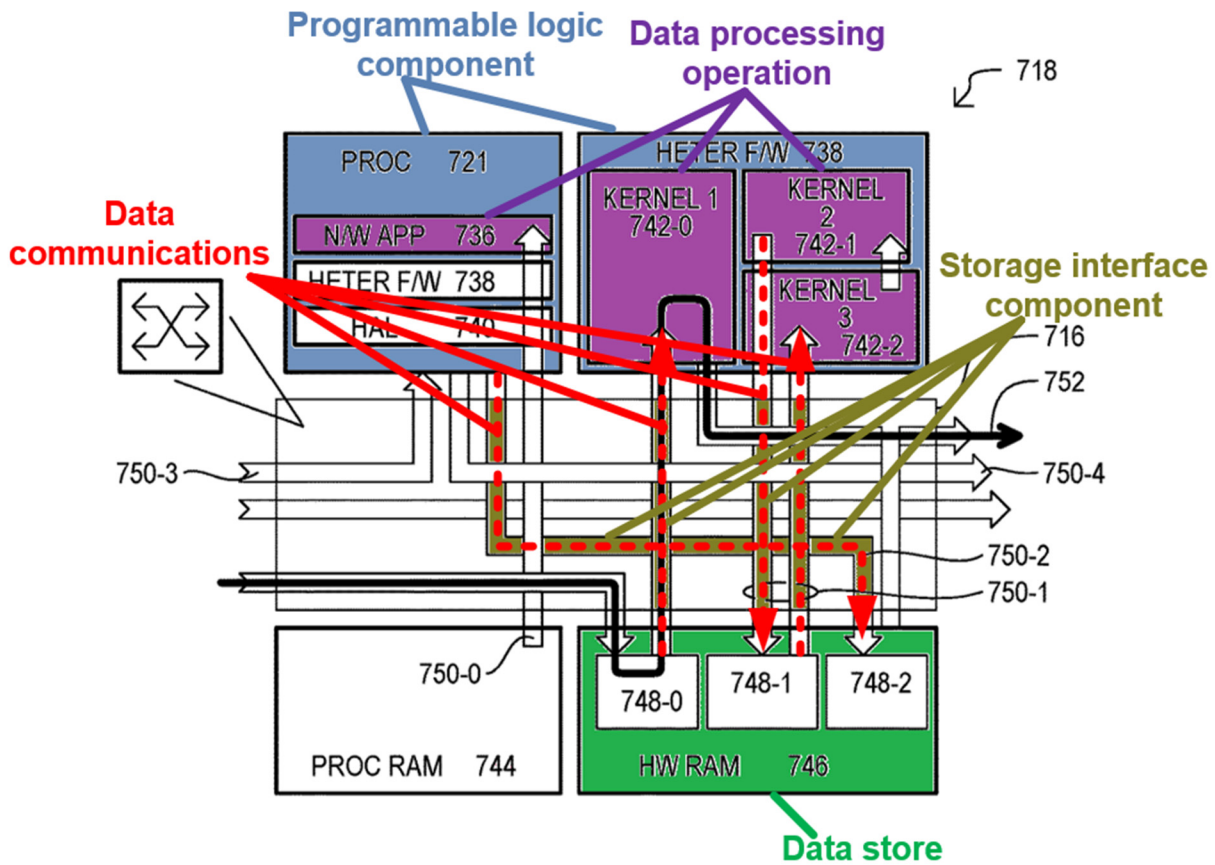
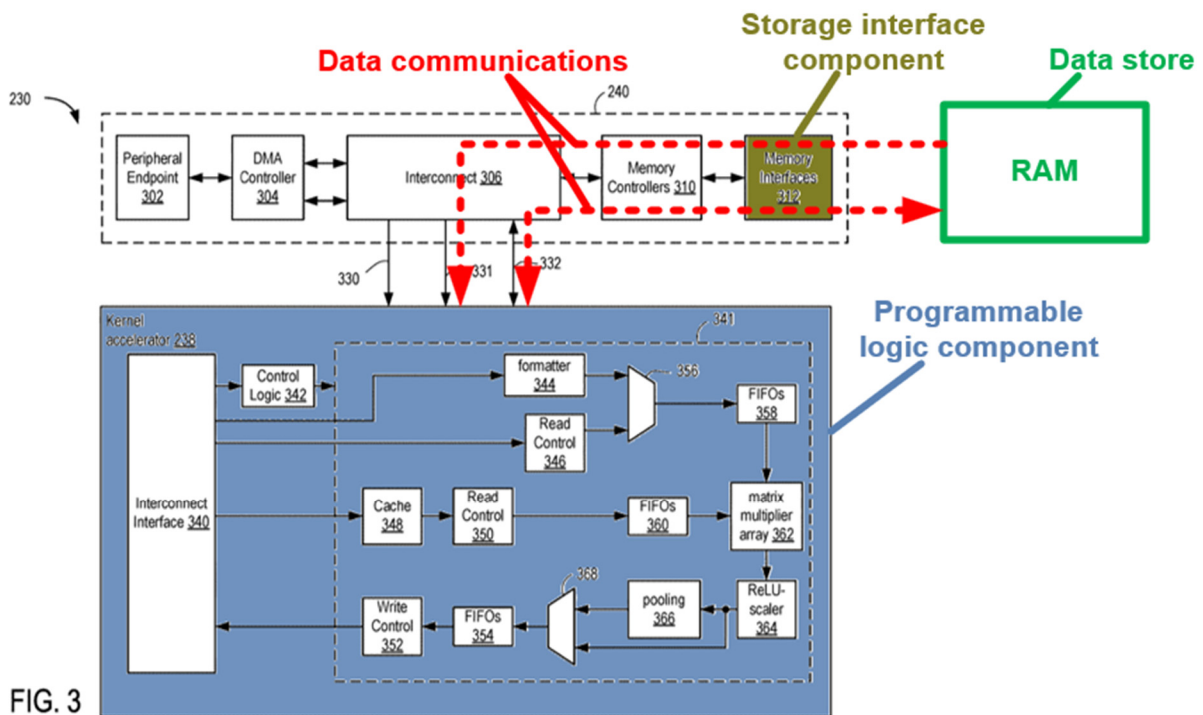


FIG. 7

AMD1004, FIG. 7.

Second, DNC also incorporates Ng's memory *interfaces*. AMD1005, [0035]–[0038], FIG. 3; AMD1003, ¶94. Ng's memory interfaces are coupled to a RAM and allow the neural network accelerator to read from the RAM and write to the RAM. AMD1005, [0036], [0038], [0040]–[0041], FIG. 3.



AMD1005, FIG. 3.

In DNC, a POSITA would have found it obvious to implement Dalal’s computing elements with a dedicated storage interface component to facilitate *data communications* over Dalal’s data transfer fabric because doing so would have provided dedicated circuitry configured with one or more memory protocols of the data store (*e.g.*, DDR RAM). AMD1004, [0080] (“second memory...accessible by programmable logic (FPGA), such as static RAM (SRAM)”), [0085]; AMD1005, [0032], [0035], [0043]; AMD1003, ¶95; *supra* §V.A.3. As depicted below in the modified version of Dalal’s Figure 7, the computing element in DNC would have included memory interfaces connected to the second memory (HW RAM) (*data*

store), just as Ng discloses that “[t]he memory interfaces...are coupled to the RAM.”

AMD1004, [0036].

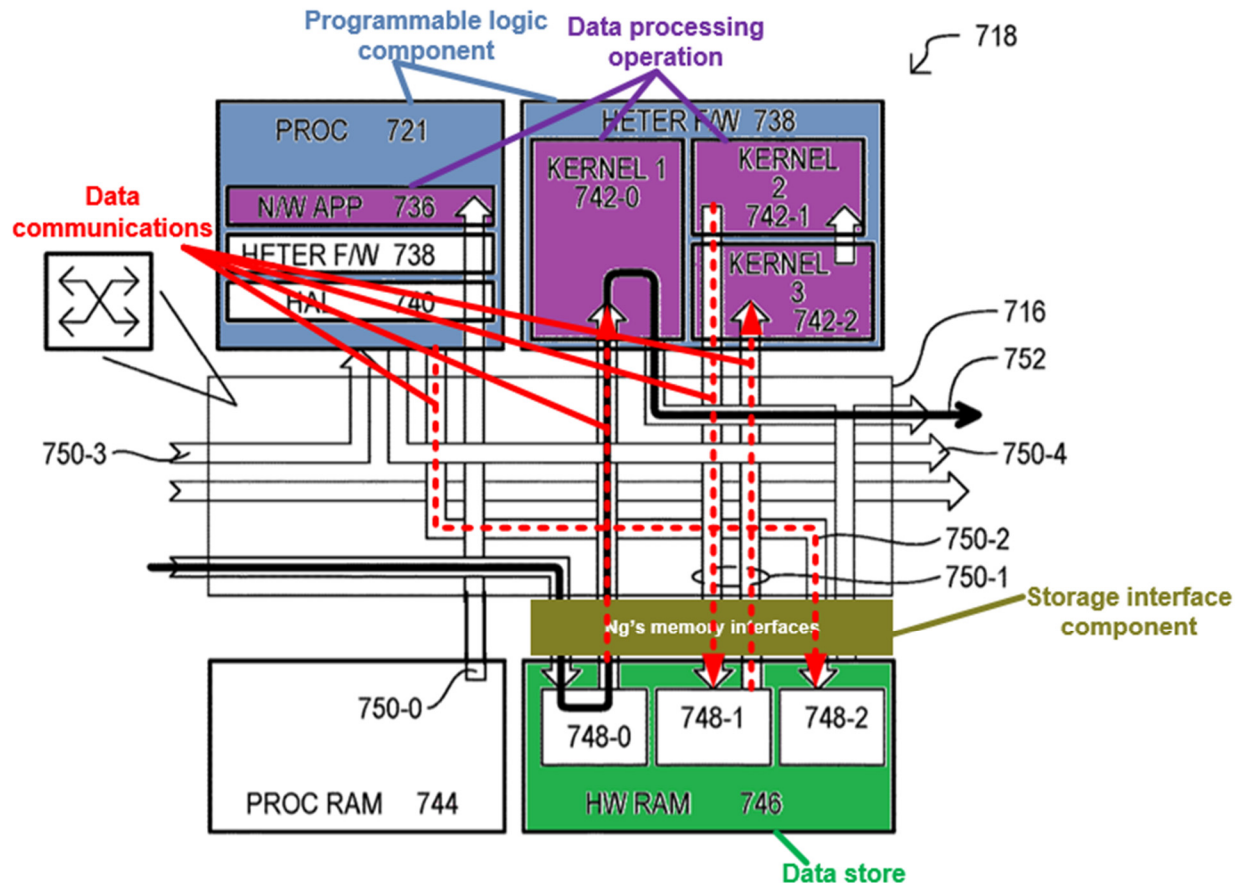


FIG. 7

AMD1004, FIG. 7 (modified).

Ng's memory interfaces are similar to the *storage interface component* of the '750 patent. AMD1001, 24:59–25:17 (describing storage interface component including a memory interface), 15:31–57; AMD1003, ¶96. Furthermore, the '750 patent acknowledges that a data store interface that manages access to a data store is “generally known in the art.” AMD1001, 10:32–34, 15:50–54; AMD1003, ¶97.

Element_[1i]

DNC renders Element_[1i] obvious (1) based on Dalal's teachings alone, and (2) based on the combination of Dalal-Ng. AMD1003, ¶98.

First, Dalal's data transfer fabric provides different data transfer paths that include "connections between...processing circuits 721 (***programmable logic component***)...[and] second memory 746 (***data store***).” AMD1004, [0082], [0081] (“various possible connections”), [0084] (“providing connections as needed”), FIG. 7; *supra* Elements_[1g]–[1h]; AMD1003, ¶99. As depicted below in Dalal's Figure 7, these connections include at least four (***a plurality***) of data transfer paths (***communications channels***) that ***transfer data*** between the processing circuits (***programmable logic component***) and the second memory (***data store***). *Id.*

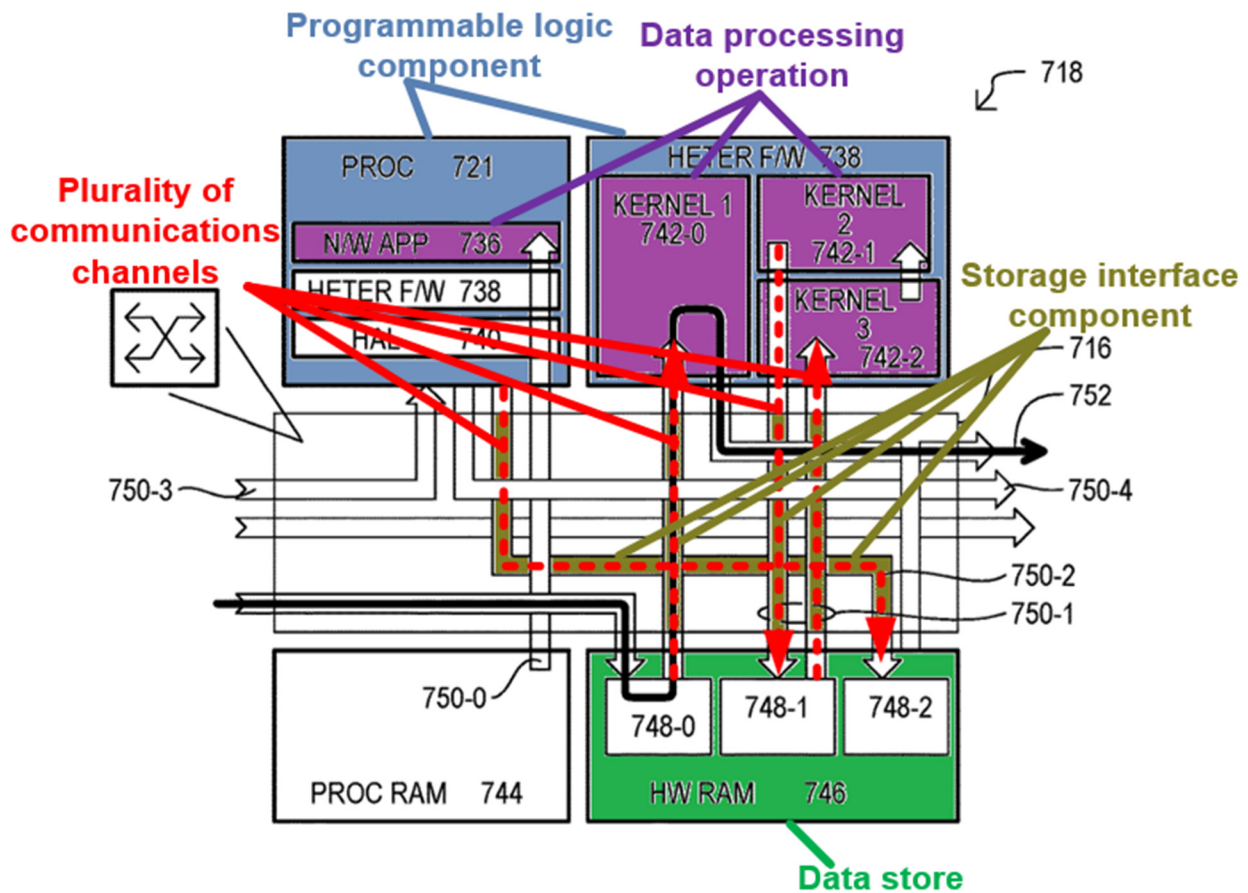
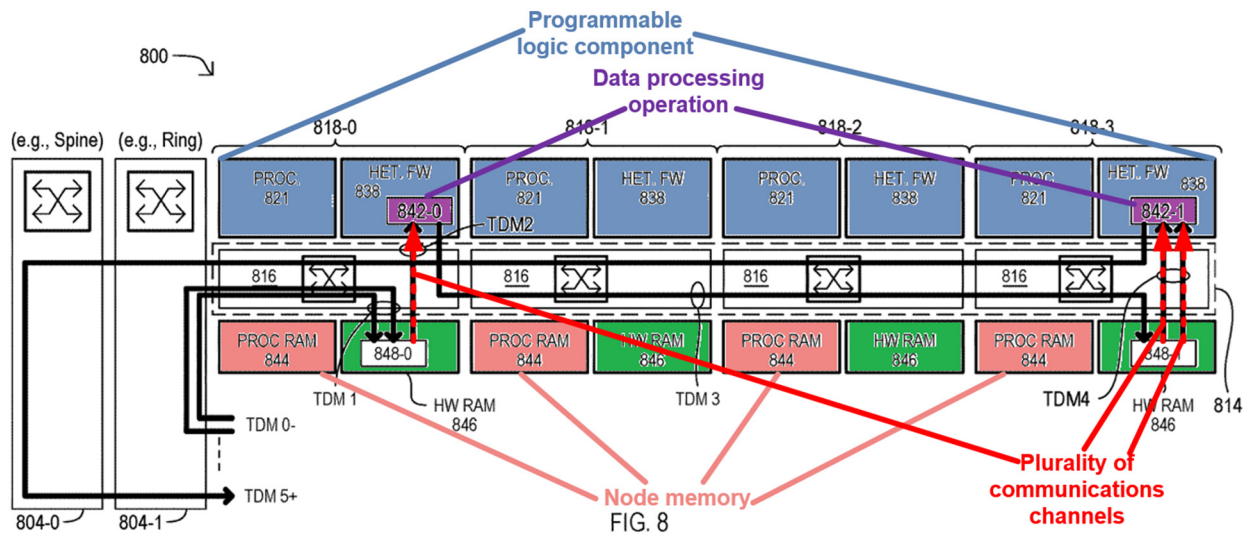


FIG. 7

AMD1004, FIG. 7 (modified).

Furthermore, Dalal's Figure 8 depicts a *plurality of communications channels* that transfer data between memory spaces (*data store*) and execution processes (*programmable logic component*) within an hwa module. AMD1004, [0094] ("data in memory space 848-0...read out by execution process...842-0...execution process 842-1...read data from memory space 848-1"), [0095], FIG. 8; AMD1003, ¶100.



AMD1004, FIG. 8.

Second, as explained above, DNC also incorporates Ng's express teaching of memory interfaces (*storage interface component*) including a first memory interface used to read from the RAM and a second memory interface used to read from and write to the RAM (*plurality of communications channels*). AMD1005, [0035]–[0038], FIG. 3; AMD1003, ¶102; *supra* §V.A.3. The modified version of Dalal's Figure 7, below, demonstrates that Dalal-Ng's memory interfaces would have utilized a *plurality of communications channels* that *transfer data*.

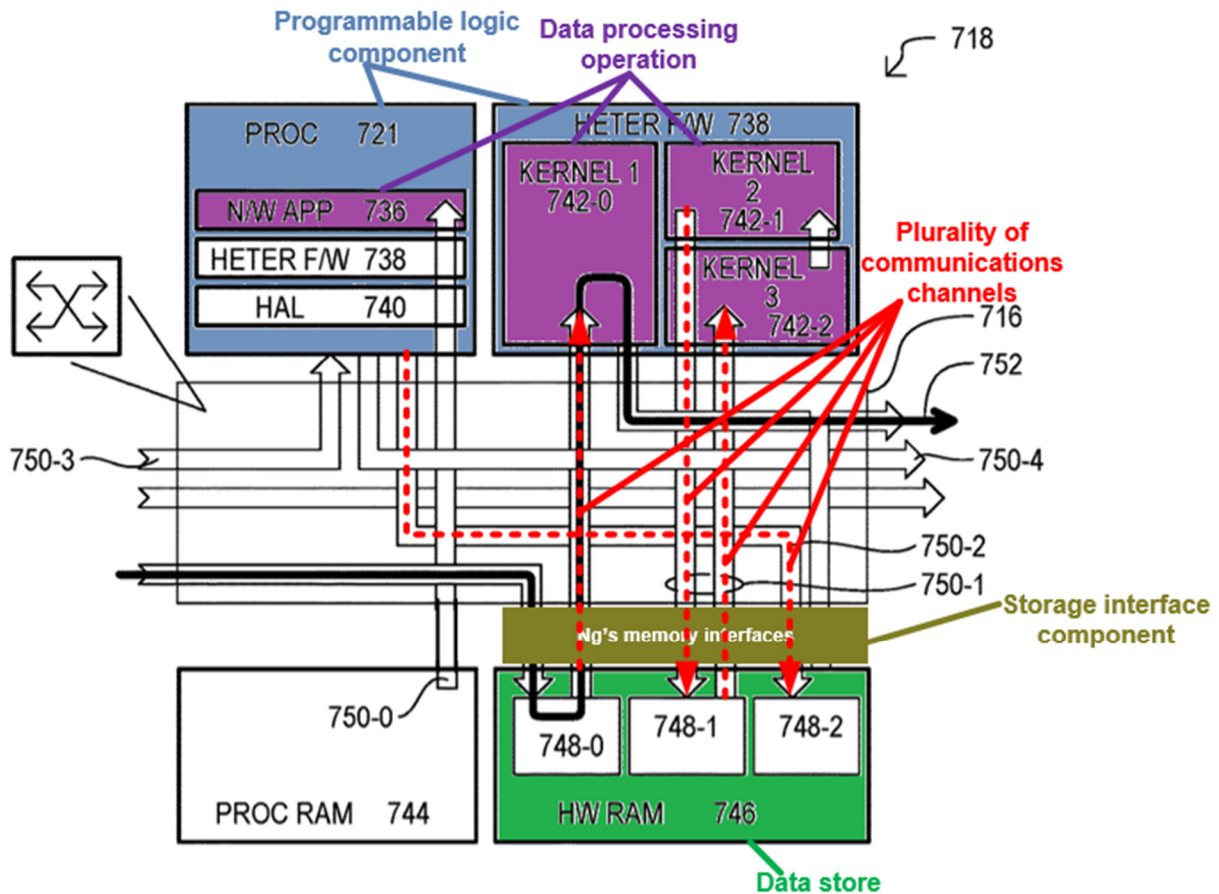


FIG. 7

AMD1004, FIG. 7 (modified).

Finally, a POSITA would have understood that it was well known for a *storage interface component* to include a *plurality of communications channels* in a hardware accelerator. AMD1028, 8:4–24; AMD1043, [0039]; AMD1003, ¶103.

Element [2]

Dalal explains that within a server, PCIe buses, serving as a fabric, connect hwa modules to each other and to other components. AMD1004, [0032]–[0036], [0043], [0064], [0089]–[0099], [0123], [0170], FIGS. 1, 2B, 8, 12, 18; *supra*

Elements_[1b]–[1c]. Because the fabric is connected to the switching circuit (*communications link*) using the PCIe standard—a *serial communications protocol*—it would have been obvious to a POSITA that the “switching circuit” (*communications link*) of each hwa module employs a *serial communications protocol*. AMD1004, [0032], [0040]–[0048], FIGS. 1, 2B; *supra* Element_[1c]; AMD1003, ¶104. Multiple references confirm that PCIe is a *serial communications protocol*. AMD1018, p.1; AMD1023, p.1; AMD1026, p.27; AMD1008, 3:30–55; AMD1003, ¶105. Ng’s disclosure further reinforces the obviousness of using a *serial communications protocol* such as PCIe. AMD1005, [0030]–[0043], FIGS. 2–4; AMD1003, ¶106; *supra* §V.A.3.

Element_[5]

Dalal’s hwa modules have a series of computing elements, each including processing circuits that comprise a *programmable logic component*. AMD1004, [0033], [0076], [0077], FIG. 7; *supra* Element_[1b], Element_[1e]. Ng describes a hardware accelerator that includes “*programmable* integrated circuits (ICs), such as *field programmable gate arrays (FPGAs)*,” and a POSITA would have implemented Dalal’s processing circuits (*programmable logic component*) with *FPGAs* as suggested by Ng because doing so would beneficially enable a developer to customize the programmable logic component for specialized operations. AMD1005, [0026], [0032], [0058], [0081]–[0086], FIGS. 2–3, 9; AMD1003, ¶107;

supra §V.A.3. This is consistent with the fact that Dalal’s processing circuits include logic “derived from...**programmable** circuits,” and that the processing circuits are part of a computing element representing an “integrated circuit device, such as a computational **FPGA**.” AMD1004, [0077], [0087], [0033], [0065], [0080], [0096], [0104]; AMD1003, ¶108.

Element_[6]

Dalal’s processing circuits within each hwa module (***compute node***) include “ARM **processors**...to control switch fabric 716,” including a **processor** that **manages operation** of the path 750-0 (***data mover component***), by “access[ing] first memory 744 through” the “path shown as 750-0.” AMD1004, [0079], [0085], [0029], cl.7; AMD1003, ¶109; *supra* Element_[1b], Element_[1g].

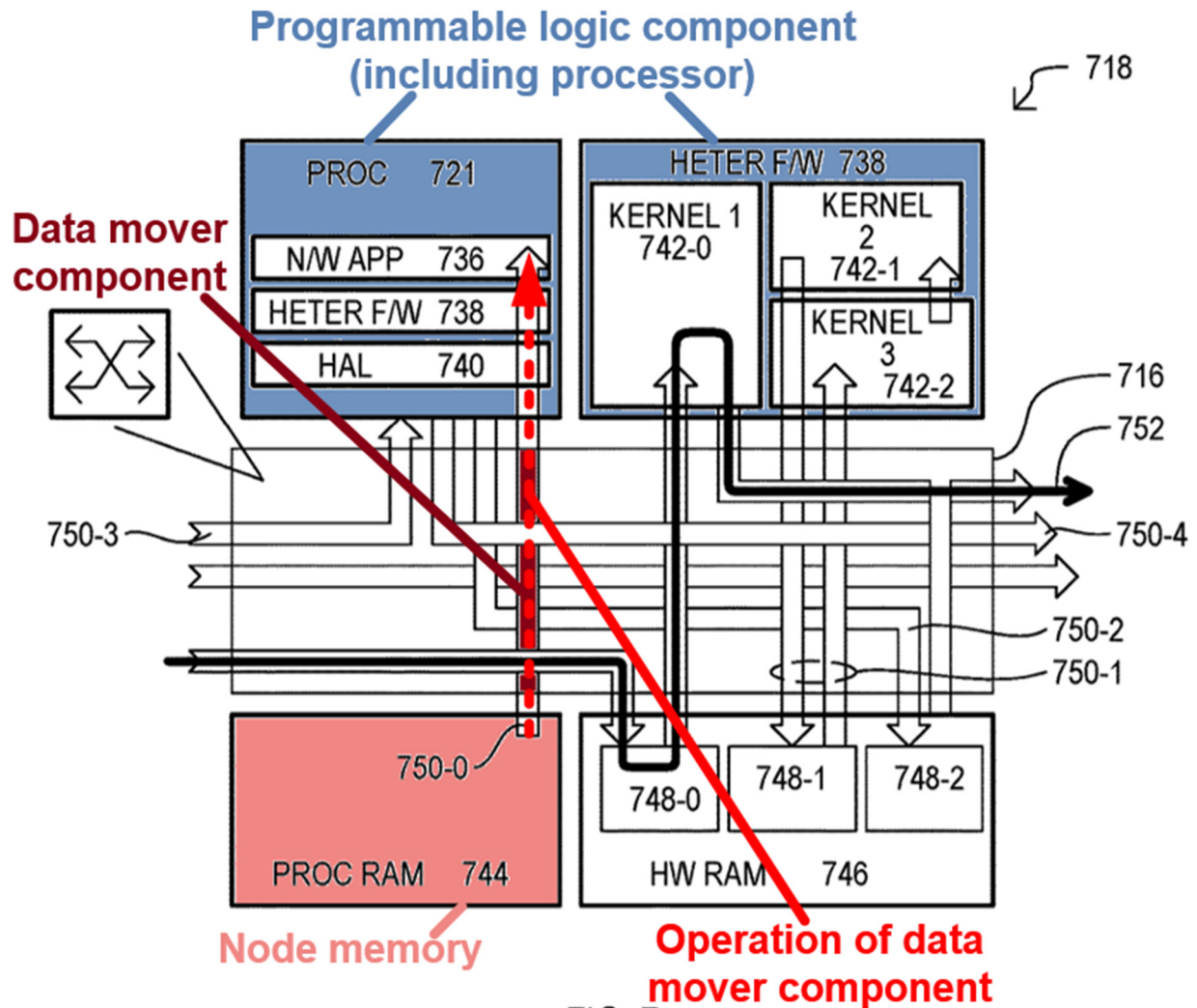


FIG. 7

AMD1004, FIG. 7.

DNC also incorporates Ng's supplemental teaching of a read control circuit and a write control circuit (*processor*) within a *programmable logic component* that facilitate communication between a *programmable logic component* and a memory via a *data mover component*. AMD1005, [0035]–[0038], [0040]–[0041], [0080], FIGS. 2–3; *supra* §V.A.3; AMD1003, ¶110. A POSITA would have integrated these

circuits into Dalal's ARM processors to provide dedicated logic that is customized to manage data mover operations required for a data processing operation. *Id.*

Element [7]

The portion of Dalal's "data transfer fabric" (***data mover component***) in each hwa module (***compute node***) connecting the processing circuits (***programmable logic component***) to the first memory (***node memory***) represents a ***device controller***. AMD1004, [0081]–[0086]; *supra* Elements_[1b], [1g], [6]; AMD1003, ¶111. This data transfer fabric can be controlled by an ARM ***processor*** of the ***programmable logic component***, similar to the '750 patent's ***device controller*** which is operative to facilitate data communications either independently or in cooperation with control signals. AMD1004, [0085]; AMD1001, 15:58–16:7; AMD1003, ¶112. Incorporating Ng's supplemental teaching of memory ***controllers*** within the hardware accelerator ***compute node*** which provide the claimed ***device controller*** would have been obvious to a POSITA because this would have beneficially provided a way to manage processor-memory communications in every module. AMD1005, [0035]–[0037], [0043], FIG. 3; AMD1003, ¶113; *supra* §V.A.3.

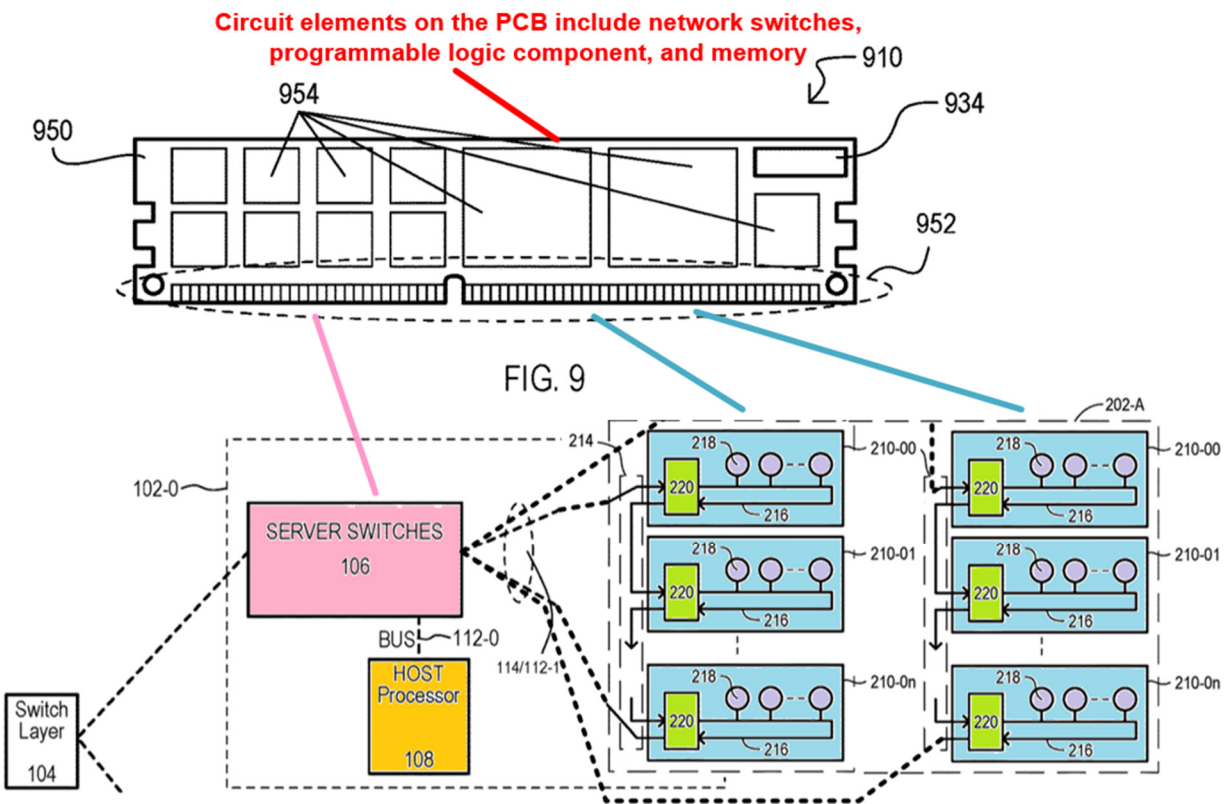
Element [8]

As described above, Dalal's host processor (***host computer system***) is connected to hwa modules (***plurality of compute nodes***). AMD1004, [0032]; *supra* Elements_[1pre]–[1b]; AMD1003, ¶114. DNC implements the host processor with

Ng’s “host computer system” (*host computer system*) that executes *instructions* (e.g., input and weight matrices, and a work request) for a “neural network accelerator to perform neural network operations of each layer of the neural network.” AMD1005, cl. 11, [0007]–[0008], [0021]–[0024], [0029], [0041]–[0050], FIGS. 4–5; AMD1004, [0032], [0070] (“[a] host processor...can be any suitable processor device”), [0063], [0069], [0073], [0077], [0099]–[0103]; *supra* §V.A.3; AMD1003, ¶¶114–115. In DNC, the *instructions* would influence *operation* of each hwa module (*compute node*) to efficiently execute a part of Ng’s operation such as a neural network layer. *Id.*

Element [9]

Dalal’s server switches (*router module*) are “connected to host processor 108” and include “one or more network switches.” AMD1004, [0032], [0123], [0170], FIG. 1; *supra*, Elements_[1a]–[1b]; AMD1003, ¶116. Dalal’s network switch is *integrated into* the same printed circuit board that also includes an hwa module (*compute node*). AMD1004, [0096], FIG. 9. Thus, a POSITA would have understood that Dalal’s printed circuit board is an *interface card* communicatively coupled to the *host compute system* that *integrates* a *router module* and a *compute node*. AMD1004, [0032], [0096]–[0097], FIG. 9; AMD1003, ¶¶117–118 (*citing* AMD1006, [0038] (describing a “card” comprising a “printed circuit board” that includes a “router” and “hardware acceleration units”)).



AMD1004, FIGS. 1, 2B Combination Figure; AMD1004, FIG. 9.

Dalal’s printed circuit board is similar to the interface card of the ’750 patent, which is also described as a “printed circuit board (PCB)” that integrates multiple different components. AMD1001, 10:59–11:7, 13:5–23; AMD1003, ¶119 (*citing* AMD1024, 8:41–60). Furthermore, the ’750 patent acknowledges that it is “generally known” for a PCB integrating a router module and compute node to “cooperate with structural components of [a] host compute system.” AMD1001, 10:59–67.

**B. GROUND 1B: Dalal-Ng-Raghava Renders Obvious Claims
3–4**

1. Raghava

Raghava’s system includes a “host processor coupled to a programmable acceleration coprocessor” that are “used to accelerate a variety of functions...[including] convolutional neural networks.” AMD1006, [0017]–[0039] FIGS. 1, 3–4; AMD1003, ¶120. The coprocessor includes one or more hardware accelerators including “external input-output (IO)” blocks that use an “Open NAND Flash Interface (ONFI)...for communicating with memory.” *Id.*

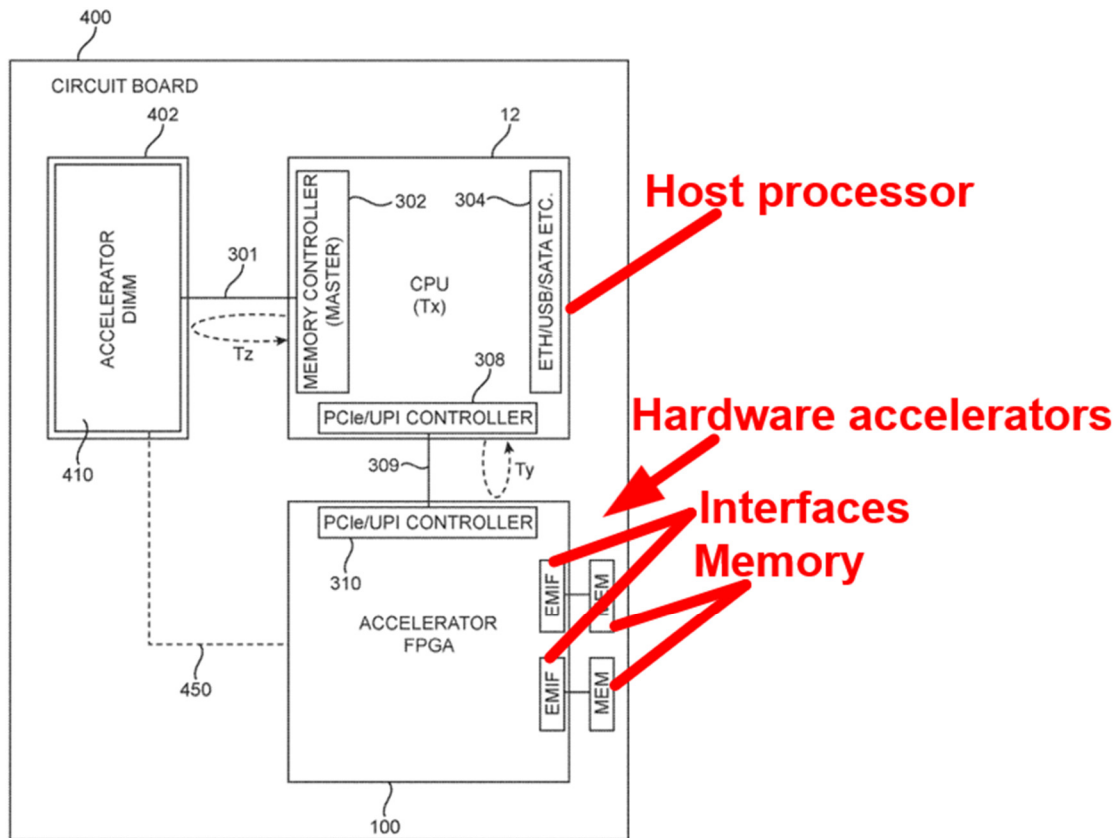


FIG. 4

AMD1006, FIG. 4.

2. Dalal-Ng-Raghava Combination

Dalal, Ng, and Raghava are analogous art to the '750 because all of these describe distributed systems where a host communicates with compute nodes having programmable logic to perform specialized operations. AMD1006, [0017]–[0022], [0030]–[0049], FIGS. 1, 3–4; *supra* §§V.A.1–3; AMD1003, ¶121.

Combining the teachings of DNC with the NAND flash memory and ONFI storage interfaces of Raghava merely involves the application of a known technique to a known system to achieve predictable results, as demonstrated by the fact that

Dalal and Ng both describe hwa modules including non-volatile memory (e.g., flash). AMD1004, [0033], [0067], [0096], [0105]; AMD1005, [0032], [0034], [0080]; AMD1003, ¶122. A POSITA would have reasonably expected success in combining Dalal and Ng because both describe a host that communicates with hardware accelerators to perform a data processing operation. AMD1004, [0032], [0078]; AMD1005, [0024].

The following reasons, taken separately or collectively, support the Dalal-Ng-Raghava Combination (DNRC). AMD1003, ¶123.

First, a POSITA would have understood that non-volatile memory such as flash provides multiple benefits including greater capacity and ability to maintain storage in the case of power loss. AMD1003, ¶124 (*citing* AMD1022, 1 (explaining that “non-volatile memory” such as “flash” has “notable features such as non-volatility” and “fast write speed,” and explaining that “flash memory....has benefits of a large capacity and relatively high performance”); AMD1038, 1 (“large capacity flash memory”)). Here, Dalal contemplates that hardware accelerator memories “can be of any suitable type” such as “RAM” or “flash memory.” AMD1004, [0067]. Implementing Dalal’s hwa modules with Raghava’s NAND flash memory would have beneficially increased memory capacity and removed power dependency on continued storage using a “suitable” type of memory. AMD1003, ¶124.

Second, because Raghava explains that “external IO blocks 312” “may include memory controllers for supporting a wide variety of external memory protocols” including “ONFI,” a POSITA would have been motivated to implement the storage interface component of DNC with aspects of Raghava’s IO blocks to ensure that any kind of memory connected to the accelerator is paired with a compatible interface. AMD1006, [0033]; AMD1003, ¶125. Dalal and Ng describe hwa modules having multiple different kinds of memory, and a POSITA would have understood that Raghava’s teachings are helpful for managing this dynamic. AMD1004, [0067]; AMD1005, [0032]; AMD1003, ¶125.

Third, a POSITA would have understood that NAND flash memory and ONFI represent one of a finite number of predictable solutions for configuring memory of an hwa module. AMD1003, ¶126. Indeed, Dalal and Ng describe hwa modules that include flash memory. AMD1004, [0067]; AMD1005, [0032]; AMD1003, ¶126.

3. Analysis

Element [3]

The ’750 patent explains that “all Flash is EEPROM (as a technical matter),” suggesting that any flash memory device includes an **EEPROM device**. AMD1001, 10:11–34; AMD1003, ¶127. Dalal’s second memory (HW RAM) corresponds to the **data store**. AMD1004, [0075], [0080]–[0082], [0094], cls.1, 11; *supra* Element [1d]; *supra* §V.B.2. For the reasons noted above, in the DNRC, the **data**

store of each *compute node* would have been implemented with Raghava's accelerator FPGA, which includes an *EEPROM*. *Id.*

In particular, Raghava discloses that an accelerator FPGA (*programmable logic component*) includes external input-output (IO) blocks (*storage interface component*) connected to a memory (*data store*) AMD1006, [0033]–[0034], [0014], [0046], [0074], cl.10, FIGS. 3–5; AMD1003, ¶128. Raghava discloses that the IO blocks support the “Open NAND Flash Interface (ONFI)” and are “suitable interfaces for communicating with memory,” indicating that the memory includes NAND flash (*EEPROM device*). *Id.*

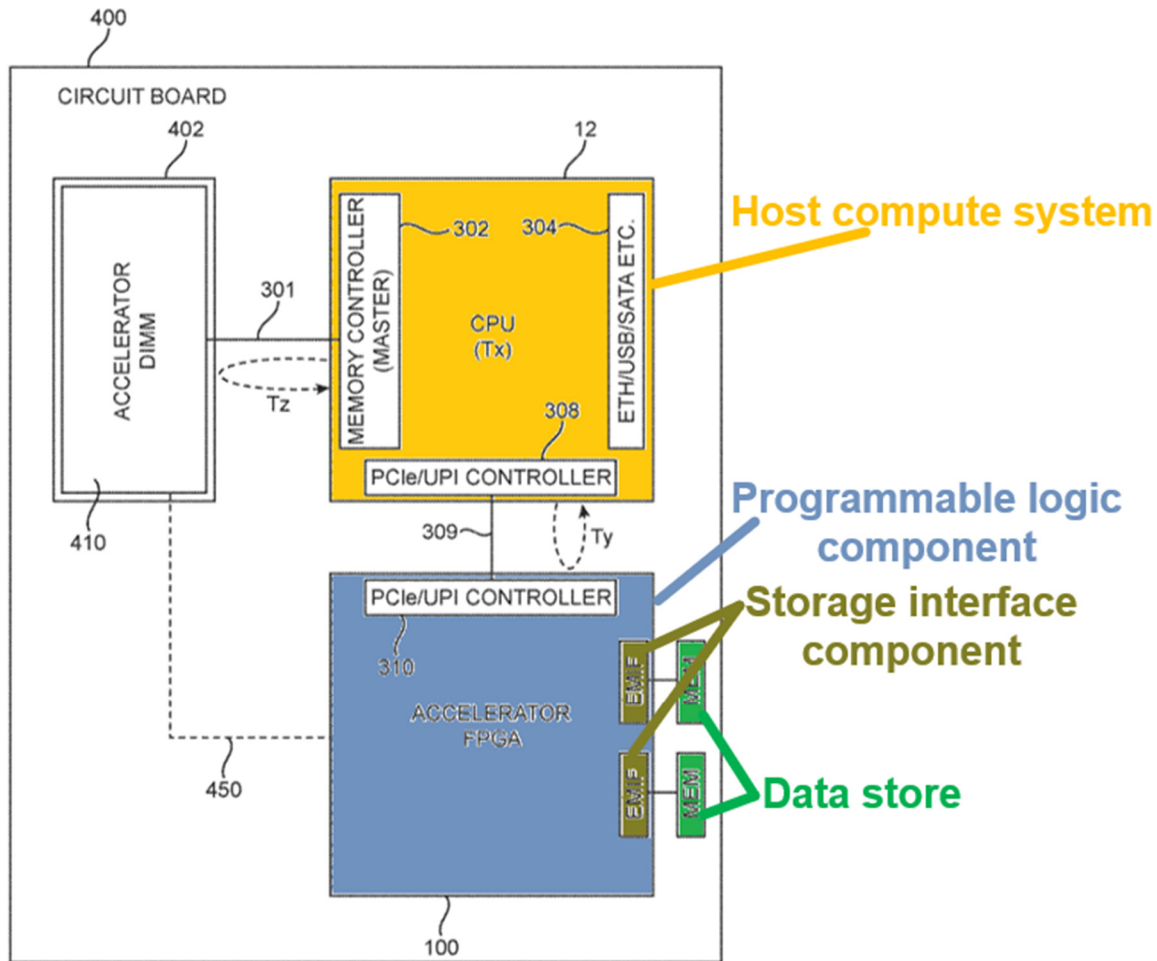


FIG. 4

AMD1006, FIG. 4.

Element [4]

Dalal's data transfer fabric, or the combination of the data transfer fabric with Ng's memory interfaces corresponds to the *storage interface component*. AMD1004, [0075], [0079]–[0083], [0089]–[0090], [0094], cl.1, 18–20; AMD1005, [0035]–[0036], [0038]; *supra* Elements [1h]–[1i]; AMD1003, ¶129. For the reasons noted above, in DNRC, the *storage interface component* of each *compute*

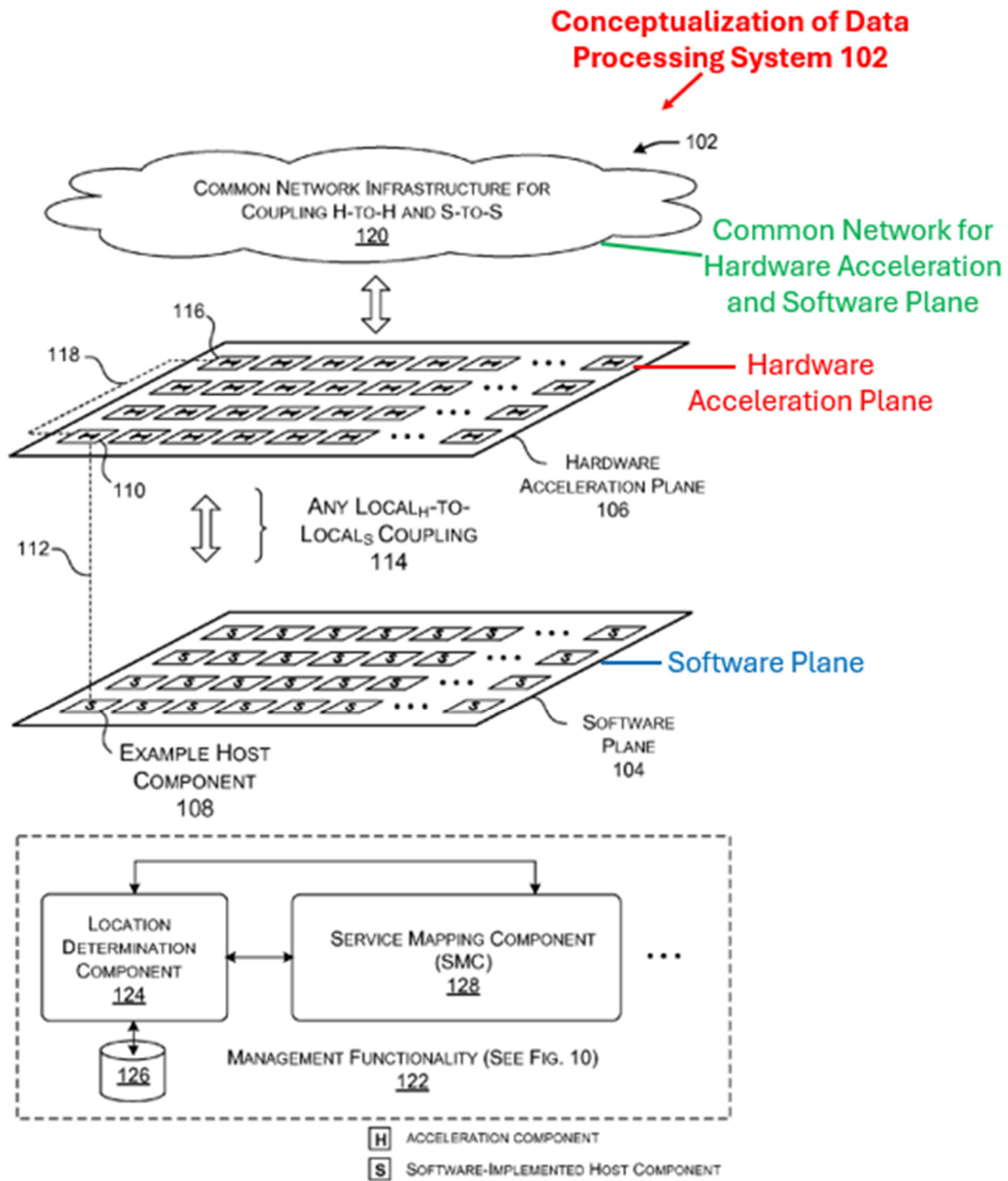
node would have been implemented with Raghava's input-output (IO) blocks that employ *ONFI*. AMD1006, [0033], [0014], [0046], [0074], cl.10, FIGS. 3, 5; *supra* Element_[3]; AMD1003, ¶130; *supra* §V.B.2. A POSITA would have found it obvious to make this combination, because doing so would have provided nonvolatile storage at increased capacity. *Id.*

C. GROUND 2A: Heil-Jun Renders Obvious Claims 1–3, 5–8

1. Heil

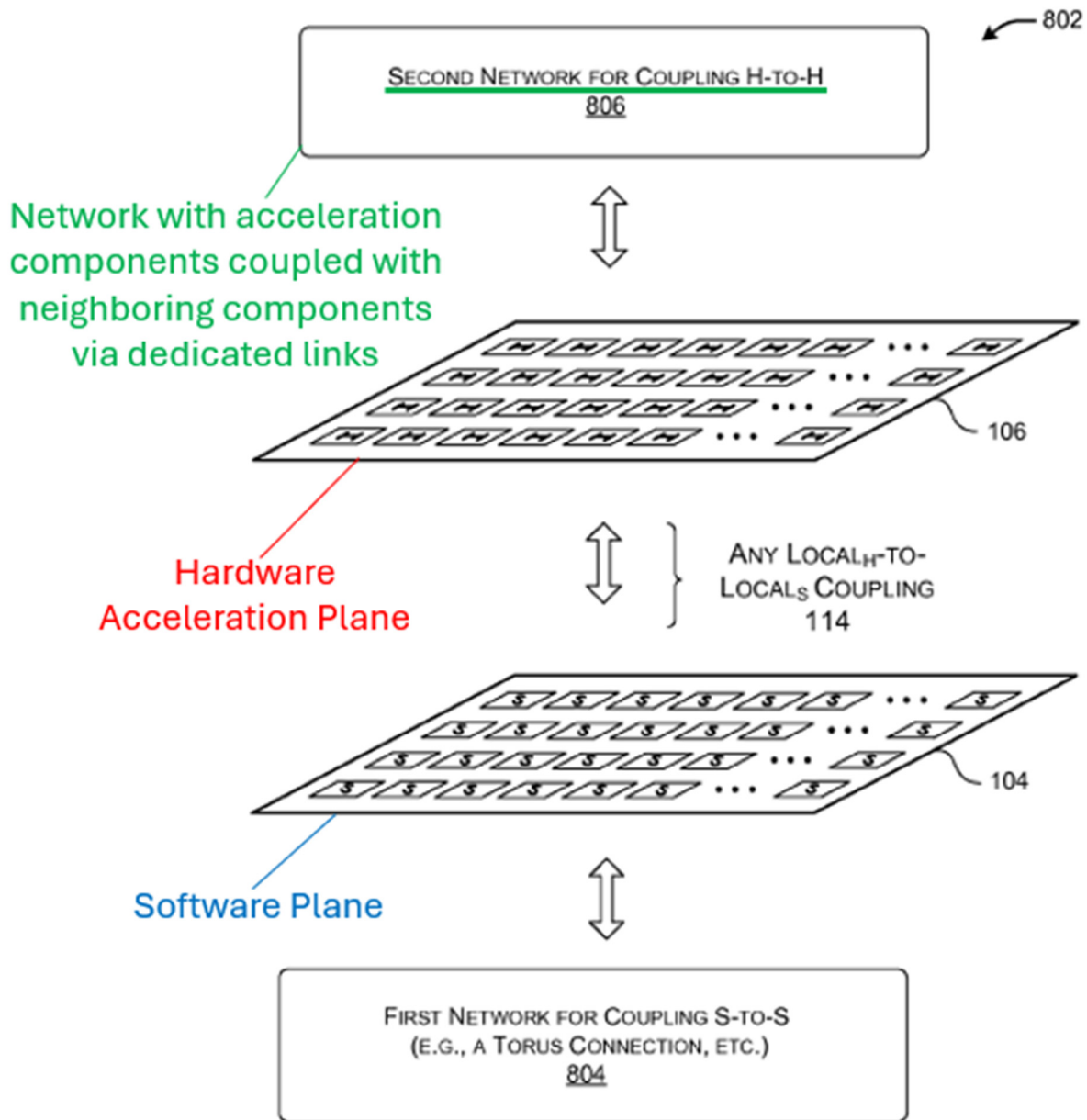
Heil discloses data processing systems for implementing multi-component services. AMD1010, [0003], [0040], [0053], [0218]–[0248]. The systems include multiple server unit components, each having a host component paired with an acceleration component. *Id.* These server unit components are communicatively connected to execute services. *Id.*; AMD1003, ¶131.

Heil's data processing systems (*e.g.*, 102 and 802 shown below) can be conceptualized as having a first logical network forming a software plane 104 with the host components and a second logical network forming a hardware plane 106 with the acceleration components. AMD1010, [0048]–[0058], [0089], FIGS. 1, 8. In Figure 1 (below), the two logical networks share a common network. *Id.*; AMD1003, ¶132.



dedicated links to couple neighboring acceleration components. AMD1010, [0089], [0014], [0084]; AMD1003, ¶133.

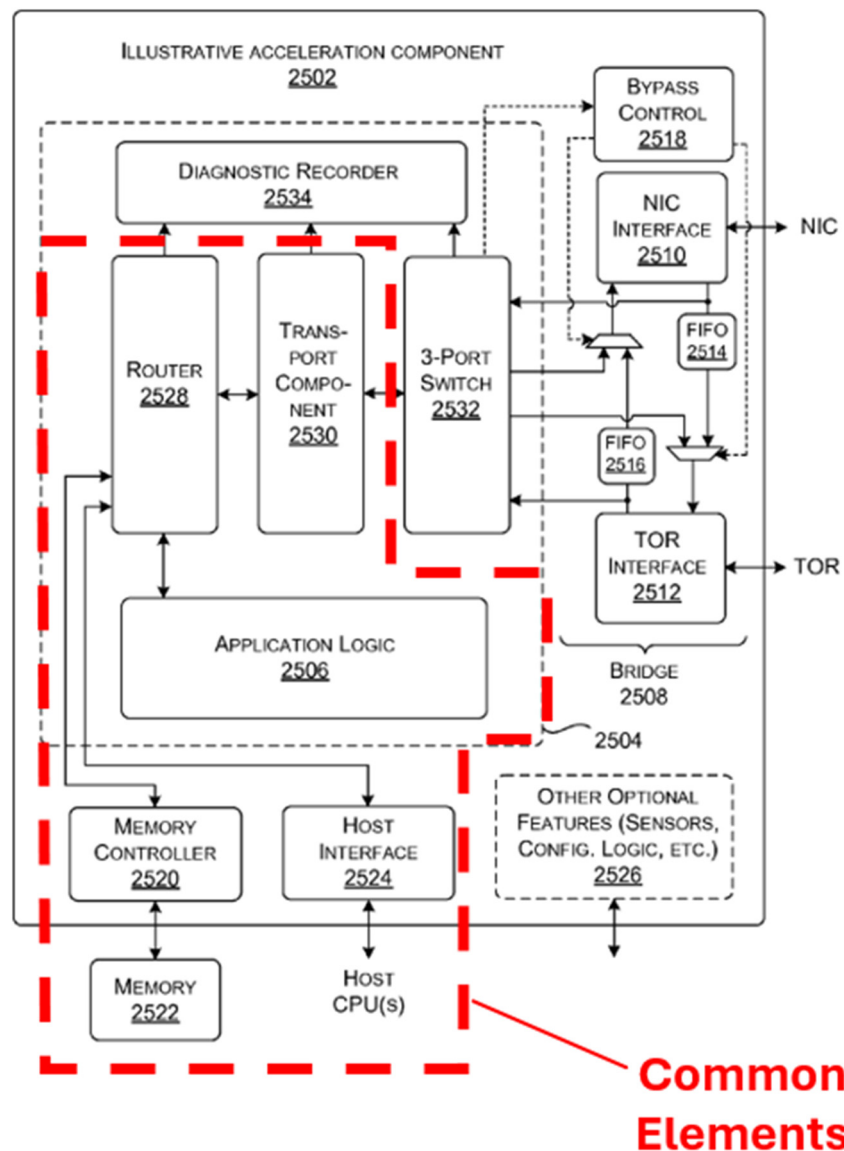
**Conceptualization of Data
Processing System 802**



AMD1010, FIG. 8.

Both Figures 1 and 8 represent Heil's acceleration components with “H” in hardware acceleration plane 106. Figure 25 (below) shows an illustrative example

of these acceleration components that includes application logic 2506, memory controller 2520, memory 2522, host interface 2525, router 2528, and transport component 2530. AMD1010, [0022], [0159]–[0168]. A POSITA would have understood or found obvious that at least these elements are common across acceleration components referenced in systems 102 and 802. AMD1003, ¶134.

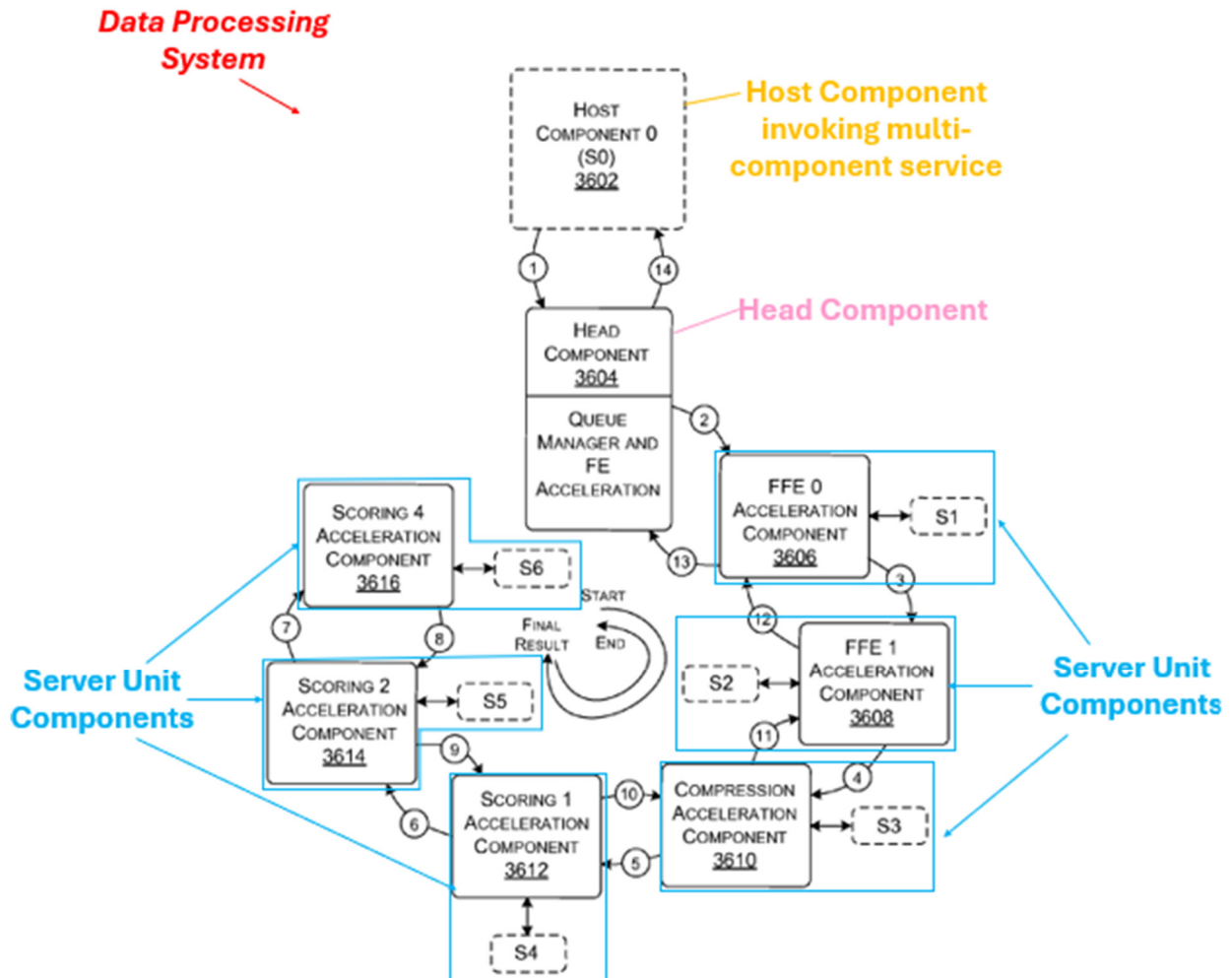


AMD1010, FIG. 25.

For example, while Heil states that Figure 25 is implemented for system 102, Heil also explains that “other data processing systems [*e.g.*, system 802] may use acceleration components having architectures which vary in one or more ways.” AMD1010, [0159]–[0162]. In view of this disclosure, a POSITA would have understood or found obvious that Heil’s acceleration forms the second network via dedicated links (*e.g.*, at TOR interface 2512) and would have still utilized the above identified common elements to implement core capabilities. AMD1003, ¶135 (describing the core capabilities of each common elements).

A POSITA would have also found it obvious to implement these common elements with system 802 because utilizing the same elements to fulfill the same core capabilities would have allowed system 802 to operate. *Id.*; *Bos. Sci. Scimed, Inc. v. Cordis Corp.*, 554 F.3d 982, 991 (Fed. Cir. 2009) (“Combining two embodiments...in a prior art patent does not require a leap of inventiveness.”). A POSITA would have also understood that implementing these common components across systems 102 and 802 would have provided similar benefits as the described outer/inner shell architecture does “across application scenarios” including negating the need to “recreate these common components” and reducing the risk that a “core inner or outer shell functions [is altered] in a manner that causes problems within the data processing system.” AMD1003, ¶¶136–137; AMD1010, [0159]–[0162].

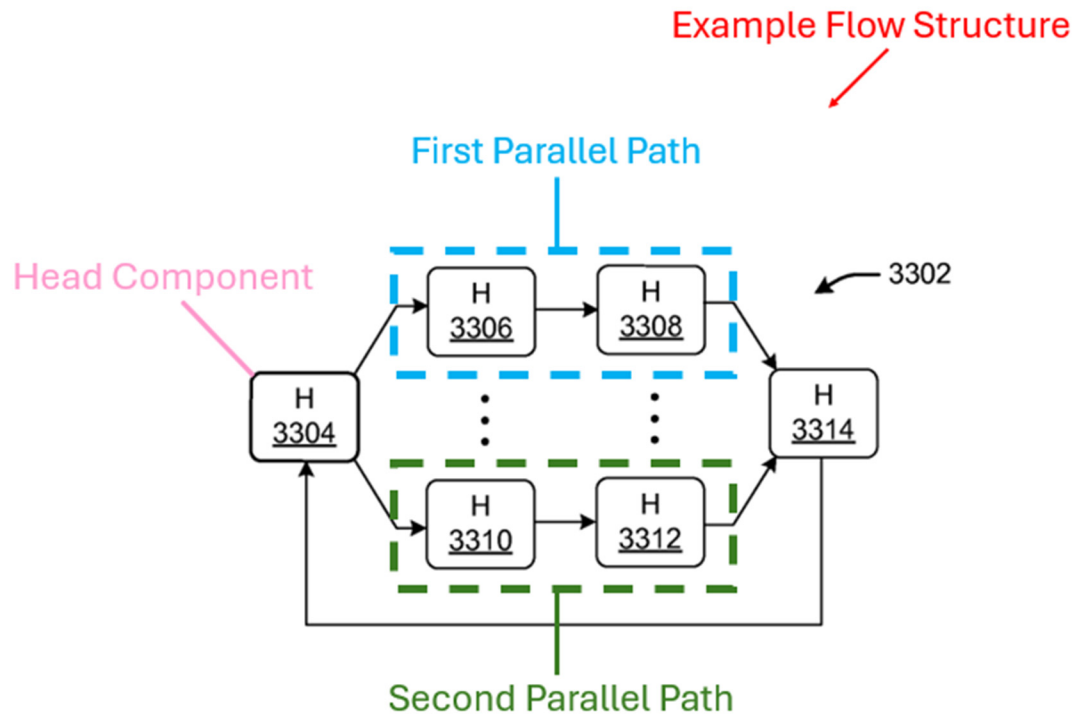
Turning to an example of a multi-component service, Figure 36 (below) shows a service for assigning “a ranking score to a pairing of a query and a document.” AMD1010, [0218]–[0229], FIGS. 36, 40. A host component 3602 invokes the service by sending a request to head component 3604. *Id.* Server unit components execute the invoked service in a processing pipeline with coupled acceleration components (*e.g.*, 3606–3616). *Id.*, [0068]; AMD1003, ¶138.



AMD1010, FIG. 36.

Heil explains that Figure 36 (above) utilizes a ring flow structure and that “many other flow structure [are] possible.” AMD1010, [0223], [0069], [0073]–[0074], [0200]. Figure 33 (below) shows an example of one such other flow structure featuring multiple parallel paths branching from head component 3304. AMD1010, [0198]–[0203], [0216]–[0217], FIG. 33⁴; AMD1003, ¶¶139–141. In this example, one path consists of acceleration components 3406/3408 and a second path comprises components 3610/3612. *Id.*

⁴ Heil discloses that each acceleration component is paired with a “local host component” within a server unit component. AMD1010, [0053], [0089], FIGS. 6, 8. Accordingly, although not shown in Figure 33, each of the acceleration components in the collection 3302 is paired with a “local host component” as part of a server unit component. *Id.* AMD1003, ¶140.



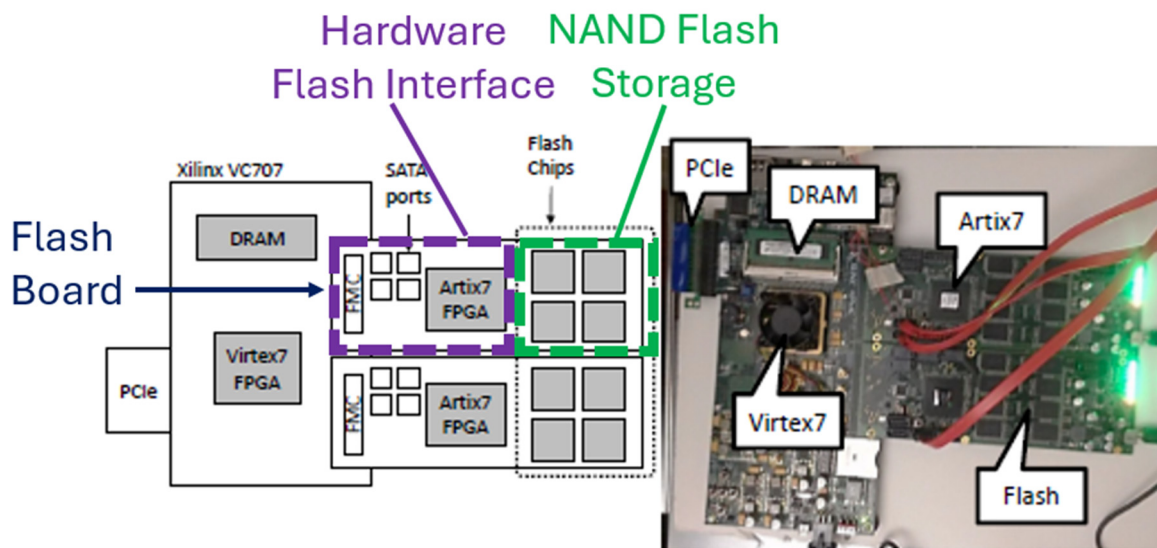
AMD1010, FIG. 33.

2. Jun

Jun qualifies as prior art at least because it is a technical paper published in the Proceedings of the 42nd Annual International Symposium on Computer Architecture, and publicly accessible at least by 2016, years before the Critical Date. AMD1013, ¶¶6-9 (explaining that Jun was made available to the public shortly after April 6, 2016).

Jun describes a data processing system that includes multiple FPGA-based compute nodes. AMD1011, pp.1-5, 7-9, 12-13. Jun's nodes integrate NAND Flash Storage (Flash) with an FPGA processor device to execute analytics operations on large data sets (*e.g.*, Big Data analytics). *Id.* Jun explains that it is advantageous to

use Flash in place of DRAM memory to effectively handle tasks on large datasets. AMD1011, pp.2–8, 12. Jun’s Flash is deployed in conjunction with a multi-channel hardware flash interface and connectors to the FPGA processor on a flash board. *Id.* The hardware interface and connectors on the flash board facilitate communication between Jun’s FPGA Accelerator and Flash. *Id.*; AMD1003, ¶¶142–143



AMD1011, FIG. 10.

3. Heil-Jun Combination

Heil and Jun qualify as analogous art because they share the '750 patent's field of endeavor in data processing systems employing reconfigurable hardware. AMD1003, ¶144; AMD1001, 2:28–45, 3:5–25, 37:44–55; FIG. 17; *supra* §§V.C.1–2. They are also reasonably pertinent to a technical problem shared with the '750 patent of accelerating processing by offloading tasks to reconfigurable hardware.

Id.; AMD1010, [0051], [0224], AMD1011, pp.1–6. Jun even states that it “has similar goals in terms of reconfigurable hardware acceleration” to Heil’s FPGA/accelerator formed network. AMD1011, p.4⁵; AMD1003, ¶145.

Heil’s data processing system utilizes multiple server unit components each comprising a host component and an FPGA-based acceleration component. *Supra* §V.C.1. In this context, a POSITA would have been motivated to implement Jun’s flash board—including its Flash, and hardware flash interface, and connectors—in place of Heil’s DRAM memory elements. AMD1003, ¶146.

⁵ Jun references AMD1014 describing Microsoft Project Catapult. AMD1011, pp.4, 14; *generally* AMD1014. This project refers to the same technology as Heil. AMD1003, ¶145.

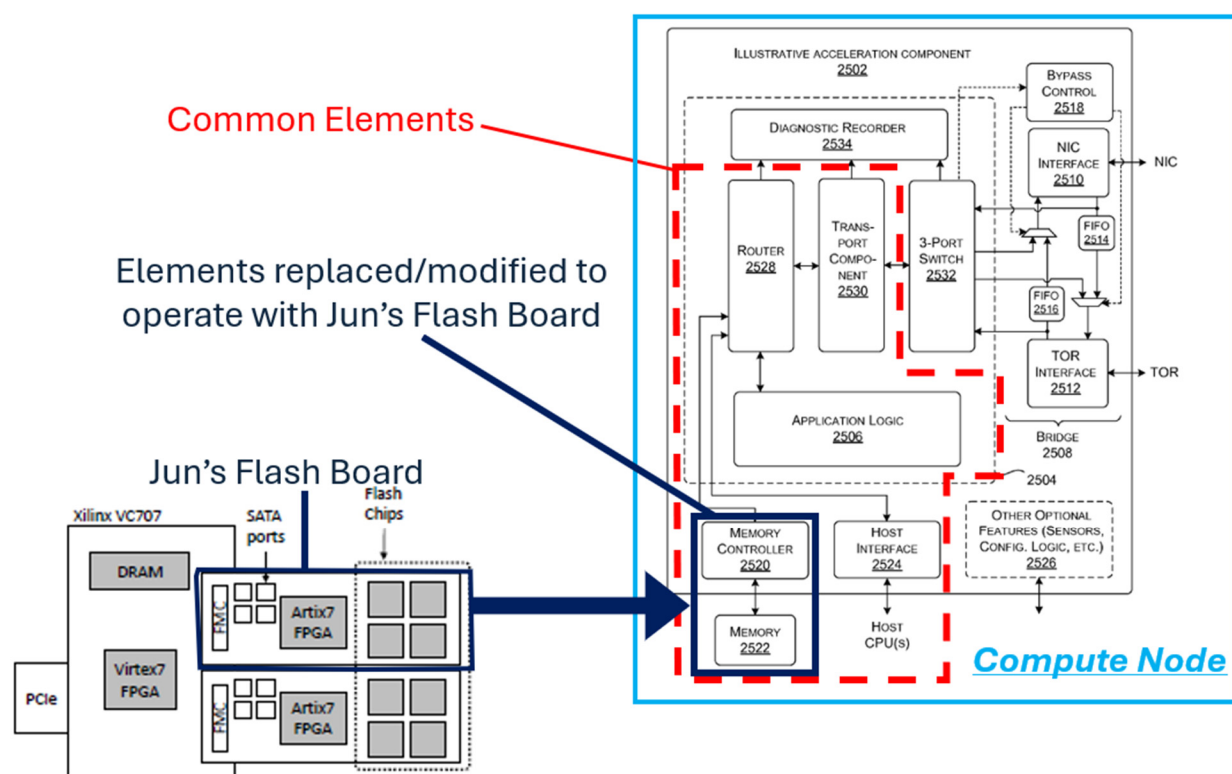


Illustration of the Heil-Jun Combination (HJC Illustration).

The following reasons, taken separately or collectively, support the Heil-Jun Combination (**HJC**). AMD1003, ¶¶147–153.

First, a POSITA would have understood that implementing Jun's flash board elements allows Heil's data processing system to accelerate multi-component services on larger datasets. AMD1003, ¶148. Jun explains that hosting large datasets on DRAM-based acceleration components requires a large cluster of compute nodes. AMD1011, pp.2–4, 12. Replacing DRAM with Flash enables high-capacity data hosting on fewer compute nodes, significantly reducing the required number of compute nodes or expanding the capacity of a given collection of compute

nodes. AMD1011, pp.3–4; AMD1003, ¶148. A POSITA would have been motivated to implement HJC to accelerate services on datasets that exceed the DRAM capacity for the collection of nodes in Heil’s data processing system. *Id.*

Second, a POSITA would have recognized that implementing Jun’s flash board elements within Heil’s acceleration component facilitates a reduction in architectural complexity by allowing implementations of services to require fewer server unit components. AMD1003, ¶149. Indeed, it would have been obvious to the POSITA that replacing Heil’s DRAM with Jun’s Flash would have allowed for a reduced number of server unit components because each NAND-Flash-based acceleration component can handle a larger dataset locally. *Id.*; AMD1011, Abstract. Reducing the number of server unit components provides at least the following benefits: simplifies the system’s physical wiring, reduces operational latency and network traffic, reduces maintenance requirements, and makes the system easier to implement and use. *Id.*

Third, it would have been obvious to a POSITA that implementing Jun’s “flash board” elements in Heil’s acceleration component would have facilitated a reduction in capital cost because NAND Flash is a more cost-effective memory medium than DRAM. AMD1003, ¶150. As Jun explains, NAND Flash Storage is “an order of magnitude cheaper price compared to DRAM.” AMD1011, p.3, Abstract, 4, 12; AMD1003, ¶150 (*citing* AMD1037).

Fourth, a POSITA would have recognized that replacing Heil’s DRAM-based storage with Jun’s flash board elements facilitates improved power efficiency. AMD1003, ¶151. As Jun explains, Flash “consumes an order of magnitude less power” than the DRAM-based alternative. AMD1011, pp.3, 11; AMD1003, ¶151.

Fifth, a POSITA would have found it obvious to combine the teachings of Heil with Jun because the combination merely involves the application of a known technique to a known system to achieve predictable results. *KSR Int’l Co. v. Teleflex Inc.*, 550 U.S. 398, 417 (2007); AMD1003, ¶152. Jun recognized a problem with existing DRAM-based acceleration components and described a straightforward technique to address it using Flash. AMD1003, ¶152. In this context, combining the teachings of Heil and Jun would have been an obvious simple substitution of known elements that a POSITA would have reasonably expected to succeed at least because both systems utilize similarly networked compute nodes that, in both references, utilize FPGA-based acceleration to perform similar processing tasks. AMD1003, ¶153.

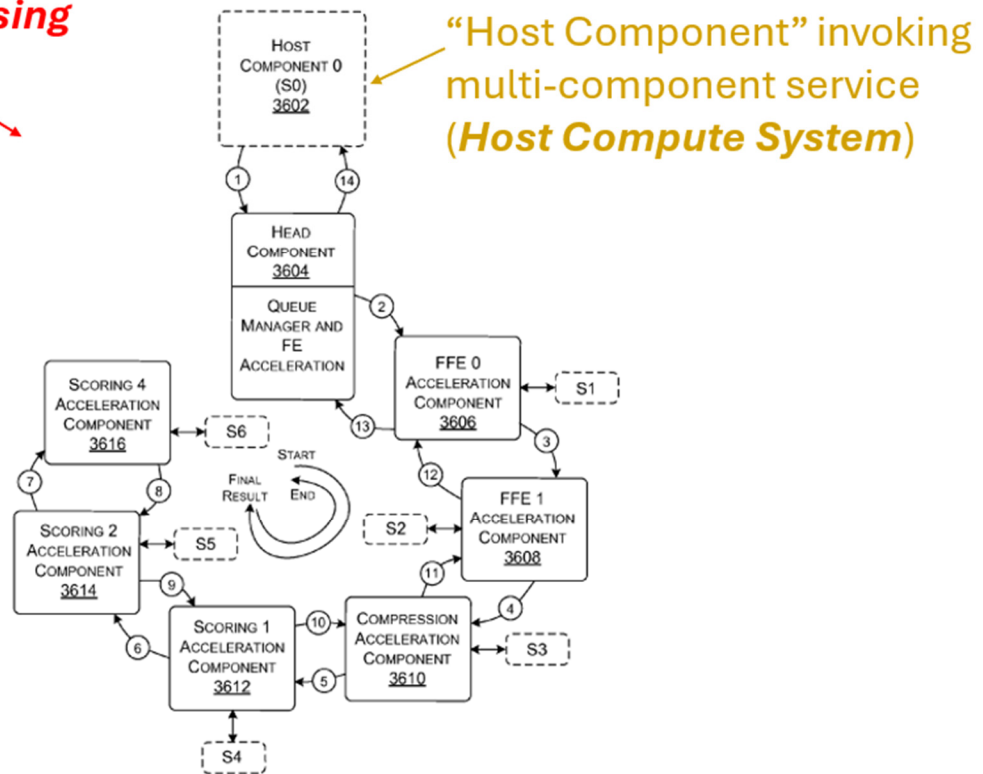
4. Analysis

Element_[1pre]

Heil describes a ***data processing system*** that provides a multi-component service. AMD1010, [0040], Abstract, [0003], [0203], [0218]–[0223], FIGS. 33, 36. In the example shown in Figure 36, Heil explains how a host component 3602 (***host***

compute system) *cooperates* with the rest of the *data processing system* to “invoke [a] service by sending a request to a first acceleration component of the multi-component service, corresponding to a head component 3604.” AMD1010, [0219]–[0220], [0236], [0251], FIG. 36; *cf.* AMD1001, 8:28–57 (“Host compute system 199...comprise[s] a computer server” that “initiate[s], instantiate[s], or otherwise to request[s] data processing operations to be executed by system 100.”). Examples of Heil’s data processing operation are described below in Elements_[1a]–[1i]. AMD1010, [0198]–[0205], [0217]–[0223] (“FIG. 36 reflect the sequence of *operations*”), [0229], FIGS. 33, 36; [0069]–[0070], [0157], [0171]. Because the host component performs computations to invoke the service and the data processing system performs data processing operations for the service, HJC renders obvious Element_[1pre]. AMD1003, ¶¶154–155; AMD1010, [0079] (“a program running on the host component 412 requests a service.”) [0068]–[0069], [0074], [0079], [0198]–[0203].

**Data Processing
System**



AMD1010, FIG. 36.

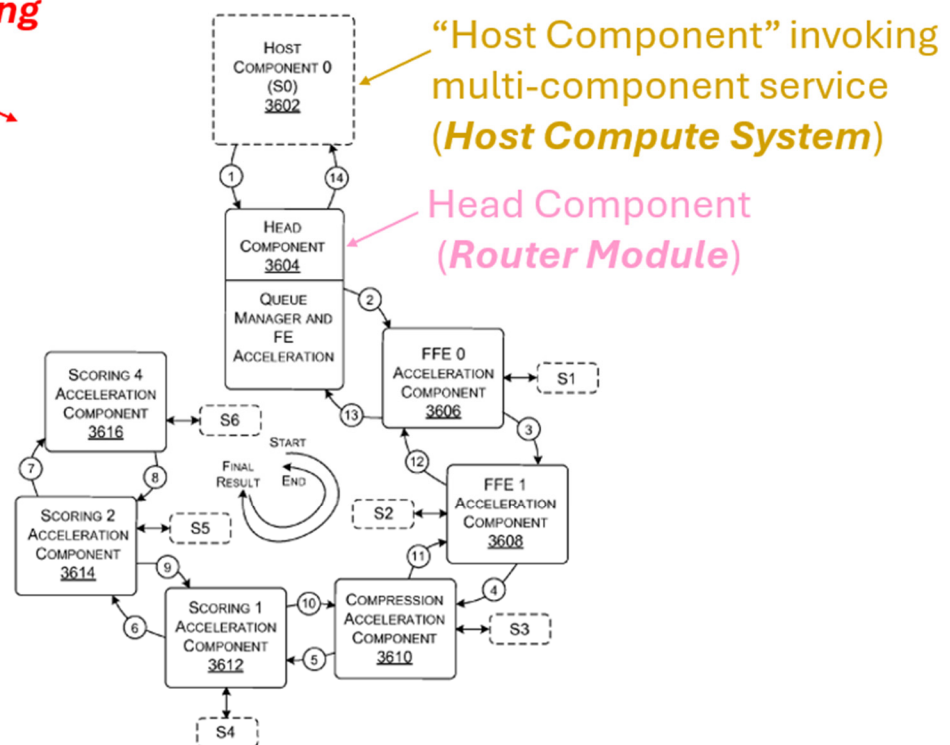
Element [1a]

As discussed for Element [1pre], Heil’s host component (*host compute system*) invokes a multi-component service by sending a request to “a first acceleration component corresponding to a head component.” AMD1003, ¶156; AMD1010, [0219]–[0220], FIG. 36.

It would have been understood or obvious to a POSITA that the head component corresponds to a *routing module* in Heil’s data processing system at least because the head component *routes* communications between the requesting host and the server unit components implementing the service. AMD1003, ¶157;

AMD1010, [0068] (“The head component corresponds to a point of contact by which entities in the data processing system 102 may interact with the multi-component service...”), [0203] (“a head component 3304 that branches out to two parallel paths”), [0219]–[0223], [0236], [0251], [0167]–[0168], [0176]–[0186], FIGS. 25, 27–28, 36, [0074], [0089], [0095]. Like the *router module* described in the ’750 patent, Heil’s head component 3604 is FPGA-based and facilitates communication between the requesting host and server unit components, including by controlling the flow/queue of requests/commands. AMD1010, [0048]–[0050], [0160], [0203], [0225]–[0227], [0252], FIGS. 33, 36; AMD1001 8:58–9:37.

**Data Processing
System**



AMD1010, FIG. 36.

As shown in the Figure 36 example (above), the head component 3604 (*router module*) is *communicatively coupled* to the host component 3602 (*host compute system*) and communicates with the host component to receive the request from the host component 3602 in operation (1) and *route* the final score to the host component 3602 in operation (14). AMD1010, [0218]–[0223], [0200], [0203], [0223] (“other flow structure [are] possible”), FIG 33 (showing an alternative flow structure with head component 3304); *supra* §V.C.1; AMD1003, ¶158.

Heil’s head component (*router module*) includes a *host interface*. AMD1010, [0165], [0167], [0173]–[0175], [0203], [0219], FIGS. 25, 27, 33, 36; AMD1001, 8:58–65 (explaining that the “various communications interfaces” were “generally known in the art”); AMD1003, ¶159. Heil’s head component (*e.g.*, 3604 in Figure 36 and 3304 in Figure 33) is a type of acceleration component, and Heil’s acceleration components include a *host interface* for interacting with its local/requesting host component (*e.g.*, host component 3602 in Figure 36). *Id.*; see §V.C.1.

Heil teaches or renders obvious that the head component (*router module*) includes a *node interface* to communicate with the server unit components. AMD1010, [0203], [0218]–[0223], FIGS. 33, 36 (operation “2” and operation “13” involving the head component communications with acceleration component 3606 directly and acceleration components 3608, 3610, 3612, 3614, 3616 indirectly);

AMD1003, ¶160; AMD1001, 9:7–14 (“node interfaces” are “generally known in the art”), 42:4–11. Heil’s acceleration component—*e.g.*, head component 3604—includes a transport component with “a register *interface* to establish connections between *nodes*.” AMD1010, [0181], [0048], [0148], [0167]–[0168], [0174], FIGS. 25, 27, 29; *supra* §V.C.1.

Element_[1b]

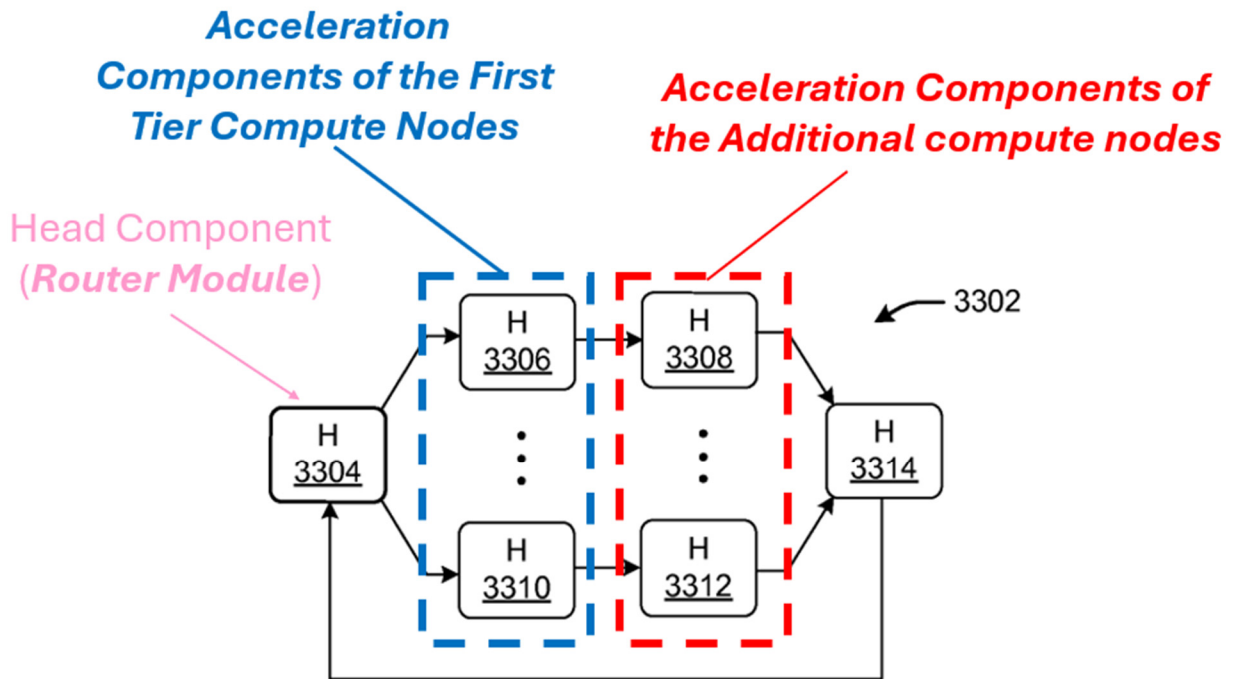
HJC satisfies Element_[1b]. AMD1003, ¶161; AMD1010, [0040], [0065]–[0069], [0073]–[0089], [0099]–[0100]; [0114]; [0160]–[0170]; [0192]; [0198]–[0205]; [0218]–[0225]; [0229]–[0236], [0248]–[0251], FIGS. 1–5, 8, 10, 16, 30, 32–33, 36, 38–40.

As just one example and as discussed in §V.C.1, Heil’s Figure 33 demonstrates an example flow structure of a multi-component service with multiple *parallel* paths, of server unit components (*compute nodes*), as represented by their acceleration components, branching from the head component 3304. AMD1010, [0203], FIG. 33 (below)⁶; [0068], [0074], [0211], [0217]–[0223] (explaining that the

⁶The ellipses between acceleration components 3306/3308 and 3310/3312 represents additional paths of acceleration components communicatively coupled in series branching from head component 3304. AMD1003, ¶163; AMD1010, [0211] (“two

example multi-component service shown in Heil's Figure 36 can have any other flow structure, which would have included the flow structure shown in Heil's Figure 33.), [0234], [0250], FIGS. 3, 36; AMD1003, ¶162. Each *parallel* path includes a *first tier* server unit component, as represented by acceleration components 3306 and 3310 in Figure 33. AMD1010, [0203], FIG. 33; AMD1003, ¶162. As shown in Figure 33, the *first tier* server unit components represented by acceleration components 3306 and 3310 are each coupled to an *additional* server unit component (*compute node*) in *series* as represented by their corresponding acceleration components 3308 and 3310. *Id.*

or more acceleration components that operate in parallel”), [0216]–[0217], [0234], [0250].



AMD1010, FIG. 33.

In Heil, each acceleration component within a respective downstream server unit component (*compute node*) can be “*coupled* to east, west, north, and south neighboring acceleration components via appropriate cable links or the like” in a two-dimensional torus network. AMD1010, [0089] (“*p*×*r* torus network”), FIG. 8. Heil explains that these “dedicated links for connecting the acceleration components together” can have “any network topology” and that “[o]ther types of torus networks can alternatively be used having any respective sizes and dimensions.” *Id.*; AMD1003, ¶164.

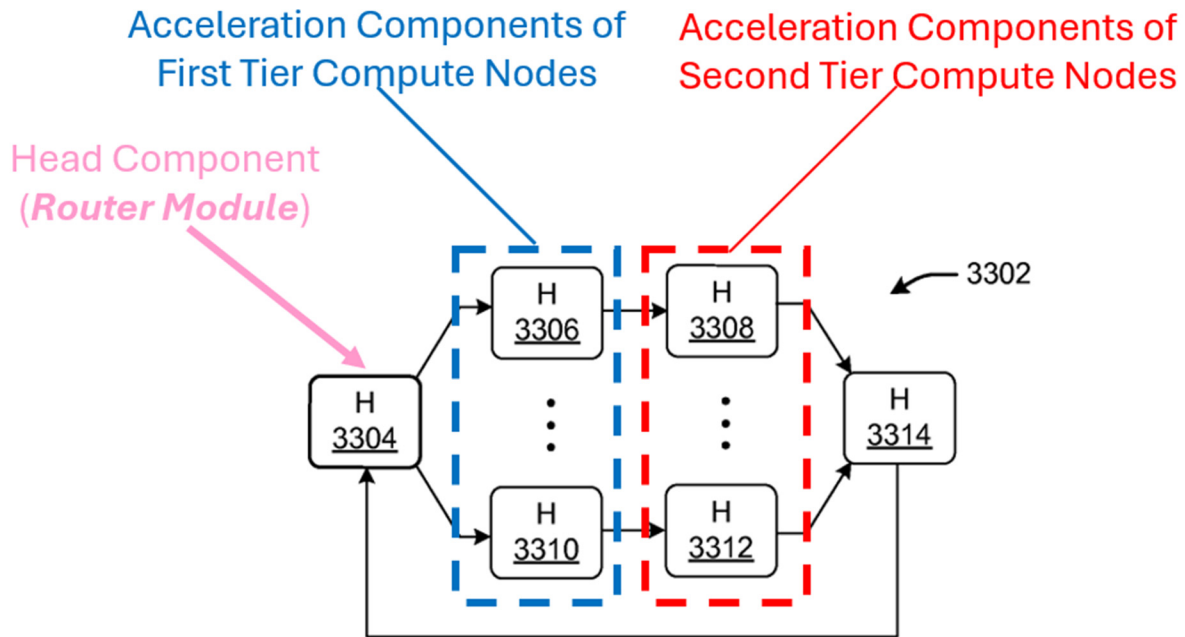
Under Heil’s broad disclosure covering “any network topology,” it would have been obvious to a POSITA that, Heil contemplates *coupling* Heil’s *first tier*

server unit components (*compute nodes*) to the *additional* server unit components (*compute nodes*) in series—via their respective acceleration components—for many reasons. AMD1003, ¶¶165–168. First, Heil’s Figure 33 illustrates “simplified collection 3302 of acceleration components” with such a network topology where the corresponding additional server unit components (*compute nodes*) are connected to the neighboring *first tier* server unit components (*compute nodes*) in series. AMD1010, [0089], [0203], FIG. 33; AMD1003, ¶166. Second, Heil teaches that “any network topology” could be used for connecting acceleration components, and the topology illustrated in Figure 33 would have been within this scope and obvious for implementing the multi-component service. *Id.*; AMD1010, [0089], [0084] (“the principles set forth herein can be applied to smaller systems”), [0211] (explaining how parallel paths can be used to “avoid a bottleneck in processing”), [0217] (explaining how parallel paths can be used to “to alleviate the congestion”); AMD1003, ¶167. Third, the POSITA would have recognized that using this specific topology for the “simplified collection 3302 of acceleration components” would have reduced the cabling complexity of the data processing system. AMD1003, ¶168 (citing AMD1014, 3 (demonstrating the tradeoff between “routability and cabling complexity.”)).

Element_[1c]:

As discussed at Element_[1b], Heil's server unit components (***compute nodes***) include ***multiple parallel first tier compute nodes coupled to one or more additional compute nodes in series***. As discussed at Element_[1a], Heil teaches a head component (***router module***) with a ***node interface*** to communicate with the server unit components (***compute nodes***). AMD1010, [0181]; AMD1003, ¶¶169–170.

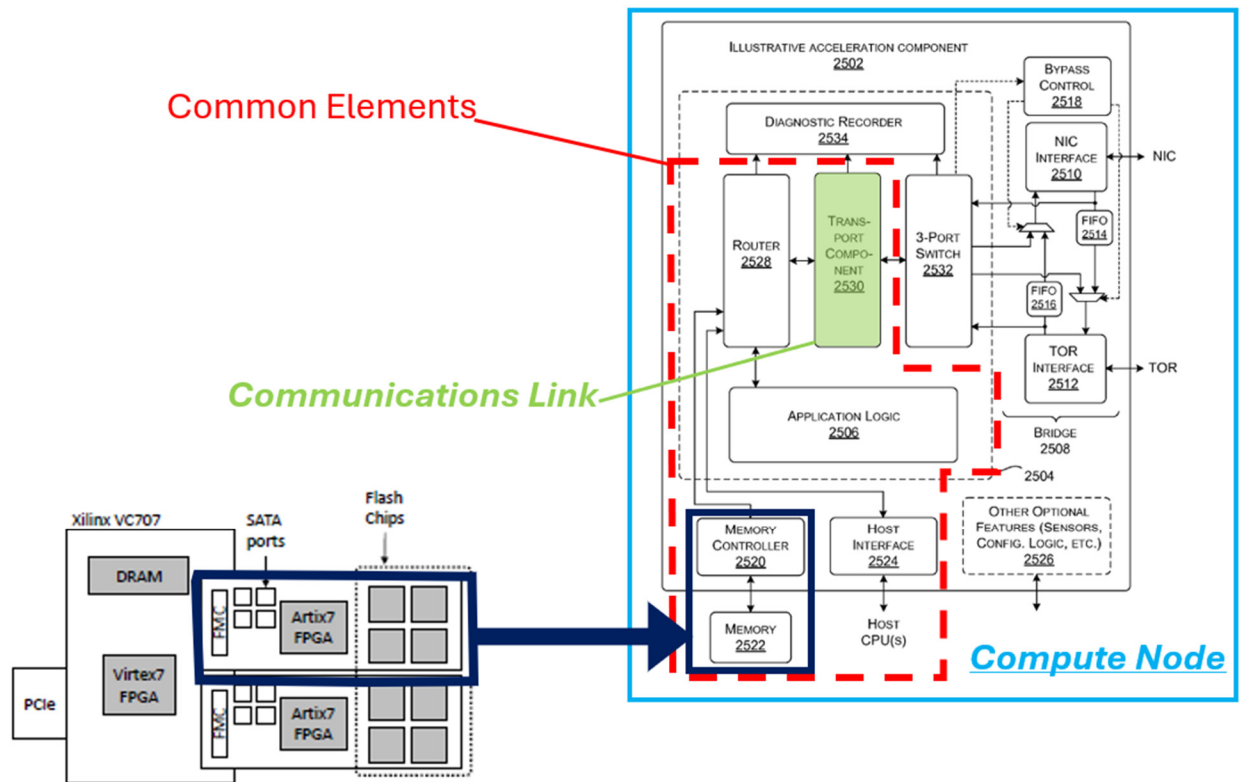
Heil's Figure 33 (below) illustrates how the head component 3304 (***router module*** using a ***node interface***) is ***communicatively coupled*** to the acceleration components 3406 and 3410 of the ***first tier*** server unit components (***compute nodes***) directly, which are then ***communicatively coupled*** to the acceleration components 3408, 3412 of the ***additional*** server unit components (***compute nodes***) ***in series*** and at a ***different successive tier***. AMD1010, [0203], [0221]–[0223], FIGS. 33, 36; *supra* §V.C.1. Each of these acceleration components establishes dedicated links to the neighboring acceleration components. AMD1010, [0089], [0203], [0068], [0221]–[0223], [0229], [0055], FIGS. 8, 33, 36, 40; AMD1003, ¶171.



AMD1010, FIG. 33.

A POSITA would have understood or found obvious that the acceleration component at each *successive tier* of Heil's server unit components (*compute nodes*), including a *first tier* and a second *tier* pictured in Figure 33 (above), includes a transport component, as shown in the "illustrative acceleration component" shown in Figure 25 (below). AMD1010, [0167]–[0168], [0181], FIG. 25; *see* §V.C.1; AMD1003, ¶172. The "transport component 2530...provide[s] a register interface to establish connections between nodes" and with other remote components, and thus functions as part of a *communications link* for communicating the server unit components (*compute nodes*) *connected at different successive tiers in series*. *Id.*

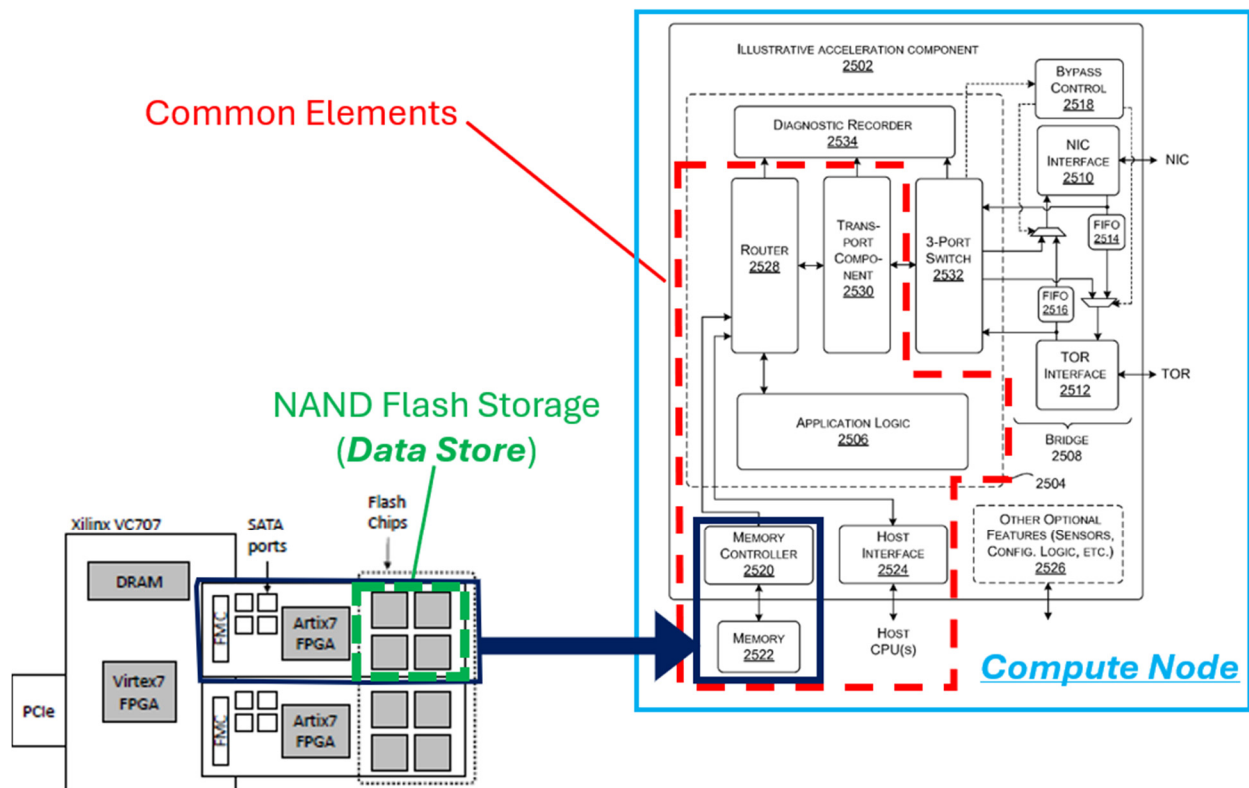
AMD1003, ¶172; AMD1010, [0203] (explaining that the parallel “branches can have any number of components” connected in series).



HJC Illustration.

Element_[1d]

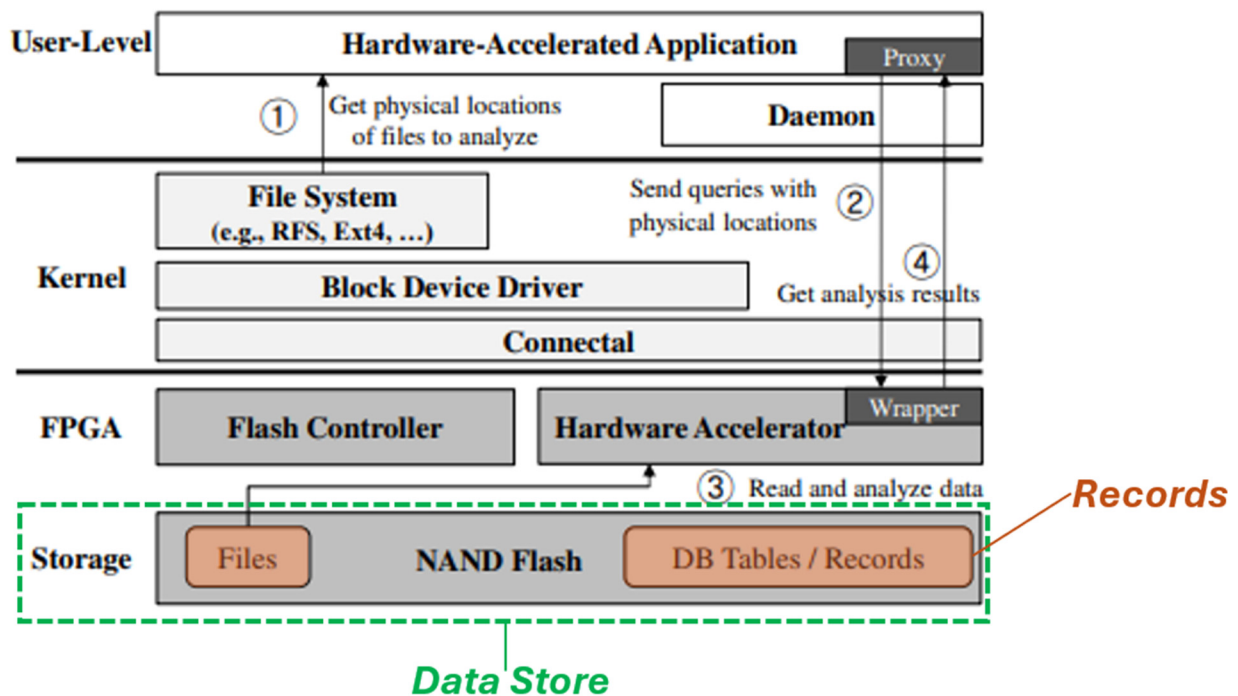
As discussed at §V.C.3, HJC integrates Jun’s teachings to utilize Flash with Heil’s acceleration components—in place of Heil’s acceleration component’s DRAM. AMD1003, ¶173.



HJC Illustration.

Jun provides teachings regarding the claimed *records associated with a data processing operation*. AMD1003, ¶174. Indeed, Jun’s Flash (*data store*) is shown as containing “Files” and “DB Tables/*Records*.” AMD1011, FIG. 8. Jun teaches that this information is read and analyzed by an FPGA based hardware accelerator, like Heil’ acceleration components. AMD1011, 4, 7–8, 10, 12, FIG. 8. Also like Heil, the data stored in the Flash (e.g., including the “Files” and “DB Tables/*Records*”) is associated with a *data processing operation*. AMD1011, pp.4 (“in-store processor to perform computation on the data”), 7 (“the in-storage

processors can directly read data from flash.”), 10 (“all of the data is stored in flash and the distance calculation is done by the in-store processor on the storage device.”), 12 (“...the accelerator, which uses these addresses to request for pages from the flash controller.”); AMD1001, 7:36–44 (“as is generally known in the art, these data are typically stored or maintained in files or datasets (or ‘records’ or ‘pages’)”).



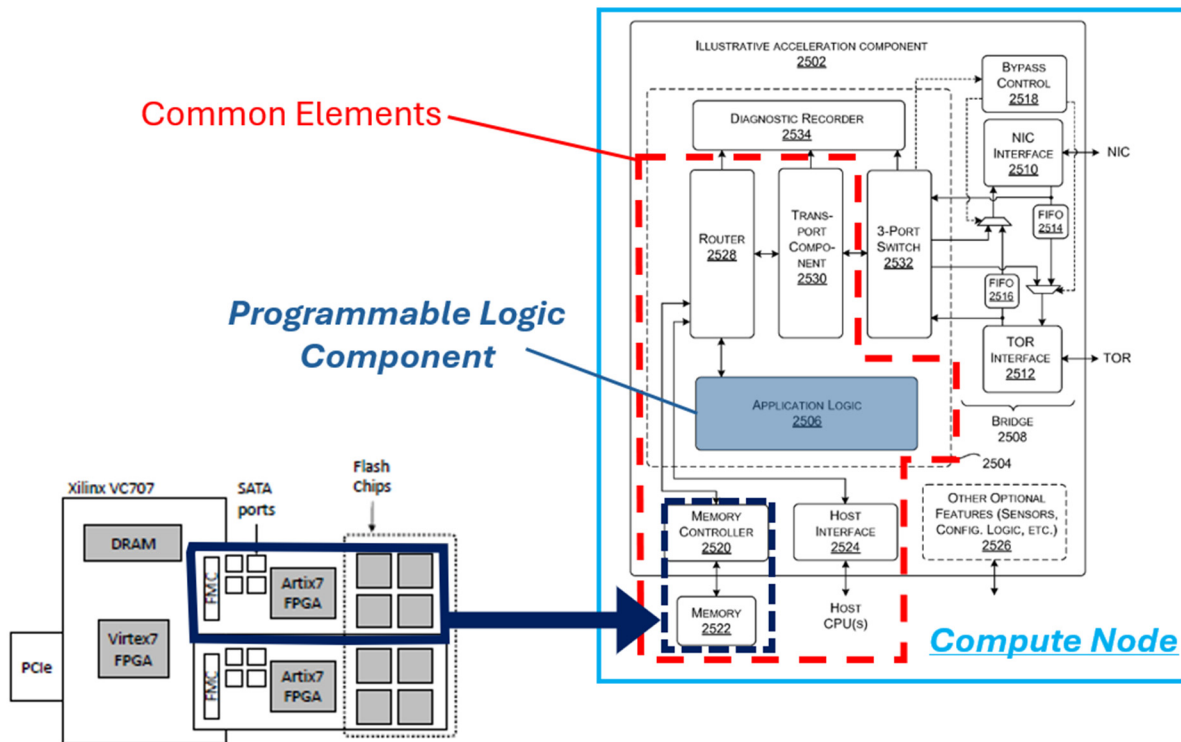
AMD1011, FIG. 8.

As discussed, HJC incorporates Jun’s Flash (*data store*) with the acceleration component in each of Heil’s server unit components (*compute nodes*). *Supra* §V.C.3. Moreover, a POSITA would have understood and found obvious to utilize Jun’s Flash for storing *records associated with a data processing operation*,

including, for example, one or more of the operations assigned for implementing the multi-component service. AMD1010, [0200]–[0205] (describing example multi-components for that “performs image recognition”), [0229] (“acceleration component performs its assigned operation”), [0218]–[0222] (“acceleration components (3606, 3608) perform more complex feature computations.”), [0223], FIGS. 32–33, 36, 40, [0069] (“data flow operations”), [0070] (“the operations that take place in the hardware acceleration plane”), [0133]–[0140], [0157], [0161]–[0162], [0171], [0178], [0198]–[0202], [0217]–[0223] (“sequence of *operations*”), [0224] (“the acceleration components (3606, 3608) that perform more advanced feature computations can also leverage parallel computational resources”), [0248]; *supra* Element_ [1pre], Element_[1b]; AMD1003, ¶¶175–177 (explaining that a POSITA would have also understood that incoming packets are stored in Jun’s Flash (as *records*) for data processing operations(including those discussed above) because Heil’s acceleration component depicted in Figure 25 includes a communication path between the router 2528 that receives the incoming data packets and the local memory 2522 (Jun’s Flash (*data store*) in the Combination) and *citing*: AMD1010, [0057], [0078], [0082], [0163]–[0164], [0167]–[0170], [0182]–[0193], [0218]–[0223], [0229], FIGS. 25, 36, 40; AMD1011; AMD1014).

Element [1e]

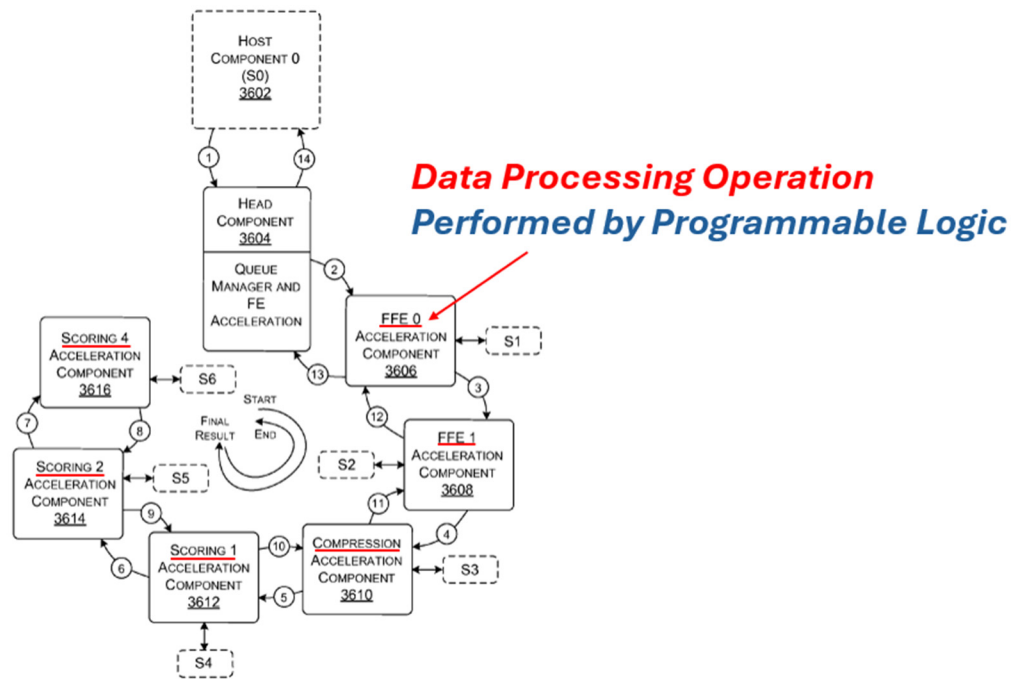
Heil's respective acceleration components within the server unit components (*compute nodes*) are implemented using an FPGA device that includes reconfigurable application logic (*programmable logic component*). AMD1010, [0050], [0139] ("reconfiguring...the application logic"), [0157]–[0158] ("reconfiguration of the application logic"), [0160]–[0162] (explaining that the application logic represents the resource most subject to change in the acceleration component), [0229], [0252], FIGS. 23–25; AMD1001, FIG. 4, 15:3–30, 37:61–67; AMD1003, ¶178. Because Heil's application logic is implemented on an FPGA device, it is a logic component programmable by reconfiguring its internal hardware connections. *Id.*



HJC Illustration.

Heil teaches that each acceleration component, via its configured application logic, performs a respective data processing operation (*executes a data processing operation*) for the multi-component service. *Supra* Element_[1pre], Element_[1b], Element_[1d]; AMD1010, [0203]–[0205], [0222] (“perform more complex feature computations.”), [0223] (“generate a final ranking score”), [0229] (“the particular acceleration component performs its assigned *operation*”), FIGS. 33, 36, 40, [0133]–[0140] (discussing “logic” on the acceleration component “for performing [a] service”), [0157] (explaining how the application logic is governed by a “model” that is implemented via a “configuration image[]” and that “corresponds to logic that performs a function in a particular manner”), [0161]–[0162], [0178] (noting

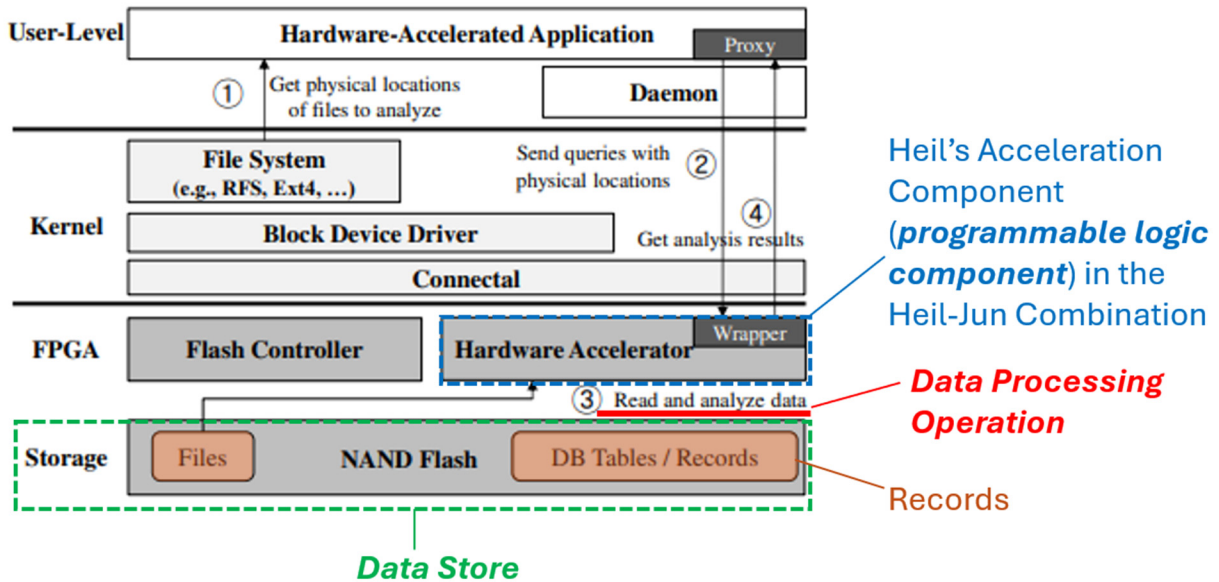
“services are implemented by the application logic”), [0198]–[0203], [0218]–[0224], [0248]; AMD1003, ¶179; *supra* §V.C.1 (explaining that the example service shown in Figure 36 can use other flow structures including the flow structure shown in Figure 33), Elements_[1a]-[1b].



AMD1010, FIG. 36.

As discussed at Element_[1d], HJC discloses storing various *records* in Jun’s Flash (*data store*) for performing the *data processing operation* assigned to the respective acceleration component of Heil’s server unit components (*compute nodes*). Accordingly, HJC provides that the application logic (*programmable logic component*) of each acceleration component operates *in cooperation with* Jun’s

Flash (*data store*) to execute its assigned *data processing operation*. AMD1003,
¶180.



AMD1011, FIG. 8.

Element_[1f]

As discussed in V.C.1 and for Elements_[1pre]–[1e], Heil's server unit components (*compute nodes*) include a local host component⁷ and local acceleration component. As detailed below, Heil's local host component uses *data and instructions* stored in a main memory (*node memory*) to configure its companion

⁷ To be clear, this Element_[1f] analysis references the “local host component” within each server unit component (*compute node*), as opposed to the requesting host component (*host compute system*) referenced at Elements_[1pre]–[1a].

local acceleration component, including the application logic (*programmable logic component*), for *supporting* the assigned *operations* and for *facilitating communications* with the respective acceleration components of neighboring *nodes*. AMD1003, ¶181.

Heil's acceleration components include an application logic (*programmable logic component*) configured according to a configuration image/stream that is "used to implement a corresponding service." AMD1010, [0155] ("Each configuration image contains logic that can be used to implement a corresponding service."), [0156]–[0157] (explaining how the application logic is governed by a "model" that is implemented via a "configuration image[]" and that "corresponds to logic that performs a function in a particular manner"), [0124] ("A configuration stream specifies the logic to be "programmed" into a recipient acceleration component."), FIGS. 21, 22; AMD1003, ¶182.

Heil also discloses that the local host component coordinates this configuration by "forwarding a configuration stream (corresponding to a configuration image) to the acceleration component." AMD1010, [0155] ("the local host component 2112 can perform the above-described [configuration-coordination] operation"). A POSITA would have understood or found obvious that a configuration stream includes data defining the logic and instructions for configuring the logic. AMD1003, ¶¶183–184 (*citing* AMD1015 pp.2, 23, AMD1027 pp.8–10,

12, 38, 47–79, 95, 118; AMD1046, 3:38–42 (“*instructions and data* (e.g., a configuration image)”)); AMD1010, [0124], [0155]–[0158], [0171]–[0172]. To transfer data to its counterpart local acceleration component, the local host component places the configuration image/stream (*data and instructions*) in the main (*node*) *memory*, so the *data and instructions* can be retrieved by the acceleration component using PCIe in conjunction with DMA (direct memory access). AMD1003, ¶183; AMD1010, [0165], [0155] (“Each configuration image contains logic that can be used to implement a corresponding service.”), [0173]–[0175], FIGS. 25, 27, *infra* Element_[1g]. The local host utilizes the conventional approach of using main memory (*node memory*) to access data and instructions for performing its described functionalities including forwarding the configuration image/stream. AMD1010, [0174], [0195] (identifying “RAM of any type(s)” and “flash devices” as “storage resources” for the host’s CPU to access “code, settings, data, etc.”), FIGS. 27, 31; AMD1003, ¶183.

Accordingly, it would have been obvious to a POSITA that each host component’s *node memory* includes *data and instructions* associated with deploying the configuration image *to support operation* of the *programmable logic component* (the acceleration component’s application logic) *in connection with the data processing operation*. AMD1003, ¶185; AMD1010, [0195], [0148]–[0152];

supra Elements_[1d]–[1e] (describing example *data processing operations* performed by configured application logic).

HJC renders obvious the *data and instructions to facilitate data communications via the communications link* at least because the acceleration component of each server unit component (*compute node*) receives “setup information” for establishing *inter node* connections. AMD1010, [0229], [0181] (“software process may set up the connections by statically allocating them before the transport component 2530 [(*communications link*)] can transmit or receive data”), [0248]. As Heil’s disclosure makes clear, this setup information received by the acceleration component is managed by a local management component implemented at the local host component within the same node. AMD1010, [0151] (“server unit component stores its own dedicated local management component...implemented by a local host component”), [0100], [0148]–[0152], FIG. 10. Like for the configuration image, Heil’s host component transfers the setup information to its counterpart local acceleration component by placing the setup information—in the main (*node*) *memory*, so it can be retrieved using PCIe in conjunction with DMA (direct memory access). AMD1003, ¶186; AMD1010, [0165], [0173]–[0175], FIG. 25, *infra* Element_[1g]. Accordingly, as would have been apparent or obvious to a POSITA, Heil’s software-implemented setup information (*data and instructions to facilitate data communications via the*

communications link) is placed into the host component's main (*node*) *memory* at least for facilitating transfer to the acceleration component. *Id.*; AMD1010, [0181], [0229], [0248]; AMD1003, ¶186. A POSITA would have also understood that implementing the local management component at the host component includes using *data and instructions* stored at the host component's main (*node*) *memory*. AMD1010, [0174], [0195], FIGS. 27, 31; AMD1003, ¶186.

HJC also renders *facilitating data communications* obvious via the *data and instructions* associated with the configuration image in Heil's local host main memory (*node memory*). For example, Heil's acceleration component processes and flow-regulates data for transfer to the node in the next tier via the transport component (*communications link*). AMD1003, ¶187; AMD1010, [0167]–[0168], [0191] (“the application logic 2506...regulate[s] the flow of packets through the use of ‘back pressuring’”), [0203], [0218]–[0223], [0229], FIGS. 25, 33, 36, 40. This data communication would not have occurred effectively, unless each acceleration unit is appropriately configured. *Id.*

Element [1g]

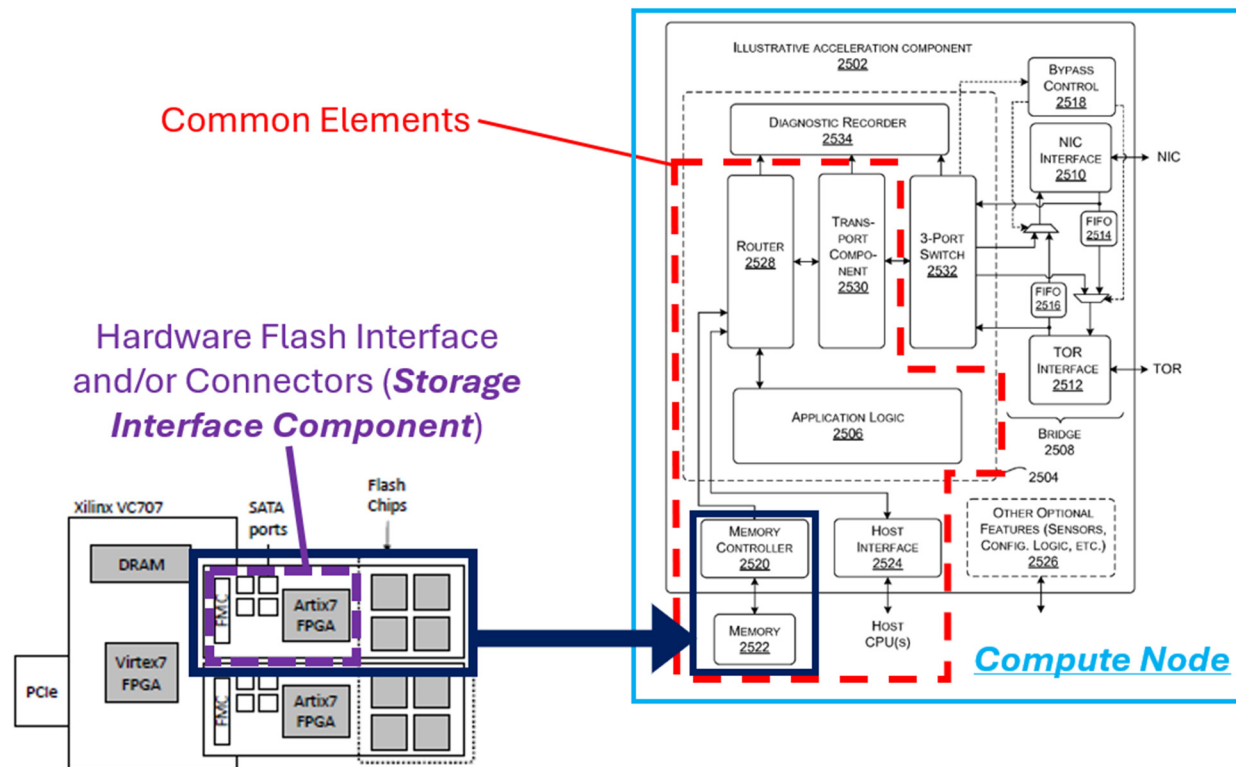
As discussed at Element [1f], each of Heil's server unit components (*compute nodes*) includes a main memory (*node memory*) for its local host component. Heil further discloses that each local host component may communicate with its local acceleration component “using PCIe in conjunction with DMA [(direct

memory access)] transfer.” AMD1010, [0173]–[0175], [0089] (“Like the data processing system 102 of FIG. 1, the data processing system 802 includes...a local_H-to-locals coupling 114 for connecting local host components to respective local acceleration components.”) [0165], FIG. 27[0165]. The components used for PCIe/DMA memory transfer operate as a *data mover component to facilitate* data transfer within the server unit component (*data communications*) *between* the local host main memory (*node memory*) and the acceleration component’s application logic (*programmable logic component*). AMD1010, [0173]–[0175] (explaining PCIe/DMA communication from buffers of the local host main memory to the application logic of the local acceleration component), [0085], [0089], [0124], [0155]–[0156], FIGS. 8, 22, 25, 27; *supra* Element_[1f]; *infra* Elements_[6]–[7]; AMD1003, ¶188–190 (citing AMD1014); AMD1001, 15:3-30 (explaining that the data mover is configured to facilitate data communications).

Element_[1h]

As discussed, HJC incorporates Jun’s Flash (*data store*) to serve as the local memory for Heil’s acceleration component. *Supra* §§V.C.2–3; Element_[1d]; AMD1011, pp.4–5, 8, FIGS. 2–4, 10. The Flash is connected to a processor (*e.g.*, “Virtex 7 FPGA”) using a “FPGA Mezzanine Card (FMC) connector.” AMD1011, p.8. “The *communication between* the flash board and the Virtex 7 FPGA [processor] is done by a 4-lane aurora channel” and “SATA connectors.” *Id.*

Moreover, Jun provides that “[t]he flash controller and Error Correcting Code (ECC) is implemented on [an] Artix chip” that “exposes a low-level, thin, fast and bit-error corrected hardware *interface*” to the Flash. AMD1011, pp.8, 4 (explaining the benefits of the hardware interface, including “cutting down on latency,” “exposing all degrees of parallelism,” and “allowing higher level system stacks”), FIGS. 2–4, 10; *infra* Element_[1i] (demonstrating the multiple channels to NAND Flash Chips). Individually or collectively, the hardware interface and connectors on Jun’s flash board function as a *storage interface component*. AMD1003, ¶191.



HJC Illustration.

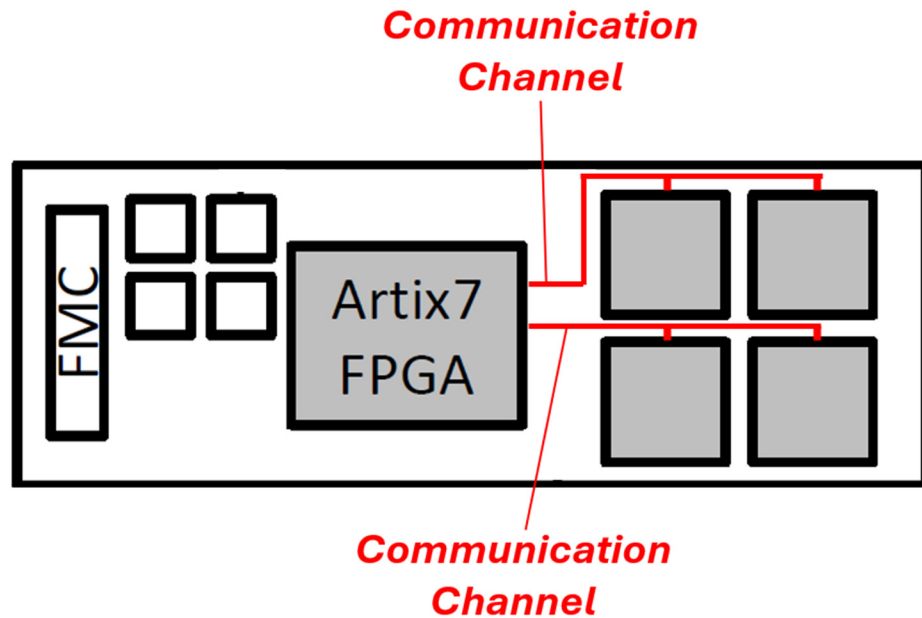
In the HJC, Heil’s acceleration components incorporate Jun’s flash board, which a POSITA would have understood included a *storage interface component to facilitate data communication between* Jun’s Flash (*data store*) and the application logic (*programmable logic component*). *Supra* §V.C.3, Elements_[1d]–[1e]; AMD1003, ¶192; AMD1001, 10:32–34 (“data store interface 145 manages access...to data store 143, as is generally known in the art.”), 15:31–16:60, 39:10–20.

Element_[1i]

As discussed, HJC incorporates Jun’s teachings of a flash board including Flash (*data store*) in conjunction with suitable connectors and hardware interface (*storage interface component*). *Supra* §§V.C.2–3, Elements_[1d]–[1e], Element_[1h]; AMD1003, ¶193.

Jun discloses and a POSITA would have found obvious that the flash board’s Artix-7 FPGA manages data transfer from the Flash utilizing *multiple parallel communication channels* to access NAND Flash Chips. AMD1011, FIG. 10 (below), pp.4 (“Flash devices...achieve high bandwidth by grouping multiple flash chips into several *channels*...”), 6 (“data reads from flash chips on multiple buses in parallel”) 9; AMD1003, ¶¶194-195 (*citing* AMD1012). In HJC, Jun’s flash interface provides high-bandwidth Flash access to the application logic. *Id.*

Jun's Flash
Board →



AMD1011, FIG. 10 (modified).

HJC also renders Element_[1i] obvious via Jun's disclosure that "[t]he communication between the flash board and the Virtex 7 FPGA processor is done by a 4-lane aurora *channel*" and "SATA connectors." AMD1011, p.8, FIGS. 2 ("Flash *interface*"), 10. In HJC, Jun's 4-lane aurora *channel* and/or SATA connectors are *communication channels to transfer data* from Flash (*data store*) to application logic (*programmable logic component*) of Heil's acceleration component. AMD1003, ¶196; *supra* §§V.C.2–3.

Element_[2]

As discussed at Elements_[1b]–[1c], Heil's server unit components (*compute nodes*) include an acceleration component with a transport component

(*communication link*) for communicating with acceleration components of neighboring server unit components in the successive tier. This inter-node communication takes place via “cable links.” AMD1010, [0089]. It would have been obvious to a POSITA that when HJC’s nodes are communicatively coupled in series, the acceleration components communicating via “cable links” in a distributed computing environment routinely would have employed *serial communication protocols* to support serial communication and high-speed data transfer. AMD1010, [0184] (describing an example employing a serial protocol (Ethernet)); AMD1003, ¶¶197–198 (*citing* AMD1047, p.310 (“Ethernet is a serial protocol”)); AMD1014, 4 (showing Heil’s torus network example with “[f]our high-speed *serial* links running *SerialLite III* (SL3), a lightweight *protocol* for communicating with neighboring FPGAs”); AMD1011, p.4 (“The FPGAs form a network...via high-speed serial links...”), 9.

HJC also renders Element_[2] obvious by virtue of Jun’s teachings for using “high-performance *serial* links” to form a network among FPGA-based devices. AMD1011, pp.5, 8 (“the link is implemented over the low-latency *serial* transceivers”), 9, FIGS. 5, 9 (Serial Advanced Technology Attachment (“SATA”) cables); AMD1003, ¶¶199–200. A POSITA would have understood that a serial protocol would be employed in a system using high-performance *serial* links with *serial* transceivers. AMD1003, ¶¶199–200 (explaining that a *serial protocol* (e.g.,

Serial ATA protocols) would have been used to transmit data over SATA cables and *citing* AMD1033). Appreciating the known low-latency benefits of high-speed data transfer, a POSITA would have been motivated to employ links using a ***serial communication protocol***, as Jun did, for connecting the acceleration components in the Heil’s server unit components (***compute nodes***). *Id.*

Element_[3]

As discussed, HJC incorporates Jun’s Flash (***data store***) for each server unit component (***compute node***). *Supra* §§V.C.2–3, Elements_[1d], [1i]–[1h]. It was known in the art and a POSITA would have recognized that a Flash is an ***EEPROM device***. AMD1001, 10:11–34 (“all Flash is EEPROM”); AMD1003, ¶201 (*citing* AMD1034).

Element_[5]

Supra Element_[1e]; AMD1003, ¶¶202–203 (*citing* AMD1014).

Element_[6]

As discussed at Element_[1g], each of Heil’s server unit components (***compute nodes***) utilizes PCIe/DMA elements (***data mover component***) to move data from the main memory of the host component to the acceleration component. AMD1010, [0165]. Heil further explains that each “[local] host component 2702 may forward information to its local acceleration component 2704” using host logic. AMD1010, [0173]–[0175], FIG. 27. The “host logic 2706 places data to be

processed into a kernel-pinned input buffer 2708 in main memory associated with the host logic 2706” and “instructs the acceleration component 2704 to retrieve the data.” *Id.* Heil’s host logic is implemented on a **processor** that instructs (*e.g., manages operation*) of **data mover component** (components for PCIe and DMA). *Id.*; [0195] (“host component 3102 can include one or more processing devices 3104...which may implement one or more hardware threads.”), [0077], [0048], [0051], [0054], [0077], [0133], [0248]; AMD1003, ¶¶204-205 (*citing* AMD1014).

Element [7]

As discussed at Elements [1g], [6], Heil’s **data mover component** at each server unit component (**compute node**) utilizes PCIe/DMA elements (**data mover**) to facilitate communications between a local host and a local acceleration component. As would have been obvious to a POSITA, PCIe/DMA functionality like Heil describes routinely involves a **device controller** to help manage the transfer process. AMD1003, ¶206 (*citing* AMD1014, FIG. 3 (“DMA Engine”); AMD1039, pp.3–5, FIGS. 3–5 (demonstrating a “**device controller**” for PCIe DMA); AMD1010, [0165], [0173]–[0175], FIG. 27. For example, the POSITA would have appreciated that a DMA **controller** would have been used in Heil’s system to receive instructions and notify the host logic for receiving data from and providing data to the host component’s main memory. *Id.*

HJC also renders Element_[7] obvious via Jun’s teachings for a DMA Engine (*device controller*) as part of a PCIe-based host interface, and it would have been obvious to integrate this teaching with Heil in order facilitate data transfer from the local host’s main memory using PCIe. AMD1011, pp.4–6 (“local host software over PCIe DMA”), 6 (“the DMA engine on the hardware will read data”), 9 (explaining use of “DMA read/write engines.”), FIGS. 7, 9–10; *cf.* AMD1001, 15:3–30; AMD1003, ¶207.

Element_[8]

As discussed at Elements_[1pre]–[1a], Heil’s requesting host component (*host compute system*) invokes the multi-component service by sending a request to the head component for the multi-component service. In one example, this request includes “various items of information, such as...the locations of the query terms within the document” and “a collection of software-generated feature values.” AMD1010, [0220]. Thus, the information included in the request *influences the operation of* each of *the* server unit components (*compute nodes*). AMD1010, [0198]–[0203], [0218]–[0220], [0229], FIGS. 33, 36, 40. The requesting host component generates and sends this request by *executing instructions*. AMD1010, [0079] (“a program running on the host component 412 requests a service”), [0195], [0198], [0203], [0218]–[0223], FIGS. 31, 33, 36; AMD1003, ¶208.

D. GROUND 2B: Heil-Jun-Raghava Renders Obvious Claims 4, 9

1. Heil-Jun-Raghava Combination

Heil, Jun, and Raghava qualify as analogous art to the '750 patent because they share the '750 patent's field of endeavor in data processing systems employing reconfigurable hardware. AMD1003, ¶209; AMD1006, Abstract, [0004], [0015]–[0017], [0040]–[0049]; *supra* §§V.B.1, V.C.1-3. They are also reasonably pertinent to a technical problem shared with the '750 patent of accelerating processing by describing solutions that offload tasks to reconfigurable hardware. *Id.*

(a) Raghava's ONFI

As discussed, HJC incorporates Jun's teachings regarding a flash board including NAND-based Flash in conjunction with suitable connectors and hardware interface (*storage interface*) for the server unit components (*compute nodes*). *Supra* §§V.C.3; V.C.4, Elements_[1d], [1h]–[1i]. A POSITA would have been motivated to implement a standard protocol like *ONFI*, as taught by Raghava, at *each* server unit component (*compute node*). AMD1003, ¶¶210–211 (*citing* AMD1012 showing a BlueDBM Node (the same as Jun's) with an “*ONFI*-compliant FPGA-based flash controller.”); *supra* §§V.B.1–2, V.B.3, Elements_[4]–[5] (describing Raghava's teachings for the ONFI protocol with NAND Flash); AMD1006, [0033], [0046]. A POSITA would have been motivated to employ Raghava's *ONFI protocol* to

support communication between Jun's NAND Flash and Heil's acceleration component and achieve known benefits including improved transfer rate, power consumption, and/or standardized interoperability. AMD1003, ¶¶212–213 (*citing* AMD1035, AMD1012).

Additionally, the combination of HJC's server unit components (*compute node*) including NAND Flash with Raghava's *ONFI* teachings for accessing the same type of storage interface would have been obvious because it represents the predictable application of a known protocol to achieve the expected results for communicating with NAND Flash. AMD1003, ¶214 (*citing* AMD1012). A POSITA would have an expectation of success for implementing a standard *protocol* like Raghava's *ONFI* on the known components (Jun's *storage interface*) to achieve an expected result of enabling access to NAND Flash at least because ONFI is a standard protocol that is specifically designed for interoperability with different hardware components to facilitate communication and in particular with functionally compatible devices like Jun's storage interface and NAND Flash. *Id.*

(b) Raghava's Interface Card

Heil discloses a host component and multiple remote acceleration components "implemented on a single physical component." AMD1010, [0084]–[0090] ("implemented on a single die"). Consistent with this disclosure, a POSITA would have implemented a single component with one or more of Heil's server unit

components (*compute nodes*) (e.g., one or more of the server unit components represented by the acceleration components forming the parallel paths in Figure 33) and remote head/acceleration component (*router module*) (e.g., head component 3304 in Figure 33). AMD1003, ¶215. Indeed, Raghava demonstrates that integrating this arrangement into a single accelerator DIMM card (*interface card*) was well known. AMD1006, [0040]–[0045], FIG. 5; *supra* §§V.B.1, V.C.4, Element_[1a]. Specifically, Raghava’s *interface card* is coupled to a host CPU and uses a memory controller and request decoding/routing circuit to route host requests to one or more FPGA-based hardware acceleration units. AMD1006, [0040]–[0045], FIG. 5; AMD1003, ¶215.

It would have been obvious to integrate Heil’s head component (*router module*) and one or more server unit components (*first node*) in an *interface card communicatively coupled to the host compute system*, based on Heil’s teaching to incorporate multiple remote acceleration components (such as the head component/*router module* and one or more server unit components (*compute nodes*)) on a single component with Raghava’s teachings for a router and one or more acceleration unit arrangement on an interface card coupled to the host. AMD1003, ¶216. The following reasons, taken separately or collectively, support such a Heil-Jun-Raghava Combination.

First, a POSITA would have been motivated based on Raghava's teachings that the accelerator DIMM *card*, by including both *router module* and one or more *compute nodes*, further supports Heil's goals to orchestrate multi-component services by offloading computationally intensive tasks to parallel computational resources. AMD1003, ¶217; AMD1006, Abstract, [0004] ("offloading computationally intensive tasks...to highly-parallel acceleration resources on a programmable device"), [0015] ("Offloading computationally intensive functions onto the accelerator DIMM card is oftentimes more efficient than performing them natively on the host CPU due to the highly parallel nature of the programmable device."); [0017], [0040]–[0044], [0048]–[0049]; AMD1010, [0051] (explaining that an acceleration component can perform "operations in less time"), [0224] ("leverage parallel computational resources").

Second, a POSITA would have understood that integrating Heil's head component (*router module*) and server unit components (*compute nodes*) within a single component as Raghava's accelerator DIMM (*interface*) *card* would have reduced latency in Heil's data processing system. AMD1003, ¶218. For example, as taught by Raghava, such an arrangement would have allowed the host component to initiate a service and execute Heil's service with less latency. *Id.*; AMD1006, [0036]–[0037], [0054]–[0055].

Third, a POSITA would have been motivated to implement Raghava's interface card based on Raghava's teachings that "the inclusion of acceleration circuitry in the DIMM card/slot improves the energy efficiency of applications." AMD1006, [0053]; AMD1003, ¶219.

Fourth, combining HJC's head component (*router module*) and server unit components (*compute nodes*), each having (FPGA-based) acceleration, with Raghava's accelerator DIMM cards also having FPGA-based acceleration would have been obvious because it represents the predictable application of a known multi-stage architecture to a high-performance interface to achieve the expected results of reduced latency for offloading complex processing to specialized/FPGA-based resources. AMD1003, ¶220. A POSITA would have an expectation of success for implementing known components (e.g., Heil's head component (*router module*) and server unit components (*compute node*) in a known arrangement on a known interface card as taught by Raghava to execute complex processing operations at FPGA-based acceleration components. AMD1003, ¶220 (*citing* AMD1014).

2. Analysis

Element [4]

The Heil-Jun-Raghava combination discussed in §V.D.1.(a) satisfies Element [4]. AMD1003, ¶221, *supra* §V.B.1, V.B.3, Elements [3]–[4].

Element [9]

The Heil-Jun-Raghava combination discussed in §V.D.1.(b) satisfies Element_[9]. AMD1003, ¶222; *supra* §V.D.1.(b).

VI. PAYMENT OF FEES

Petitioner authorizes the Patent and Trademark Office to charge any fees to Deposit Account No. 06-1050.

VII. MANDATORY NOTICES—37 C.F.R. § 42.8(a)(1)

A. Real Party-In-Interest—37 C.F.R. § 42.8(b)(1)

Petitioner, Advanced Micro Devices, Inc. is the real party-in-interest.

B. Related Matters—37 C.F.R. § 42.8(b)(2)

Petitioner is not aware of any disclaimers, reexamination certificates or petitions for inter partes review for the '750 patent. The '750 patent is the subject of civil action: *Fermat International, Inc. v. Advanced Micro Devices*, 7:26-cv-00011 (WDTX), filed January 13, 2026.

C. Lead And Backup Counsel—37 C.F.R. § 42.8(b)(3)

Petitioner provides the following designation of counsel.

Lead Counsel	Backup counsel
Kenneth W. Darby, Jr., Reg. No. 65,068 Fish & Richardson P.C. 60 South Sixth Street, Suite 3200 Minneapolis, MN 55402 Tel: 202-783-5070 Fax: 877-769-7945 Email: IPR54788-0024IP1@fr.com	David Hoffman, Reg. No. 54,174 Usman A. Khan, Reg. No. 70,439 Cameron A. Ubel, Reg. No. 77,081 Joseph W. Bauer, Reg. No. 81,218 Fish & Richardson P.C. 60 South Sixth Street, Suite 3200 Minneapolis, MN 55402 Tel: 202-783-5070 / Fax: 877-769-7945 PTABInbound@fr.com

D. Service Information

Please address all correspondence and service to the address listed above.

Petitioner consents to electronic service by email at IPR54788-0024IP1@fr.com
(referencing No. 54788-0024IP1).

Respectfully submitted,

Dated May 12, 2026

/Usman Khan/
Kenneth W. Darby, Jr., Reg. No. 65,068
David Hoffman, Reg. No. 54,174
Usman A. Khan, Reg. No. 70,439
Cameron A. Ubel, Reg. No. 77,081
Joseph W. Bauer, Reg. No. 81,218
Fish & Richardson P.C.
60 South Sixth Street, Suite 3200
Minneapolis, MN 55402
T: 202-783-5070
F: 877-769-7945

(Control No. IPR2026-00358)

Attorneys for Petitioner

CERTIFICATION UNDER 37 CFR § 42.24

Under the provisions of 37 CFR § 42.24(d), the undersigned hereby certifies that the word count for the foregoing Petition for *Inter partes* Review totals 13,919 words, which is less than the 14,000 allowed under 37 CFR § 42.24.

Dated May 12, 2026

/Usman Khan/

Kenneth W. Darby, Jr., Reg. No. 65,068

David Hoffman, Reg. No. 54,174

Usman A. Khan, Reg. No. 70,439

Cameron A. Ubel, Reg. No. 77,081

Joseph W. Bauer, Reg. No. 81,218

Fish & Richardson P.C.

60 South Sixth Street, Suite 3200

Minneapolis, MN 55402

T: 202-783-5070

F: 877-769-7945

Attorneys for Petitioner

CERTIFICATE OF SERVICE

Pursuant to 37 CFR §§ 42.6(e)(4)(i) *et seq.* and 42.105(b), the undersigned certifies that on May 12, 2026, a complete and entire copy of this Petition for *Inter partes* Review, Power of Attorney, and all supporting exhibits were provided by Federal Express, to the Patent Owner by serving the correspondence address of record as follows:

Ethos Legal P.C.
30 Gruver Road
Pipersville, PA 18947

/Diana Bradley/
Diana Bradley
Fish & Richardson P.C.
60 South Sixth Street, Suite 3200
Minneapolis, MN 55402
bradley@fr.com