

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

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

Petitioners,

v.

CONCURRENT VENTURES, LLC AND XTREAMEDGE, INC.,

Patent Owners.

Case No. IPR2025-00223

U.S. Patent 10,985,943

Declaration of Donald Alpert, Ph.D.

Table of Contents

I.	OVERVIEW OF THE TECHNOLOGY.....	5
A.	Programmable Logic Devices (PLD).....	5
B.	Scalable Computing Systems.....	11
II.	THE '943 PATENT.....	15
A.	Prosecution History.....	17
B.	Priority Date.....	19
C.	Challenged Claims.....	20
D.	Patent Owner's District Court Allegations.....	20
III.	LEVEL OF ORDINARY SKILL IN THE ART.....	25
IV.	CLAIM CONSTRUCTION.....	27
A.	“programmable logic device”.....	28
V.	INVALIDITY GROUNDS.....	31
A.	Grounds 1-2: the Challenged Claims are Obvious based on Seshadri alone (Ground 1) and Seshadri in view of Biederman (Ground 2).	31
1.	U.S. Patent No. 10,776,142 (“Seshadri”) (Ex-1004).....	32
2.	U.S. Patent Application Publication No. 2018/0191642A1 (“Biederman”) (Ex-1005)	35
3.	Independent Claim 1	38
4.	Independent Claim 10.....	58
5.	Dependent Claim 2	70
6.	Dependent Claim 11	74
7.	Dependent Claim 3	75

8.	Dependent Claim 12	77
9.	Dependent Claims 4 and 13	78
10.	Dependent Claim 5	84
11.	Dependent Claim 14	86
12.	Dependent Claims 9 and 18	86
B.	Grounds 3-4: the Challenged Claims are Obvious based on Atsatt-113 alone (Ground 3) and Atsatt-113 in view of Biederman (Ground 4).	93
1.	U.S. Patent Application Publication No. 2019/0095113A1 (“Atsatt-113”) (Ex-1006).....	94
2.	Independent Claim 1	97
3.	Independent Claim 10	126
4.	Dependent Claim 2	135
5.	Dependent Claim 11	139
6.	Dependent Claims 3 and 12	140
7.	Dependent Claims 4 and 13	143
8.	Dependent Claims 5 and 14	145
9.	Dependent Claims 9 and 18	149
C.	Ground 5	152
1.	Tarafdar et al., “Enabling Flexible Network FPGA Clusters in a Heterogeneous Cloud Data Center” (“Tarafdar”) (Ex-1007)	152
2.	Dependent Claims 6 and 15 are Rendered Obvious in View of Tarafdar in combination with Grounds 1-4.	155
D.	Ground 6	167
1.	U.S. Patent No. 8,015,386 (“Kulkarni”) (Ex-1008)	168

2.	Dependent Claims 7 and 16 are Rendered Obvious in View of Kulkarni in combination with Grounds 1-4.....	170
E.	Ground 7.....	182
1.	U.S. Patent Application Publication No. 2019/0041923A1 (“Atsatt-923”) (Ex-1009).....	182
2.	Dependent Claims 8 and 17 are Rendered Obvious in View of Atsatt-923 in combination with Grounds 1-4.	184
VI.	BACKGROUND AND QUALIFICATIONS.....	191
A.	Compensation.....	193
B.	Materials and Other Information Considered	193
VII.	UNDERSTANDING OF THE LAW	194
VIII.	RESERVATION OF RIGHTS.....	194
	APPENDIX 1: CURRICULUM VITAE OF DONALD ALPERT	196
	APPENDIX 2: MATERIALS CONSIDERED IN THE PREPARATION OF THIS DECLARATION.....	202
	APPENDIX 3: CHALLENGED CLAIMS	204
	APPENDIX 4: UNDERSTANDING OF THE LAW	212

I, Donald Alpert declare as follows:

1. My name is Donald Alpert. I am an independent consultant with Camelback Computer Architecture, LLC. I have prepared this report as an expert witness retained by Advanced Micro Devices, Inc. and Pensando Systems, Inc. In this report I give my opinions as to whether certain claims of U.S. Patent No. 10,985,943 (“the ’943 patent”) are invalid. I provide technical bases for these opinions as appropriate.

2. This report contains statements of my opinions formed to date and the bases and reasons for those opinions. I may offer additional opinions based on further review of materials in this case, including opinions and/or testimony of other expert witnesses. I make this declaration based upon my own personal knowledge and, if called upon to testify, would testify competently to the matters contained herein.

I. OVERVIEW OF THE TECHNOLOGY

A. Programmable Logic Devices (PLD)

3. In contrast to microprocessors with logic circuits that are fixed when the integrated circuit (“IC”) is manufactured, PLDs are a class of IC devices that provide logic circuit area that can be configured after manufacturing, typically by a user. PLDs typically presented a large number of logic gates where the user configured how the gates operated and how they were connected to form circuits.

Because these circuits are based on logic gates that provide binary outputs, they were typically connected in digital circuits to perform digital logic functions.

4. PLDs have been widely used in the art for decades. This includes a variety of technologies and device types including programmable array logic (PAL), programmable logic arrays (PLA), complex programmable logic devices (CPLD), and Field Programmable Gate Arrays (FPGAs) among others. PLAs were common by the 1980s, and FPGAs were common by the 1990s. By the 2000s, it was common for undergraduate EE courses to teach FPGA programming and related subject matter was covered in textbooks for courses such as digital design and computer architecture.

5. In particular, FPGAs were a popular solution to develop logic circuits due to their reprogrammable nature. Ex-1010, Abstract. This allowed for lower non-recurring engineering costs, as well as a faster turn-around, compared to ASICs or other custom ICs that required the production of expensive mask sets for production. Ex-1010, 000002-3. As the name suggests, FPGAs could be easily reprogrammed in the field, which opened up the possibility of field updates after deployment to correct bugs or to provide new functions and increased performance.

6. FPGAs contain programmable logic blocks and programmable interconnects. Ex-1010, 000004. For several decades, the configuration of these

components were typically loaded as configuration data into volatile memory (SRAM) on the FPGA on startup. Ex-1010, 000009-11; Ex-1008, 1:30-37. The configuration data implemented logic functions by specifying which of the logic functions and connections were to be activated (and deactivated) in the FPGA. *Id.* In this way, FPGAs allowed users to specify the logical interconnections of circuits within the FPGA and create custom hardware to perform custom functions.

7. The following figure illustrates the general organizational principle of FPGAs. Ex-1011, Fig. 4:

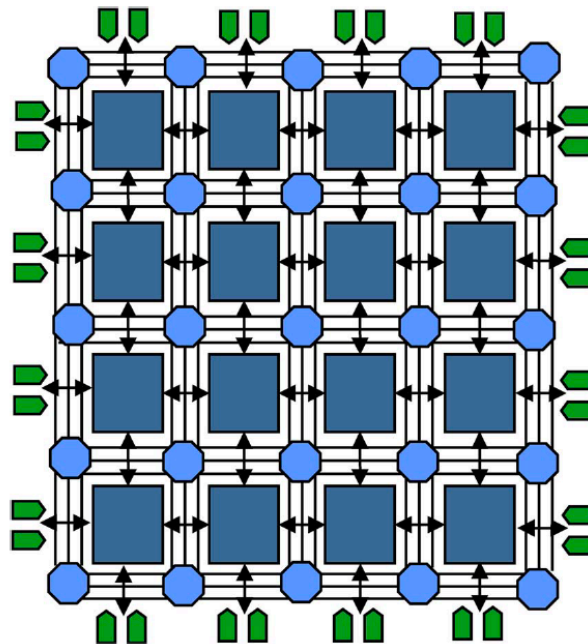


Fig. 4. Generic array FPGA architecture. 4×4 array with three wiring tracks per row and column. Switches are at the circles at intersections. Device inputs and outputs are distributed around the array.

PLDs, including FPGAs, generally included two regions: one (in blue below) with programmable logic blocks and another (in red) for control functions and input/output. *E.g.*, Ex-1004, 10:36-47, Fig. 2; Ex-1006, Fig. 9:

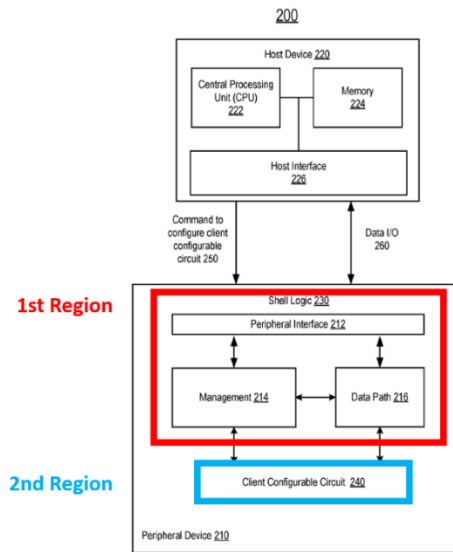


FIG. 2

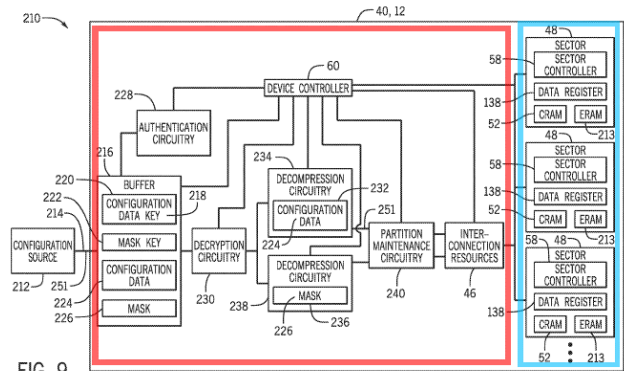


FIG. 9

8. The figure shows an example of how programmable logic blocks in an FPGA were wired by programmable interconnects. The interconnects function as a network to relay signals between programmable logic blocks, as well as to I/O circuits to interface with external circuitry or devices. For example, since FPGAs were often sold as cards that were plugged into PCIe slots, FPGA ICs typically included I/O circuits with a PCIe interface, and the interconnection network within the FPGA connected the programmable logic blocks with the PCIe interface. Below are examples of FPGA cards that are plugged into a PCIe slot. Ex-1027, Fig. 3 at 000006:



Ex-1028, 000001:



9. FPGAs could be programmed in whole or in part. When entirely reprogrammed, the configuration data was simply written or rewritten as explained above. Another technique used for programming FPGAs was partial reconfiguration which let users “change the configuration of part of [a] FPGA dynamically without disturbing other parts.” Ex-1020, §1 at 000001. This feature was commonly used to divide the programmable fabric into a number of partitions and independently manage them. Ex-1020, §2.1 at 000001-2. For example, the logic sectors could be distributed in various ways. Ex-1020, Fig. 1:

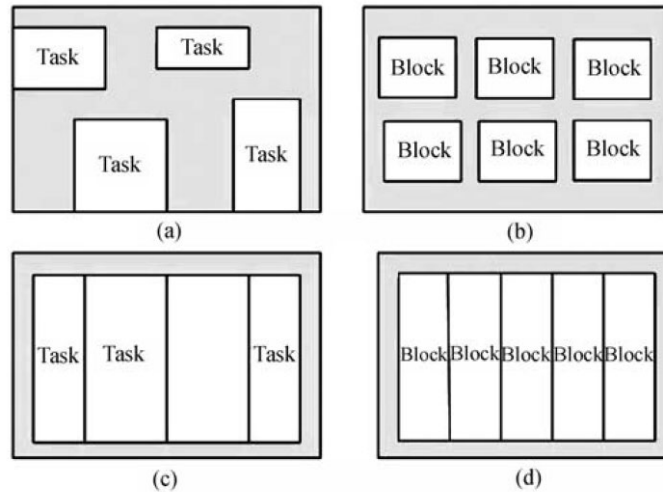


Fig.1. Reconfigurable area models. (a) Flexible 2D. (b) Partitioned 2D. (c) Flexible 1D. (d) Partitioned 1D.

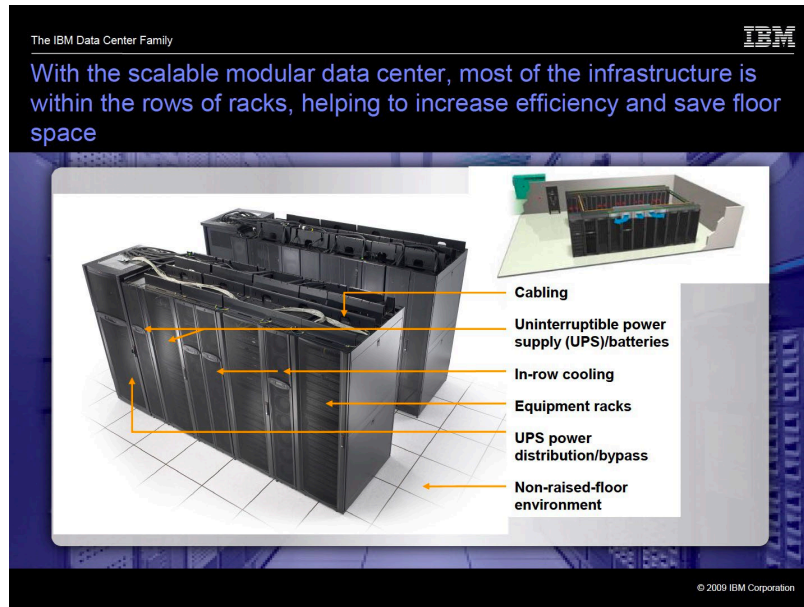
10. Partitioning was particularly useful for server environments because it allowed multi-tenancy, that is, for “sharing a single physical device among multiple users.” Ex-1025, §2 at 000004. Partitions were assigned to different users, who each loaded their own logic design into their designated partitions. Ex-1026, §II.C at 000003; *e.g.*, Ex-1007, §2 at 000003, Ex-1024, §III at 000003-4, Fig. 4, §II.A at 000002; Ex-1023, §3.2 at 000004, Fig. 2 at 000005, Fig. 1 at 000003. It was also known to dynamically resize partitions. *See, e.g.*, Ex-1020, §4.1-4 at 000005-8.

11. In addition to the reprogrammable fabric, PLDs such as FPGAs also included various fixed function circuits for commonly used functions. These often included communication ports to allow communication using specific protocols or standards, such as PCIe. Ex-1004, 10:48-65; Ex-1006, ¶30.

12. Because PLDs, including FPGAs, are reprogrammable and often used in networked applications, such as data centers for online servers or cloud computing that I discuss below, they are subject to a number of security risks. Ex-1012, 00100. One such risk was that of logic designs being stolen, copied, or reverse-engineered, from the chip itself or during transmission for reconfiguration. Ex-1013, 000001-2. Another risk was that of operating logic being corrupted. Ex-1013, 000001. Therefore, FPGAs included various techniques for increasing security. Ex-1013, 000002-8; Ex-1012, 000101. One well-known approach for securing configuration data was by locking access to the configuration data and only unlocking access if the correct key was provided. *E.g.*, Ex-1006, ¶¶61-62, Fig. 8. Keys were also used to lock and unlock access to the PLD or parts of it. Ex-1006, ¶¶59-62, ¶¶66-69, ¶¶41-42.

B. Scalable Computing Systems

13. The exponential growth of the Internet since the turn of the millennium resulted in ever-growing demand for server infrastructure. *See, e.g.*, Ex-1014, 00009. Cloud computing and storage emerged as a mainstream service in the 2000s and has further exacerbated this need. Associated with the rise of this on-demand computing paradigm was increased performance and storage requirements for data center systems. The following illustration shows an example configuration of equipment in server rooms:



Ex-1014, 00015.

14. Various solutions were developed to maximize computing power as a function of resources, in terms of space, energy, maintenance, and cost. *See, e.g.,* Ex-1014, 00014. The most common approach, going back to the 1990s, was, and still is, to utilize modular platforms that concentrate large numbers of modules (e.g., cards/blades/boards/pods) in an enclosure (e.g., racks/chasses/cabinets) based on component and system standards. *E.g.,* Ex-1015, 000002, Fig. 1:

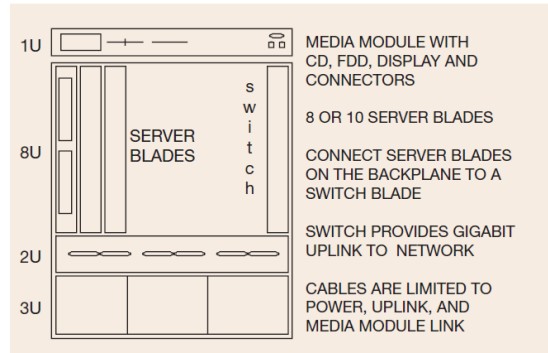


Figure 1

The earliest concept drawing of the BladeCenter architecture, created on October 20, 1999.

15. One concern arising from physically concentrating a large number of processors in small spaces was keeping temperatures within operational bounds. *See, e.g.,* Ex-1014, 00022. Operating electronics at higher speeds releases more heat. Temperature affected performance because the characteristics (e.g., resistance, etc.) of semiconductor materials change with temperature. Heat could also affect the integrity of bonding, sealing, insulating, or packaging materials. Integrated circuits could malfunction, or at the extremes, be permanently damaged. Various techniques were known and used to manage temperature. External solutions included active or passive cooling using heatsinks, fans, or air conditioning. *E.g.,* Ex-1014, 00023. Commonly employed intra-chip solutions included adjusting the operating frequency and voltage at which the IC operates. *E.g.,* Ex-1009, Abstract.

16. For example, IBM's BladeCenter architecture provided rows of racks in a chassis. The racks were loaded with multiple blades, each blade being

“general- or special-purpose servers that contain processors, memory, optional local...disk drives, Gigabit Ethernet controllers....” Ex-1016, 000001. A mid-plane board provided I/O connections between the blades, allowing the servers to work in concert and provided a shared connection to the wider network. *See* Ex-1016, 000003-4 (discussing “I/O fabric”). Various processing elements were used on such blades, including general purpose central-processing units (CPU), application-specific integrated circuits (ASIC), and programmable logic devices (PLD), in addition to memory and storage components.

17. Due to their lower power consumption, lower heat profile, high performance, and flexibility (e.g., reprogrammability), PLDs such as field-programmable gate arrays (FPGAs) became popular for use as processing accelerators in data center blades. Ex-1017, §2.5.10 at 000087; Ex-1007, §1 at 000001. For example, “[t]he FPGA High Performance Computing Alliance (FHPCA)[1] in Scotland built a 64-FPGA parallel computer system called ‘Maxwell’” consisting of “32 computational blades contained within five IBM BladeCenter[7] chassis. Each double-width blade...holds two PCI-X FPGA accelerator cards. Half of the blades hold two Nallatech H101[8] cards each containing a Xilinx[9, 10] Virtex-4-160 FPGA....” *E.g.*, Ex-1018, §1, §3 at 000001; Ex-1019, 000001:

Products in the H100 Series

PCI-X Mainstream



H101-PCIXM

- » HPC solution for optimal price/performance
- » PCI-X form factor
- » Single Xilinx® Virtex 4 FPGA processor
- » SRAM and SDRAM external memory
- » Four high-speed serial I/O system links
- » Supported by Nallatech's application development flow and ANSI C-to-FPGA compiler
- » Ideal for adding FPGA compute capability to new or existing clusters

6.4
DOUBLE PRECISION
GFLOPS SUSTAINED

20
SINGLE PRECISION
GFLOPS SUSTAINED

64
INT16 GigaOPS
SUSTAINED

Accelerators were often used in situations that involved data flow processing.

Common applications were for processing large streams of data flows, including for example, network data, multimedia data, big data, AI training data, etc. Ex-1012, 00021-7. For the reasons explained above, FPGAs were a popular solution for these types of data flow processing. Ex-1012, 00065, 00067.

II. THE '943 PATENT

18. The '943 patent describes “a platform for data flow processing” with a “programmable device for data flow processing in a user-configurable server with swappable pods or cards.” Ex-1001, 1:23-44. The '943 patent does not purport to invent programmable logic devices or data flow processing. It does not provide details regarding how PLDs are designed or manufactured. Rather, it references

known devices such as “[a]n FPGA (field programmable gate array), PLD (programmable logic device), ASIC (application specific integrated circuit), or some hybrid of FPGA/PLD/ASIC [] can be used in pods and cards.” Ex-1001, 3:8-12. The ’943 patent also notes the existence of prior art “systems [] appropriate for many data flow and data processing applications.” Ex-1001, 1:7-21.

19. The ’943 patent purports to invent a PLD with two regions, where the first region includes functionality, such as a router, port or bridge, that is lockable or unlockable to enable user access to sandboxes in the second region, and the second region comprises one or more sandboxes with user-definable programmable electronic circuits of the PLD. Ex-1001, Claim 1, Figs. 3-4.

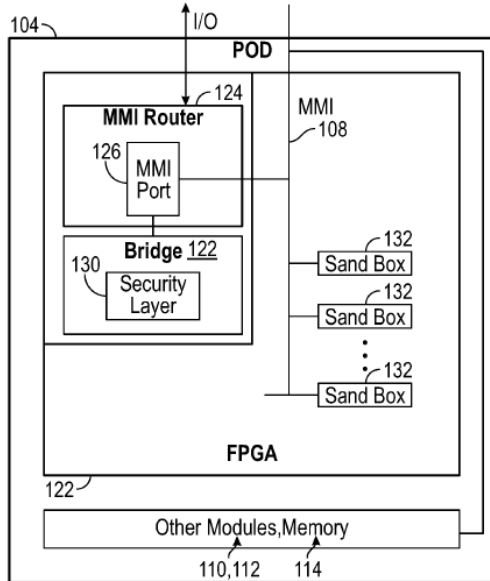


FIG. 3

Ex-1001, Fig. 3

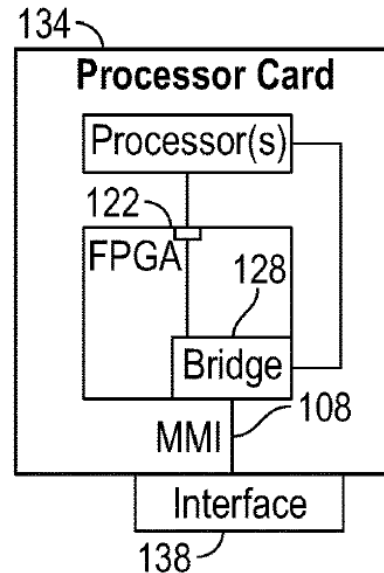


FIG. 4

Ex-1001, Fig. 4

20. However, as explained above, programmable logic devices, including FPGAs, were widely known and used in data centers for over a decade, and they were commonly used in distributed processing platforms for data-flow processing. *Supra* §I.B. As I explained above, typically included security features to prevent unauthorized access, particularly in server environments, by locking and unlocking the user-programmable portions of the FPGA. *Supra* §I.A.

A. Prosecution History

21. The '943 patent issued March 31, 2021 from Application No. 16/035,674 (“the '674 application”). I understand that the '943 patent contains no priority claim to a prior filing. The patent was allowed after a single non-final office action rejecting the claims under 35 U.S.C. §102 and §103, and a §112 objection to as-filed dependent claim 5. Ex-1002, 000139-146, 000096-110.

22. In the only office action, the Examiner rejected then-pending claims 1-4 and 6 as anticipated by the prior art reference Earl (Ex-1002, 000099-102) and claims 1, 3-4, and 7 as anticipated by the prior art reference Gerszberg (Ex-1002, 000102-103). The examiner rejected claim 2 as obvious over Gerszburg in view of another prior art reference Shur, and claim 7 as obvious over Earl in view of Gerszburg. Ex-1002, 000104-106. The Examiner also noted the disclosures of prior art made of record but not relied upon, including “sandboxes.” Ex-1002, 000106-108.

23. To overcome the anticipation and obviousness rejections, the Applicant amended claim 1 to specify the electronic circuits of the one or more sandboxes comprised the programmable electronic circuits of the PLD:

1. (currently amended) A programmable device for data flow processing in a user-configurable server with

swappable pods or cards, comprising:

a programmable logic device (PLD) having a first region and a second region;

the first region comprising a hardware-based or firmware-based router with a port enabling

communication between the router and the second region; and

the second region comprising one or more sandboxes with user-definable programmable electronic circuits of the PLD.

Ex-1002, 00125 (text formatting as in original).

24. In connection with this amendment, the Applicant argued that the cited prior art “is silent regarding a programmable logic device.” Ex-1002, 000130-132. In particular, Applicant emphasized that “[p]rogrammable logic device, or PLD, is a term of art referring to a specific type of integrated circuit that has user programmable electronic circuits.” Ex-1002, 000130.

25. The claims were allowed after an amendment initiated by the Examiner to add the last limitation of locking/unlocking the bridge to prevent/enable access to the sandboxes. Ex-1002, 000141-142. Applicant had

agreed to this amendment over a telephonic interview. Ex-1002, 000146. In the notice of allowance, the Examiner cited the Applicant's arguments about PLDs as "persuasive." Ex-1002, 000143-144.

26. The limitation of locking/unlocking the bridge was originally recited in as-filed dependent claim 5. Ex-1002, 000036. The Examiner rejected this claim as lacking written description support under 35 U.S.C. §112. *Id.*, 000098-99. Applicant argued that this limitation was disclosed in paragraphs 26-28 of the specification (Ex-1002, 000129-130), which correspond to column 4 line 48 through column 5 line 49 of the issued patent and discuss Figure 3.

27. To summarize, the single point of novelty the Applicant argued during prosecution was that the limitations were embodied in a "programmable logic device," with an Examiner's amendment to add the final limitation of locking and unlocking access to the sandboxes. However, programmable logic devices had been used on swappable PCI and PCIe cards for more than a decade before the '943 patent, including in data centers for data flow processing, and FPGAs often included security features that restricted access to programmable logic areas.

Supra §I.

B. Priority Date

28. The '674 application that issued as the '943 patent was filed on July 15, 2018, claiming priority to itself. For purposes of this IPR, I have analyzed

obviousness using a priority date of July 15, 2018, without providing any opinions on whether the patent is entitled to this priority date.

C. Challenged Claims

29. I understand that Petitioner is challenging the validity of claims 1-18 of the '943 patent in the Petition for Inter Partes Review to which this declaration will be attached. Those claims are reproduced in Appendix 3. I have considered all 18 claims of the '943 patent, as well as portions of the '943 patent prosecution history in forming my opinions.

D. Patent Owner's District Court Allegations

30. In district court, Patent Owner accused Pensando network interface cards (NICs) of infringement based on their inclusion of special-purpose processors, called DPUs (data processing units), that run software written in the P4 programming language to process packets and perform various networking functions. Ex-1021, 000002-14. Ignoring the claims' requirements of a PLD (limitation [1a]), Patent Owner broadly accused P4 processors running software of being PLDs, apparently taking a view that any type of processor is a PLD. Ex-1021, 000015:

Claim Language	Infringement Contentions and Exemplary Evidence of Infringement
[1a] a programmable logic device (PLD) having a first region and a second region;	The Accused Products comprise a programmable logic device (PLD) having a first region and a second region. For example, the AMD Pensando DSC contains a programmable logic device (PLD) having a first region and a second region. The AMD Pensando DSC includes hardware or software modules that are programmable using P4 pipelines to accelerate and customize cloud infrastructure services.

Ex-1021, 000002-14:

Claim Language	Infringement Contentions and Exemplary Evidence of Infringement
[1pre] A programmable device for data flow processing in a user-configurable server with swappable pods or cards, comprising:	To the extent the preamble is limiting, Defendants make, use, test, sell, and offer to sell the claimed programmable device for data flow processing in a user-configurable server with swappable pods or cards, including by making, using, testing, selling, and offering to sell the Accused Products; implementing and/or testing those Accused Products in their own platforms; and providing them to customers. See, e.g., AMD-CV-0003945–AMD-CV-0004184. AMD Pensando Accused Products (AMD Distributed Service Card (DSC) or AMD Pensando Distributed Modules (DSMs)) are Data Processing Units (DPUs) and enabled system and programmable devices for data flow processing in a user-configurable server with swappable pods or cards. Further, AMD Pensando DSCs are deployed in Server, Smart Switch, and Appliance platforms for data flow processing. The AMD Pensando Data Processing Unit (DPU) features user-configurable P4 Pipelines for dynamic data processing.

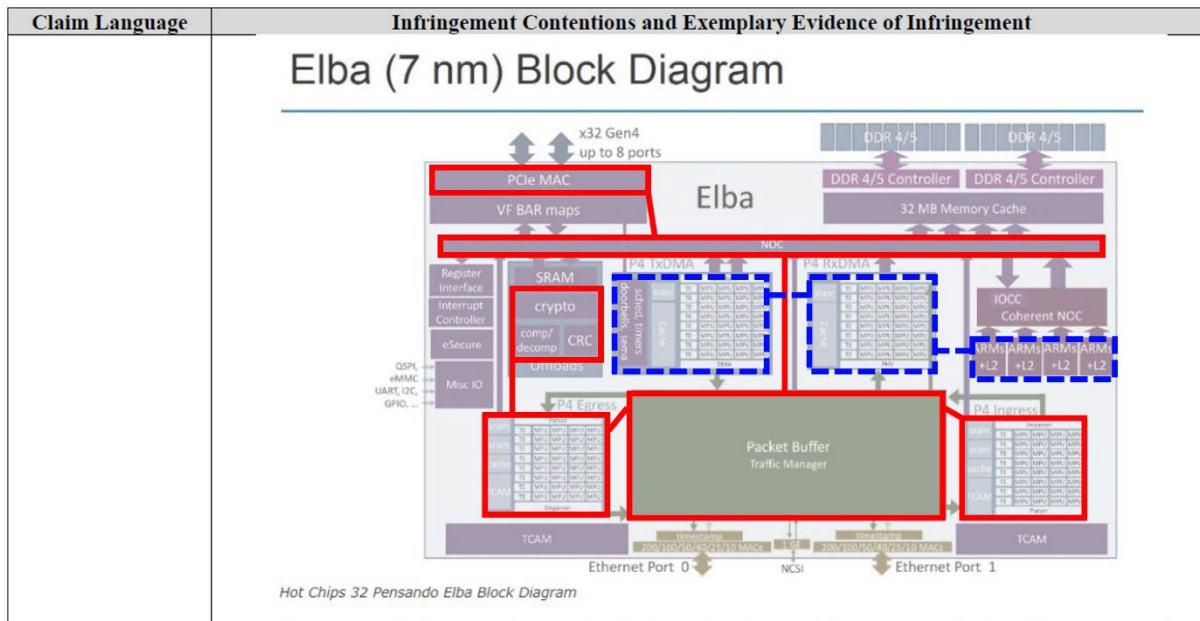
* * *

	<i>The Pensando distributed services architecture is designed to provide programmable security, network, storage, and visibility services to a data center landscape growing in scale and complexity. This domain-specific architecture combines P4 packet processing pipelines with embedded ARM CPUs and hardware offload engines in a tightly coupled, cache coherent interconnect. Millions of flows are tracked by hardware and distributed to a combination of P4 programs, ARM processes, custom hardware engines, and PCIe based hosts and devices. There are currently 16- and 7-nm ASIC implementations of this distributed services architecture, future generations are in design.</i> Michael Galles and Francis Matus, “Pensando Distributed Services Architecture” in Theme Article: Hot Chips (2021), CV-AMD0016133.
--	--

Claim Language	Infringement Contentions and Exemplary Evidence of Infringement
	P4 Pipelines Pensando ASICs include multiple P4³ pipelines (Figure 2) as the centerpiece of a domain-specific processing architecture. Each P4 pipeline starts with a programmable parser. At its core, the programmable parser includes multiple engines designed to extract header fields and populate a packet header vector (PHV) at rates of 100 million packets per second or more. Following the parser are six to eight match action stages, each stage able to load multiple tables per packet and perform complex actions which modify the PHV and update memory data structures. A single P4 pipeline includes Michael Galles and Francis Matus, “Pensando Distributed Services Architecture” in Theme Article: Hot Chips (2021), CV-AMD0016134.

Patent Owner additionally accused ARM CPU cores of being the claimed sandboxes of a PLD. Ex-1021, 000027 (“In addition, the accused product also includes ARM cores (second region) that users can program, and also are the claimed sandboxes.”).

31. For the first and second regions recited in the claims, Patent Owner created arbitrary groupings of P4 pipelines. Patent Owner accused P4 pipelines of being the recited sandboxes but placed some P4 pipelines in the first region and other P4 pipelines in the second region. Ex-1021, 000015 (“For example, the Coherent interconnect, P4 Ingress, P4 Egress and packet buffer are some of the modules in the first region. Similarly, P4 TxDMA, P4RxDMA, ARM cores are few examples of the modules within the second region”). To satisfy other claim limitations, Patent Owner lumped additional chip functions into the alleged first region. *See, e.g.*, Ex-1021, 000037-38 (discussing “various hardware security features” and “hardware based firewall at the PCIe interface” in connection with the lockable bridge of limitation [1d]). I have annotated the following figure from Patent Owner’s infringement contentions to reflect the alleged first region with solid, red lines and the alleged second region with dotted, blue lines. Ex-1021, 000034 (annotated):



32. Patent Owner has taken the view that the claimed router, bridge, and port can be commensurate in scope. For example, in its allegations for limitation [1b], Patent Owner accused “P4 Ingress and Egress pipelines,” the “packet buffer,” and the “NX Coherent Interconnect” as being in the alleged first region (Ex-1021, 000020) and stated that “[e]ach of these, *separately and together*, constitute the claimed ‘hardware-based or firmware-based router’ and ‘port.’” *Id.* Patent Owner then cited the packet buffer as the alleged bridge (Ex-1021, 000033, 000047-49) and accused a “hardware based firewall at the PCIe interface” of also being the bridge because it “prevent[s] unauthor[iz]ed access” (Ex-1021, 000037-38, 000052-55).

33. Regarding the limitation [1d]’s requirement of a bridge being lockable to prevent user access and unlockable to enable user access to each sandbox, Patent

Owner alleged that limitation [1d] is satisfied by the use of a development tool kit that copies firmware to an accused device, because the accused device “does not allow user[s] to directly access” it and “the user must log in” to verify account permissions to manage the programming of the accused card. Ex-1021, 000034-35:

Based upon information and belief, the AMD Pensando DPU does not allow user to directly access the P4 engines or the P4 pipelines. To program the P4 engines, the user must log in with SSDK compiler. Accordingly, the port thus includes a bridge that is lockable to prevent user access, and unlockable to enable user access, to each of the sandboxes and the user-definable programmable electronic circuits therein.

Infringement Contentions and Exemplary Evidence of Infringement
Copying an Image to the DSC After building the DSC firmware image you must copy it to the DSC. You can use any of the following in-band or out-of-band management interfaces: • The internal management NIC (int_mnic) interface • The memtun interface • A serial cable connected to the DCS console • The external RJ45 out-of-band (OOB) interface Using the Internal Management NIC Interface To copy the DSC firmware image using the internal management NIC (int_mnic) interface: 1. Boot up the host with the DSC plugged in. 2. <u>Login to the host.</u> https://www.amd.com/content/dam/amd/en/documents/pensando-technical-docs/user-guides/pensando-ssdk-ipsec-gw-user-guide.pdf, CV-AMD0015832.

See also, Ex-1021, 000055-56. Patent Owner further based its allegations on “various hardware security features to prevent unauthor[iz]ed access.” Ex-1021, 000037-41.

34. For claim 8, Patent Owner alleged that “scaling down processing speed/frequency” satisfies “clock cycle utilization regulation for controlling power consumption or thermal characteristics.” Ex-1021, 000058. Patent Owner’s

pointed to processors specifying operation frequencies of “up to 2 GHz” and “2.8 GHz” (Ex-1021, 000059) and asserted that this “indicate[s] that clock cycle utilization regulation takes place.” *Id.*

35. Patent Owner alleged that claim 10 was satisfied for the same reasons cited for claim 1, copying and pasting the same support from claim 1 for each limitation of claim 10. *Compare*, Ex-1021, 000074-108, *with*, Ex-1021, 000015-41.

III. LEVEL OF ORDINARY SKILL IN THE ART

36. I have analyzed the '943 patent and determined that the field of the patent is PLDs and data processing systems. The specification discusses the need to improve prior art systems “for many data flow and data processing applications.” Ex-1001, 1:6-21. “Video processing is one such data processing application....” *Id.* PLDs are recited in each claim and discussed extensively in the embodiments. E.g., *id.*, 3:8-12, 4:27-47, 6:37-45, 5:25-38. The '943 patent is titled “Hardware Defined Anything...” which I understand is a reference to the user-definable hardware discussed in the patent’s claims and specification. Embodiments of the alleged invention include “a platform for data flow processing” and “[a] programmable device for data flow processing.” Ex-1001, 1:23-44. The specification further explains that it provides “a variety of flexible

features for hardware-defined and software-defined functionality in data streaming and data processing systems, including video.” Ex-1001, 2:47-3:20.

37. In determining the characteristics of a hypothetical person of ordinary skill in the art (“POSITA”) of the ’943 patent at the time of the claimed invention, I considered several things, including various prior art techniques relating to the field (discussed above), the type of problems that such techniques gave rise to, and the rapidity with which innovations were made. I also considered the sophistication of the technologies involved, and the educational background and experience of those actively working in the field at the time. I also considered the level of education that would be necessary to understand the ’943 patent. Finally, I placed myself back in the relevant period of time and considered the engineers and programmers that I have worked with and managed in the field of PLDs and data processing systems.

38. I came to the conclusion that a person of ordinary skill in the field of art of the ’943 patent would have been a person with a bachelor’s degree in computer engineering, electrical engineering, or a similar major with at least two years of experience in the field (described above) or a person with a master’s degree in computer engineering, electrical engineering, computer science, or a similar field with a specialization in the field (described above). A person with less education but more relevant practical experience may also meet this standard.

39. In the 2018 timeframe, a POSITA would have known and had the skills necessary to:

- Design distributed processing and streaming systems for data flow processing (including the processing of network data, multimedia data, big data, machine-learning training data, etc.);
- Understand and deploy hardware and software components used for streaming, networking, and data processing;
- Understand integrated circuit architectures, including those of programmable logic devices;
- Design and implement programmable logic devices such as FPGAs;
- Develop and configure logic circuits in programmable logic devices such as FPGAs;
- Program software and firmware for general purpose CPUs and microprocessors;
- Implement known algorithms as electronic circuits or software; and
- Interface between various bus protocols and communication systems.

IV. CLAIM CONSTRUCTION

40. For purposes of this inter partes review, I have considered the claim language, specification, and portions of the prosecution history, to determine the meaning of the claim language as it would have been understood by a person of

ordinary skill in the art at the time of the invention. The “plain and ordinary meaning” or Phillips standard has traditionally been applied in district court litigation, where a claim term is given its plain and ordinary meaning in view of the specification from the view point of a person of ordinary skill in the art.

41. I have applied the Phillips standard in my analysis. Unless otherwise stated, I have applied the plain and ordinary meaning to claim terms.

A. “programmable logic device”

42. Each prior art ground teaches a programmable logic device irrespective of whether the below construction is adopted.

43. Based on my review of the claims and specification of the '943 patent, it is my opinion that a POSITA would have understood that “programmable logic device” means an integrated circuit where the user can alter the connection pattern of the circuits.

44. The independent claims recite “programmable logic device” and define an acronym “PLD.” Ex-1001, Claims 1 (limitation [1a]), 10 (limitation [10a]). The claims require the PLD to have “user-definable programmable electronic circuits.” Ex-1001, Claims 1 (limitation [1c]), 10 (limitation [10c]).

45. Consistent with the claims, the specification describes the claimed device to be any “other programmable device that can configure electronic

circuitry.” Ex-1004, 4:30-47. “The programmable device could be, for example, an FPGA..., a PLD or an ASIC....” Ex-1004, 4:30-35, 3:8-12.

46. During prosecution, the Applicant argued for this construction, stating that “[p]rogrammable logic device, or PLD, *is a term of art*^[1] referring to a specific type of integrated circuit that has user programmable electronic circuits.” Ex-1002, 000130-132.

47. This construction is consistent with the conventional meaning of the term, which a POSITA with experience in the field would have recognized, as comprising a category of devices with hardware circuits that are programmable. For example, the JEDEC standard shares this understanding, defining the term as “[a]n integrated circuit consisting of an array of logic elements whose interconnection pattern can be programmed (either mask-programmed or user-programmed) to perform specific logic functions.” Ex-1029, 000018 (entry for “programmable logic device (PLD)”). The JEDEC standard further explains that programming or configuring in the context of PLDs refers to “modify[ing] the logic connections inside a programmable device, causing its functions to change.” Ex-1029, 000018 (entry for “program; configure”).

¹ All emphasis added unless otherwise noted.

48. The IEEE Standards for In-System Configuration of Programmable Devices defines “programmable logic devices” as the family of logic devices in which the hardware circuits are programmable. Ex-1030, 000014 (§3.1 Definitions, bolding in original):

3.1.48 programmable logic devices: This family of devices includes complex programmable logic device (CPLD), electrically erasable programmable read-only memory (EEPROM), erasable programmable logic device (EPLD), field-programmable gate array (FPGA), flash random access memory, and programmable array logic.

49. The '943 patent states that “[t]he programmable device could be, for example, an FPGA..., a PLD or an ASIC, or more than one of these....” Ex-1001, 4:30-35, 3:8-12. However the abstract and summary of the invention is silent to any subcategories and only mentions that the “device [(the alleged invention)] includes a programmable logic device (PLD).” Ex-1004, 1:23-49. Therefore, the specification’s discussion of “an FPGA..., a PLD or an ASIC,...or other programmable device that can configure electronic circuitry” refers to different kinds of devices that fall under the category of a PLD, consistent with the known meaning in the art as discussed above. A POSITA would not have understood this listing, from columns 3-4, as distinguishing FPGAs from PLDs. Rather, the '943 patent specifically points out FPGAs (which was a commonly used type of PLD

and the only embodiment discussed in detail in the specification), while claiming PLDs in general without specifically naming any subcategories.

V. INVALIDITY GROUNDS

50. Based on my review and analysis of the materials cited herein, my opinions regarding the understanding of a POSITA in the 2018 timeframe, and my training and experience, it is my opinion that the challenged claims of the '943 patent are invalid based on the following grounds:

Grounds	Claims	Statutory Basis	Prior Art
1	1-5, 10-14	§103	Seshadri
2	1-5, 9-14, 18	§103	Seshadri in view of Biederman
3	1-5, 10-14	§103	Atsatt-113
4	1-5, 9-14, 18	§103	Atsatt-113 in view of Biederman
5	6, 15	§103	Grounds 1-4 in view of Tarafdar
6	7, 16	§103	Grounds 1-4 in view of Kulkarni
7	8, 17	§103	Grounds 1-4 in view of Atsatt-923

The reasons for my conclusions are explained more fully below.

A. Grounds 1-2: the Challenged Claims are Obvious based on Seshadri alone (Ground 1) and Seshadri in view of Biederman (Ground 2).

51. It is my opinion that a POSITA would have found the Challenged Claims obvious based on Seshadri's teachings for PLDs, including FPGAs (Ground 1), as I explain below in detail. *Infra* §V.A.3-9. The combination with

Biederman (Ground 2) provides additional teachings for using and resizing multiple partitions in an FPGA, which is relevant for claims 9, 10 and 18.

52. A POSITA would have been motivated to combine Seshadri with Biederman because both references use FPGAs to accelerate data processing in the context of multi-user servers and virtual machines, and Biederman teaches a technique for managing FPGA partitions in a multi-user environment. *Infra* §V.A.1, §V.A.2. I explain the motivation to combine and reasonable expectation of success in further detail for the individual limitations below. *Infra* §V.B.9 (Motivation to Combine subsection).

1. U.S. Patent No. 10,776,142 (“Seshadri”) (Ex-1004)

53. I have reviewed the Seshadri reference. I understand that Seshadri was not cited or considered during prosecution of the ’943 patent, based primarily on the fact that Seshadri is not cited on the face of the patent, nor have I seen the reference discussed in the prosecution history.

54. Seshadri is entitled “Reconfiguring programmable hardware when a virtual machine is active.” Ex-1004, (54) Title. Seshadri was filed on December 13, 2017 and issued on September 15, 2020. Based on the priority date of July 15, 2018 of the ’943 patent, I understand Seshadri is prior art to the ’943 patent under post-AIA §102(a)(2).

55. Seshadri teaches the use of “programmable integrated circuits (ICs), such as field programmable gate arrays (FPGAs)” in networked server nodes with circuits that users (clients) can configure with custom logic. Ex-1004, 10:36-47, 2:60-3:27, Fig. 3:

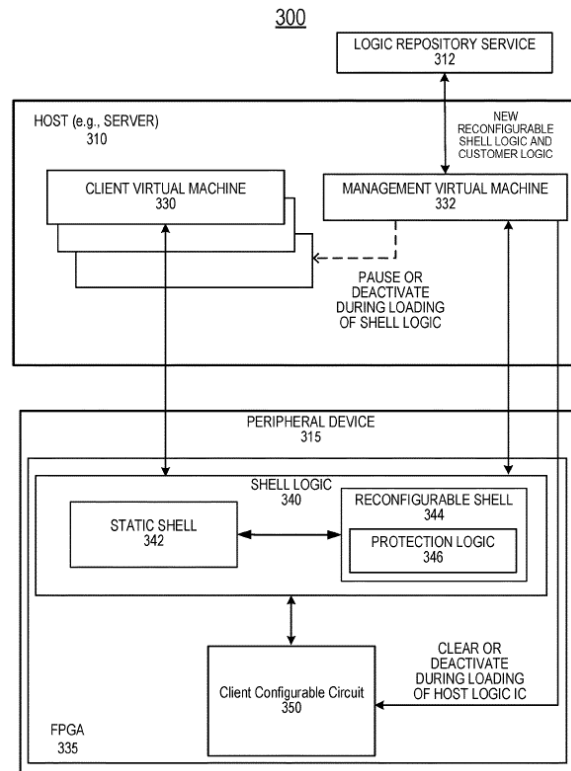


FIG. 3

This architecture allowed users to deploy custom logic in a distributed, scalable computing environment. *Id.*, 2:60-3:27.

56. Because users are each assigned dedicated space on FPGAs, Seshadri partitions the programmable fabric of the FPGA into different areas, including “[s]hell logic 340 and client configurable circuit 350.” Ex-1004, 10:36-49, 10:7-11. The shell logic “wrap[s] or encapsulat[es] the client configurable circuit[,]”

and provides “bridging functions” to control communication between the client-configurable circuit and the external system. Ex-1004, 3:41-64. The shell logic also checks user account permissions when reconfiguring the client-configurable circuit with new logic. Ex-1004, 9:48-10:15, Fig. 6:

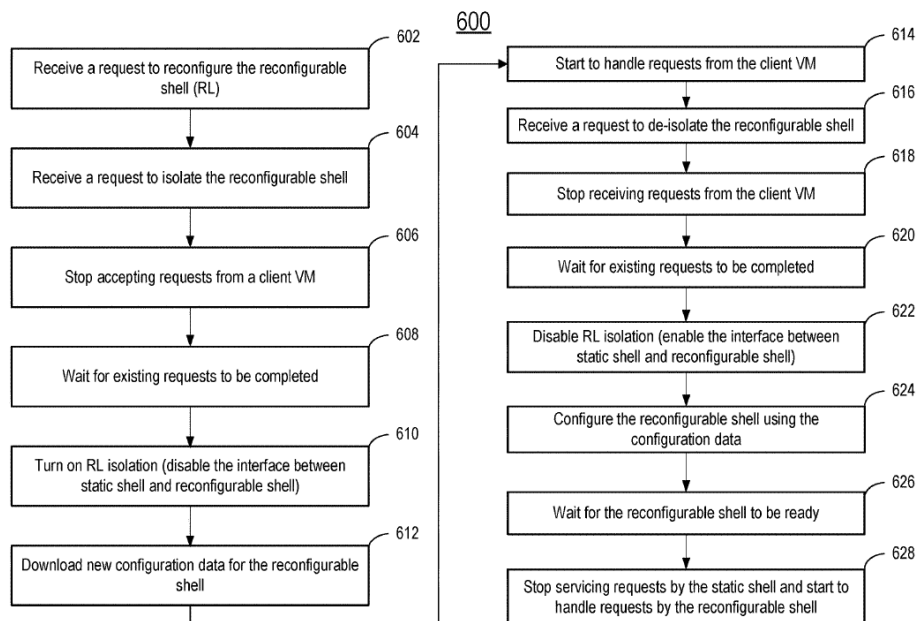


FIG. 6

57. Seshadri is analogous art to the '943 patent because it is in the same field of endeavor as the '943 patent, which is PLDs and data processing systems. *Supra* §III. Seshadri proposes an architecture for processing streams of data with FPGAs in a server environment. Ex-1004, 4:2-11, 26:27-61, 20:67-21:4, Fig. 1. The FPGAs are provided to users as “networked computing resources” for customized data processing. Ex-1004, 1:5-32, Figs. 1 & 8. Additionally, Seshadri is reasonably pertinent to the problem addressed by the '943 patent because, like the '943 patent, Seshadri contemplates the use of FPGAs in servers, and the use of

multiple reconfigurable processing elements in servers for data flow processing.

Id.

**2. U.S. Patent Application Publication No. 2018/0191642A1
("Biederman") (Ex-1005)**

58. I have reviewed the Biederman reference. I understand that Biederman was not cited or considered during prosecution of the '943 Patent, based primarily on the fact that Biederman is not cited on the face of the patent, nor have I seen the reference discussed in the prosecution history.

59. Biederman was filed on December 30, 2016 and was published on July 5, 2018. Based on the priority date of July 15, 2018 of the '943 patent, I understand Biederman is prior art to the '943 patent under post-AIA §§102(a)(1)-(2).

60. Biederman discloses "a network interface device" that processes network data streams using an FPGA, distributing packets to processing elements on computing units, including programmable logic devices such as FPGAs. Ex-1005, Abstract, ¶¶27-30, Fig. 3:

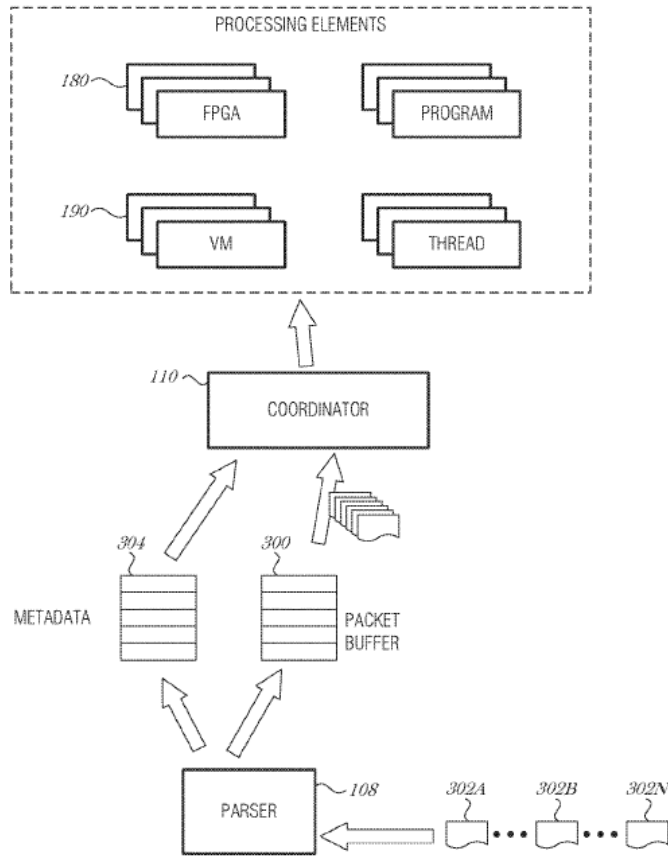


FIG. 3

Biederman applies its teachings for server-based data flow processing services such as software defined networking, network function virtualization, virtual switches, and data centers. Ex-1005, ¶¶174-177, ¶¶182-184.

61. Biederman discloses techniques to reconfigure the logic in the FPGA by resizing partitions. Ex-1005, ¶134. Biederman recognizes that using “a larger cache/partition space...for complex processing, [can] ensur[e] faster processing and a better user experience.” Ex-1005, ¶134. Figure 10 illustrates Biederman’s

process for changing the “program to load into an FPGA” based on the priority of the operation, in order to ensure processing in time:

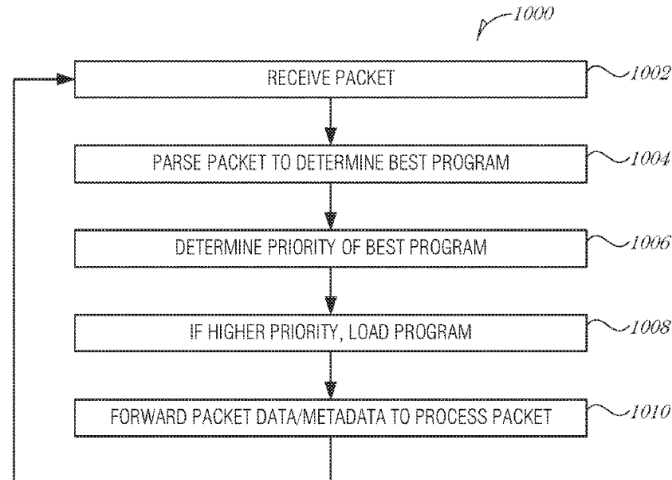


FIG. 10

62. Biederman is analogous art to the '943 patent because it is in the same field of endeavor as the '943 patent, which is PLDs and data processing systems. *Supra* §III. Biederman discloses an architecture for processing streams of network data, including video data streams. Ex-1005, (57) Abstract, ¶134. For example, Biederman parses an incoming stream of network packets and passes the data to the appropriate logic hosted in different partitions of an FPGA for processing. Ex-1005, ¶123-129, Fig. 8. Biederman is reasonably pertinent to the problem addressed by the '943 patent because, like the '943 patent, Biederman contemplates the use of multiple reconfigurable FPGA processing elements in a server for data flow processing.

3. Independent Claim 1

1[pre]. A programmable device for data flow processing in a user-configurable server with swappable pods or cards, comprising:

63. To the extent the preamble is limiting, it is my opinion that Seshadri teaches it.

64. Seshadri teaches a **programmable device...in a user-configurable server with swappable pods or cards**. For example, Seshadri discloses “a programmable circuit (e.g., an FPGA).” Ex-1004, 2:44-51, 2:60-3:11, 10:36-65. Seshadri uses the FPGA in a “configurable computing system” with “configurable device[s]” that allows remote users to configure the server by loading custom logic on FPGAs for client virtual machines to utilize. Ex-1004, Abstract, 13:40-14:6, 20:67-21:4 (“a client application using the client virtual machine may perform a memory mapping using DMA using a first customer logic at a first time period and may desire to use a second customer logic for a streaming process during a second time period”). This includes servers providing “networked computing resources.” Ex-1004, 1:7-22, Figs. 1-2 (computing devices 100a-c in host domain 101):

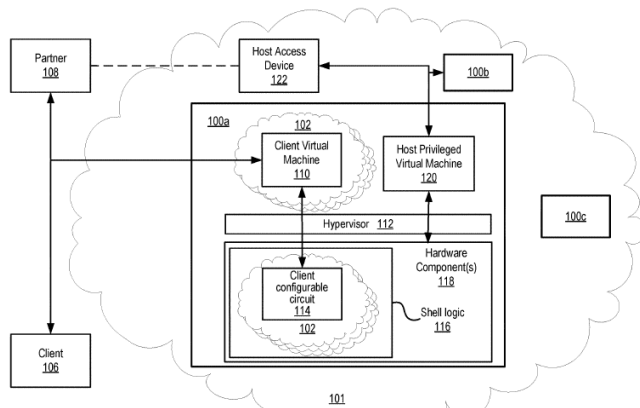
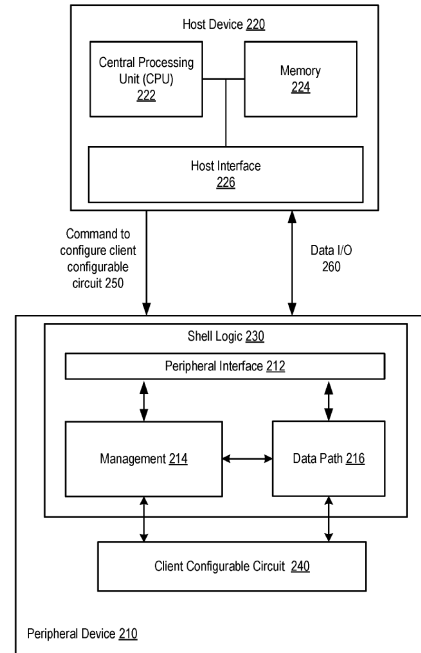


FIG. 1



See also Ex-1004, Fig. 8.

65. Seshadri explains, for example, that “[a] computing service provider can provide systems with varying combinations of processing performance, memory capacity and performance, storage capacity and performance, and networking capacity and performance.” Ex-1004, 1:7-22. Seshadri provides a more detailed example of its computing systems that includes “a host 412 (e.g., a PCIe *server*) and a peripheral device 410....” Ex-1004, 13:40-48, Fig. 4 (highlighted):

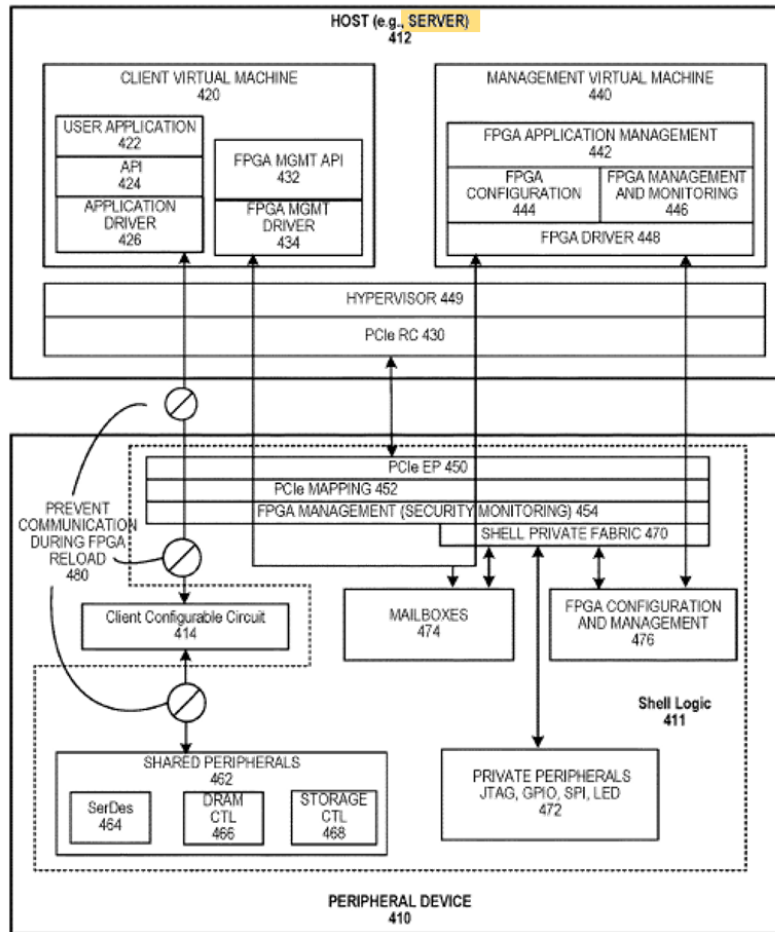


FIG. 4

66. Seshadri's servers include swappable cards with FPGAs. For example, the host servers are equipped with "plug-in cards" that have "programmable ICs." Ex-1004, 13:40-48 ("Peripheral device 410 may include an FPGA or other types of programmable ICs, and may be positioned on one or more plug-in cards on host 412...."), 2:60-3:11 ("a computing device with an *add-in card* or other peripheral device including a field-programmable gate array

(FPGA).”). Therefore, users configured servers by swapping out FPGA cards and loading those cards with custom logic. Ex-1004, Abstract, 13:40-14:6.

67. Seshadri teaches using the servers and FPGAs for **data flow processing**. For example, Seshadri discloses loading user logic “to perform data streaming”² using “different customer logics.” Ex-1004, 4:2-9. Seshadri further teaches using servers for cloud computing and Internet applications, which process flows of data in packets. *Id.*, 1:7-31.

68. For at least the reasons explained above, Ground 2 also teaches limitation [1a].

[1a] a programmable logic device (PLD) having a first region and a second region;

69. Seshadri teaches this limitation. For example, Seshadri discloses “programmable integrated circuits (ICs)” that have a first “shell logic” region and a

² The ’943 patent confirms that data streaming falls within the scope of the recited “data flow processing” because the specification describes “processing stream live data” as an example of data flow processing. Ex-1001, 11:42-12:42, 3:50-64, Figs. 1B, 10.

second “client configurable circuit” region.³ Ex-1004, 10:36-47 (“[I]n some embodiments, shell logic 340 and client configurable circuit 350 may be implemented on an FPGA 335 residing on peripheral device 315.”), 2:44-3:11. Figure 2 illustrates the main functional blocks of Seshadri’s PLD configurations. Ex-1004, Fig. 2 (excerpt, annotated):

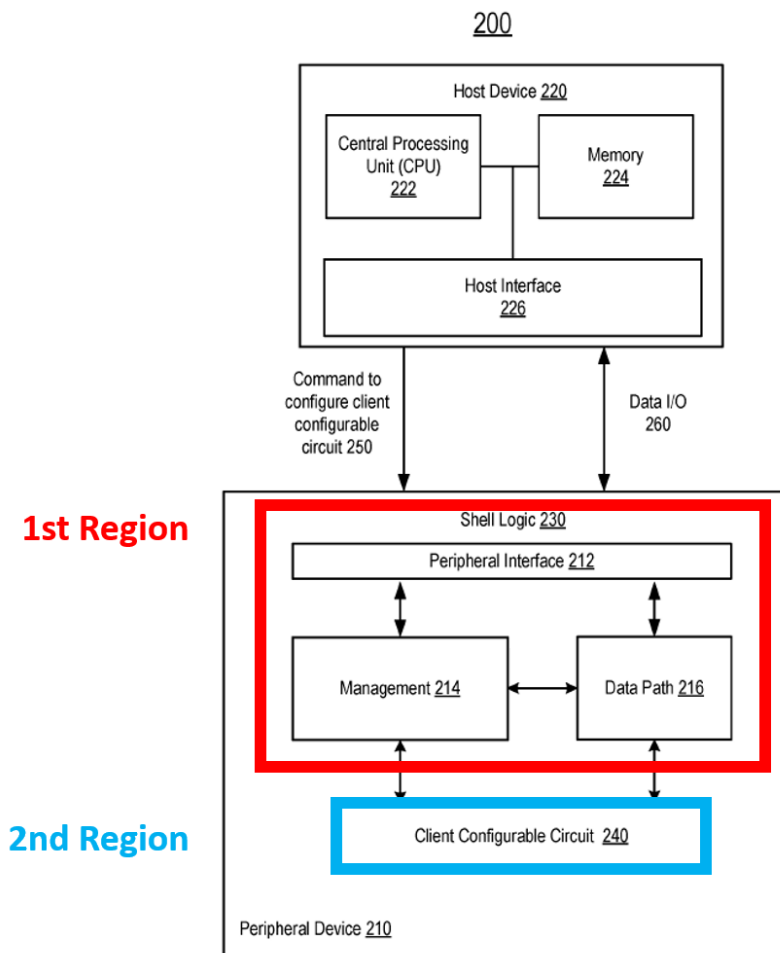


FIG. 2

³ I note that Patent Owner took a broad view of the first and second regions recited in the claims, as I explained above. Ex-1021, 000002-15, 000027; *supra* §II.D.

See also Ex-1004, Fig. 3 (further illustrating blocks and steps related to reconfiguration of client-configurable circuit), Fig. 5 (further illustrating functional blocks within static shell).

70. Seshadri uses a number of figures to illustrate aspects of its exemplary computing system, which are all consistent with the general configuration of Figure 2. *See* Ex-1004, 1:40-51, Figs. 2-5. Seshadri explains that “FIG. 2 is a simplified diagram” of the host device, card, and FPGA. Ex-1004, 6:42-52. Figure 3 further illustrates retrieving customer logics from a repository service to configure the client-configurable circuit and the reconfigurable portion of the shell logic. Ex-1004, 10:16-35, 10:66-11:30, Fig. 3. “FIG. 4 is a block diagram showing further details” of the computing system. Ex-1004, 13:40-47, Fig. 4. Figure 5 further illustrates functional blocks within the shell logic that support continuous operation during reconfiguration. Ex-1004, 16:61-17:15, Fig. 5. Rather than describing alternative embodiments, Seshadri’s teachings are complementary, overlapping, and meant to be used together. Indeed, as Seshadri explains for Figures 4 and 5, the different figures “include the software and hardware components described above with respect to” the other figures, even if they are not shown. Ex-1004, 17:16-36 (“Peripheral device 560 may include the software and hardware components described above with respect to, for example,

peripheral device 410 of FIG. 4, some of which may not be shown in FIG. 5 in order not to obscure the example embodiment in unnecessary detail.”).

71. Seshadri teaches programmable logic devices including programmable ICs “such as field programmable gate arrays (FPGAs)^[4] or programmable array logics (PALs).”⁵ Ex-1004, 10:36-47, 2:44-3:11 (“programmable circuit (e.g., an FPGA)”); *supra* §I.A.

72. Seshadri also teaches a PLD under its proper construction: an IC where the user can alter the connection pattern of the circuits. *Supra* §IV.A. Seshadri teaches an FPGA (Ex-1004, 10:36-47, 13:40-48, 20:46-48, Fig. 3) and

⁴ The ’943 patent confirms that FPGAs fall within the scope of the claimed “PLD” because the written description in the specification uses FPGAs as the PLD. Ex-1001, 4:27-5:49, 13:14-16, Figs. 3-4. Thus, the ’943 patent uses PLD to broadly refer to a class of devices that includes FPGAs (explained in my claim construction analysis *supra* §IV.A). *Id.*

⁵ Because Seshadri explains its teachings using an FPGA, I will refer to an FPGA when discussing Seshadri to refer to Seshadri’s teachings explained in connection with an FPGA as well as its teachings that are generally applicable to other types of PLDs, such as “PALs” as Seshadri suggests. Ex-1004, 10:36-47, 29:62-30:11 (“Other variations are within the spirit of the present disclosure.”).

further explains that the FPGA's programmable circuits can be altered to perform a logic function specified by the user, meaning the connection pattern of the circuits is modified to implement the user's logic function. Ex-1004, 2:50-3:11 ("The client configurable circuit may be hardware that can be programmed or configured to perform a logic function specified by configuration data that is applied to or loaded on the configurable circuit."), 2:44-51 ("In one example, a logic may be implemented using a programmable circuit (e.g., an FPGA) and a configuration file (e.g., a bit stream) stored in a nonvolatile memory."). Thus, Seshadri teaches that the user (client) programs the actual FPGA circuits, including their logical interconnections, to perform specific logic functions. *Id.*

73. Below, I explain Seshadri's teachings of the first and second regions in more detail for limitations [1b]-[1d]. For at least the reasons explained above, Ground 2 also teaches limitation [1a].

[1b] the first region comprising a hardware-based or firmware-based router with a port enabling communication between the router and the second region; and

74. As I explained above for limitation [1a], Seshadri teaches a PLD (e.g., FPGA) having a first region (e.g., shell logic) and a second region (e.g., client-configurable circuit). *Supra* §V.A.3[1a]. Seshadri further teaches the first region

comprising a router with a port⁶ enabling communication with the second region.

Ex-1004, Fig. 2:

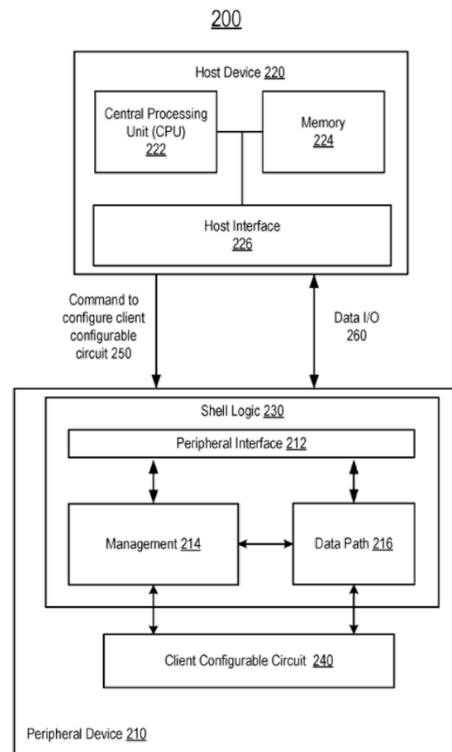


FIG. 2

75. Regarding the router, the first shell logic region includes functionality that “perform[s] bridging functions between the local interconnect (e.g., the PCIe interconnect) and the client configurable logic.” Ex-1004, 6:11-41. This

⁶ Patent Owner alleged that various IC components “*separately and together*, constitute the claimed ‘hardware-based or firmware-based router’ and ‘port.’” Ex-1021, 000020; *supra* §II.D (discussing Patent Owner’s allegations as to limitation [1b]).

functionality (including but not limited to management functions and the data path) routes message communication between the shell logic and the client-configurable circuit in order to convey messages to/from the host device. Ex-1004, 8:37-61 (*“Data path module 216 in shell logic 230 may be used to exchange information (e.g., data input/output 260) between host device 220 and peripheral device 210....Thus, data path module 216 can be interposed between client configurable circuit 240 and the physical interconnect (e.g., the PCIe bus).”*), Fig. 2. Inbound communication from the host device is received by the FPGA over the peripheral interface in the shell logic, and functionality in the shell logic “route[s] commands to functional modules or blocks of peripheral device 210 based on an address.” Ex-1004, 7:56-8:8; *see also id.*, 14:37-53, Fig. 4. Outbound communication from the FPGA to the host device is passed from the client-configurable circuits to the peripheral interface in the shell logic that “us[es] a command including an address associated with host device 220.” *Id.* The data path module routes communications between the first (shell logic) region and the second (client-configurable circuit) region. *Id.* The data path facilitates this communication because the client virtual machine on the host device relies on the accelerator logic in the client-configurable circuit for its processing. *Id.*, 5:32-57 (“Client domain 102 may include associated virtual or physical components, such as a virtual

machine running on a host and the underlying customer logic implemented in client configurable circuit 114.”), 6:43-51, 12:32-49; *see also, id.*, 16:3-10.

76. To facilitate routing of messages with the second (client-configurable circuit) region, the routing functionality further includes a port that enables communication with the second region. This includes, for example, management functions and the data-path, such as the mailbox functionality through which messages and information are passed between the first region and the second region. Seshadri explains this in terms of communication between the control plane (including the shell logic) and the data plane (including the client-configurable circuit). Ex-1004, 14:63-15:8 (“Shell logic 411 of peripheral device 410 may include mailboxes 474 that are accessible from either the control plane or the data plane described above. Specifically, *mailboxes 474 may be used to pass messages and other information between the control plane and the data plane. Mailboxes 474 may include buffers, control registers (such as semaphores), and status registers....*Mailboxes 474 may also allow communication between client virtual machine 420 and management virtual machine 440.”), 12:32-49 (“*The control plane functions may include shell logic 340. ...The data plane functions may be used to manage data transfer between host 310 and the customer defined logic implemented using client configurable circuit 350. For example, a data plane function may be used to accelerate a client's application that is running on host*

310.”). The mailboxes comprise part of the data path because they provide communication functionality. Figure 4 illustrates the mailboxes’ relation to some of the other functions in the FPGA and the host server. Ex-1004, Fig. 4 (highlighted):

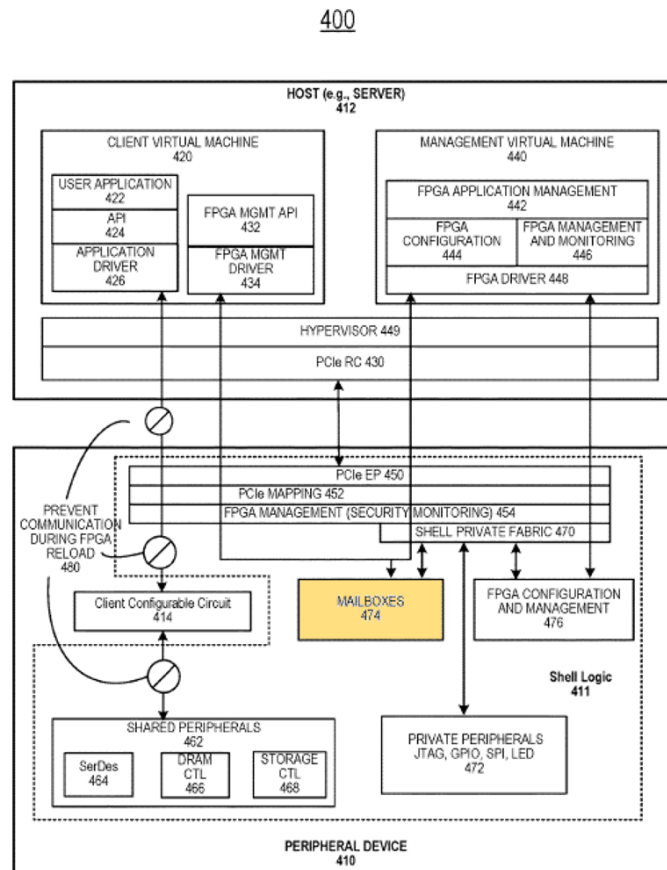


FIG. 4

77. In summary, Seshadri teaches a port that comprises the functionality in the shell logic for enabling communication with the client configurable circuits. The port also comprises a bridge, explained for limitation [1d]. *Infra* §V.A.3[1d]. Seshadri teaches a router that comprises the shell logic port and other functionality in the shell logic for bridging and routing communications.

78. Seshadri teaches the router in the shell logic is hardware-based.

Seshadri teaches the shell logic being loaded into the programmable fabric⁷ of the FPGA. Ex-1004, 2:2-5 (“the client configurable circuit can be programmed with a shell logic”), 3:24-27 (same). Because the shell logic is implemented using electronic circuits configured in the hardware of an IC, it is hardware-based.

79. For at least the reasons explained above, Ground 2 also teaches limitation [1b].

[1c] the second region comprising one or more sandboxes with user-definable programmable electronic circuits of the PLD,

80. As I explained above for limitation [1a], Seshadri teaches a PLD (e.g., FPGA) having a first region (e.g., shell logic) and a second region (e.g., client-configurable circuit). *Supra* §V.A.3[1a]. Seshadri further teaches the limitations of [1c].

⁷ Here, the discussion in Sheshadri columns 2-3 regarding a “client configurable circuit” that can be programmed with shell logic and client logic refers generally to the programmable fabric of an FPGA. Ex-1004, 2:2-5, 2:44-54, 3:2-14, 3:41-64. Elsewhere, Sheshadri describes a particular portion of the programmable fabric that is left for custom client logic, which is called the “client configurable circuit” in Figures 1-5.

81. Seshadri teaches **the second region comprising one or more sandboxes**. For example, Seshadri discloses “sandboxing” of the client-configurable circuit (second region). Ex-1004, 10:66-11:16. Seshadri explains that “[s]hell logic 340 may provide encapsulation or sandboxing of client configurable circuit 350 so that one client may not obtain secure information associated with the operation of another client's customer logic.” *Id.* The shell logic isolates the client-configurable circuit from other client-configurable circuits and associated resources. *See also* Ex-1004, 2:44-59 (“a shell may refer to a wrapper of a configurable circuit, where the wrapper may isolate the configurable circuit”), 6:11-41 (“Encapsulating the client configurable logic can limit or restrict the client configurable logic’s access to configuration resources, physical interfaces, hard macros, and various peripherals of a computing device.”). Therefore, Seshadri teaches user sandboxes that are isolated from each other. *Id.*, 10:66-11:16, 2:44-59.

82. The client-configurable circuit comprises one or more sandboxes with **user-definable programmable electronic circuits of the PLD** (e.g., FPGA) that the user can program and reprogram with user-defined logic. For example, Seshadri discloses the “client configurable circuit 240 [] may be used to implement a customer logic for the client.” Ex-1004, 7:22-36, 12:10-31 (“*Client configurable circuit 350 may be reconfigured and controlled by a client* associated with a client

virtual machine 330 running on host 310.”). Seshadri explains the client-configurable circuit is configured with user-provided specifications that define the logic functions of the circuit. Ex-1004, 3:2-14:

The client configurable circuit may be hardware that can be programmed or configured to perform a logic function specified by configuration data that is applied to or loaded on the configurable circuit. For example, *a client of a computing system may provide a specification (such as source code written in a hardware description language or a configuration file) for configuring the client configurable circuit, and the client configurable circuit may then be configured according to the specification to perform a task for the client.* For example, the client can program the client configurable circuit so that it functions as a hardware accelerator that is tightly coupled to the computing device.

These client-configurable circuits in the FPGA are defined by the user because the user programs the actual interconnections of the circuits of the FPGA to create logical functions defined by the user (the “customer logic”). Ex-1004, 7:22-36, 10:36-65, 12:10-31.

83. The sandboxes in the client-configurable logic are user-definable because the “client configurable circuit 350 ...may be part of one or more FPGAs.” Ex-1004, 10:36-65. Seshadri further explains that these consist of the programmable electronic circuits of the FPGA. Ex-1004, 17:16-36:

...[C]lient configurable circuit 595 may include *programmable logic blocks (e.g., FPGAs or PALs) comprising combinational logic and/or LUTs and sequential logic elements (such as flip-flops and/or latches), programmable routing and clocking resources, programmable distributed and block RAMs, digital signal processing (DSP) bit slices, and/or programmable input/output pins*. Client configurable circuit 595 may be programmed or otherwise configured to implement customer defined logic, as described above with respect to, for example, FIGS. 3 and 4.

84. Additionally, integrated circuits are electronic circuits, and the configurable logic elements in PLDs, including FPGAs, were electronic circuits.

85. For at least the reasons explained above, Seshadri teaches limitation [1c].

[1d] wherein the port comprises: a bridge being lockable to prevent user access, and unlockable to enable user access, to each of the sandboxes and the user-definable programmable electronic circuits therein.
--

86. As I explained above for limitation [1a], Seshadri teaches a PLD (e.g., FPGA) having a first region (e.g., shell logic) and a second region (e.g., client-configurable circuit). *Supra* §V.A.3.[1a]. The first (shell logic) region comprises a router with a port for enabling communication with the client-configurable circuits. *Supra* §V.A.3.[1b]. Seshadri further teaches that the port functionality comprises a bridge being lockable/unlockable to prevent/enable user access to each of the sandboxes (e.g., in the client-configurable circuit).

87. Seshadri teaches that **the port comprises a bridge**. For example, Seshadri explains how the shell logic includes functionality to perform “bridging functions” between the PCIe interface and the client-configurable logic. Ex-1004, 3:41-64 (“[T]he shell logic can perform bridging functions between the local interconnect (e.g., the PCIe interconnect) and the client configurable circuit...”), 6:11-41 (same). “The static shell [logic] may also include circuits...for bridging the communications between the client virtual machine and a management virtual machine on the host.” Ex-1004, 4:11-41. For example, the shell logic’s “mailbox 588 may be used as a bridge to enable communications between client virtual machines 520 and management virtual machine 530.” Ex-1004, 17:37-57, 18:4-24 (“mailbox 588 for bridging the communications”), Fig. 5 (“Mailbox(es) 588”):

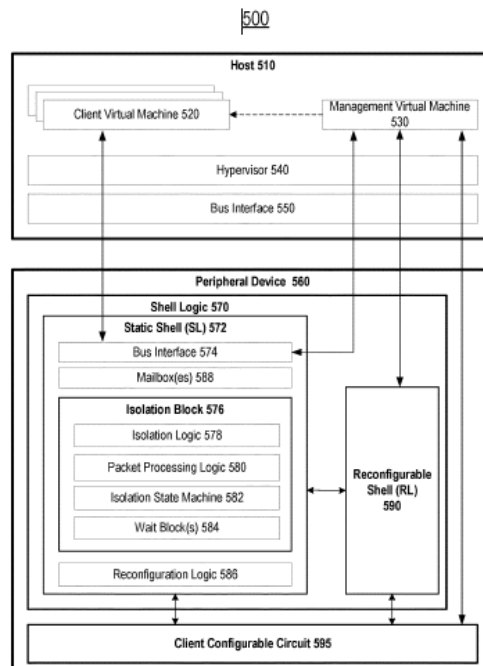


FIG. 5

88. Seshadri teaches the bridge is **lockable to prevent user access, and unlockable to enable user access, to each of the sandboxes and the user-definable programmable electronic circuits therein.**⁸ For example, the shell logic includes functionality to “isolate a configuration access port”; this functionality is locked and unlocked by the shell logic to disallow or allow configuration of the client-configurable circuit. Ex-1004, 6:11-41 (“[T]he computing service provider may manage the programming of the client configurable circuit using the shell logic, which may provide a framework for the client configurable logic to work within....For example, the shell logic may isolate a configuration access port so that the customer logic cannot cause the client configurable circuit to be reprogrammed without using services provided by the computing service provider (e.g., the shell logic).”). That is, “[t]he programming

⁸ In district court, Patent Owner alleged this limitation is satisfied by logging in to a development tool kit to copy firmware to a device because the device “does not allow user[s] to directly access” it and “the user must log in” to verify account permissions to manage the programming of the accused card. Ex-1021, 000034-35, 000055-56; *supra* §II.D. Patent Owner further based its allegations on “various hardware security features to prevent unauthor[iz]ed access.” Ex-1021, 000037-41.

of client configurable circuit 350 may be controlled and managed by shell logic 340.” Ex-1004, 12:10-31.

89. As a first step, the user requests reconfiguration through the management virtual machine on the host server, which in turn “send[s] a transaction to FPGA configuration and management block 476 to initiate loading of the customer defined logic on client configurable circuit 414.” Ex-1004, 15:9-17. The shell logic rejects or accepts the request by checking user account permissions to lock or unlock the reprogramming functionality for the client-configurable circuit. Ex-1004, 9:48-10:15:

[I]n order to protect the host during a transition that replaces the first customer logic with the second customer logic, the following operations may be performed in response to a request from a client virtual machine to load a new customer logic. *First, the customer account permissions can be checked.* A check may then be performed to determine whether an appropriate reconfigurable shell is already loaded and, if so, then the customer logic can be loaded. Otherwise, a check can be made to ensure that the load of the new customer logic can be performed. For example, the load of the new customer logic can be performed if the reconfigurable shell for the new customer logic is available and host configurable circuits in the peripheral device can be used to implement the reconfigurable shell. *A response can be generated to inform the client virtual machine that the load request is accepted.* ...Subsequently, the reconfigurable shell can then be configured on the host configurable circuits, and the customer logic

can be configured on the client configurable circuits and activated using the reconfigurable shell.

By locking and unlocking the bridge to control user access to programming, Seshadri “increase[s] the security, performance, privacy, and/or availability of the computing resources” in the FPGA. Ex-1004, 3:41-64.

90. In summary, Seshadri’s shell logic includes bridging functionality that is locked to prevent users from rewriting the FPGA and unlocked to enable users to reconfigure the sandboxed portion of the client-configurable circuit that they have account permissions to configure. Ex-1004, 9:48-10:15, 6:11-41, 12:10-31. When there is only one sandbox, the entire client-configurable circuit is a sandbox and the shell logic includes functionality for controlling user access to it. *See id.* This functionality is part of the bridge on the port because it is the portion of the shell logic that bridges and enables (or disables) communications to the client-configurable circuits, including communications for reconfiguration. *See id.* Moreover, this would have been obvious because Seshadri teaches that the shell logic includes routing functionality with a port for enabling communications to the client-configurable circuits (*supra* §V.A.3[1b]), and that is what the bridging functionality performs when it is locked/unlocked to enable or disable communications for reconfiguring the client-configurable circuits. *See id.* As

explained above, the client-configurable circuit comprises user-definable programmable electronic circuits. *Supra* §V.A.3.[1c].

91. For at least the reasons explained above, Ground 2 also teaches limitation [1d].

4. Independent Claim 10

92. Claim 10 recites many of the same limitations as claim 1. *Infra* Appx-3 (claim appendix showing claims side-by-side). Grounds 1-2 teach the overlapping portions of claim 10 for the same reasons explained above for claim 1. Grounds 1-2 also teach the additional language of claim 10 as explained below.

10[pre]-[a]. A programmable device...
--

93. The limitations of 10[pre] and 10[a] are also recited in 1[pre] and [1a]. Therefore, these limitations are satisfied for the same reasons that I explained above for [1pre] and [1a], respectively. *Supra* §V.A.3.1[pre]-[1a] (incorporated by reference).

[10b] the first region comprising a hardware-based or firmware-based router having a bridge with a port enabling communication between the router and the second region;

94. Limitation [10b] recites similar language to limitation [1b], further specifying the router “having a bridge” with a port. Limitation [1d] recites a

similar limitation: “the port: comprises a bridge.”⁹ Therefore, for both claims, the first region comprises a router with a bridge and port.¹⁰

⁹ I note that Patent Owner relied on the same support for limitations [1b] and [10b] in district court, copying and pasting content from claim 1, as I explained above.

Supra §II.D; *Compare*, Ex-1021, 000079-84, *with*, Ex-1021, 000020-26.

¹⁰ I note that the '943 patent uses router and bridge interchangeably. Figure 3 shows the *router* providing communication between internal and external modules and having an external port and an MMI port. Ex-1001, Fig. 3. But elsewhere, the specification describes the *bridge* as having these features. Ex-1001, 4:52-59 (“The bridge 128 is thus a communication bridge between the outside of the server or system, and internal modules..., and has at least two ports, I/O for external communication and MMI for internal (and possibly also external) communication”). The specification further discloses the bridge comprising a router, instead of the router comprising a bridge: “[a]n FPGA...has a bridge with an MMI router.” Ex-1001, 3:8-12. Independent claim 1 recites a router with a “port compris[ing] a bridge” (limitations [1b], [1d]), while claim 10 recites a router having a “bridge with a port” (limitations [10b]), but the specification depicts the bridge external to the router and port. Ex-1001, Fig. 3.

95. As explained above, Seshadri teaches the first region comprising a hardware-based router with a bridge and port that enables communication between the router and the client configurable circuits. Ex-1004, 6:11-8:61, 14:63-15:8, 12:32-49, Fig. 2; *supra* §V.A.3.[1b] (incorporated here by reference), §V.A.3.[1d] (incorporated here by reference). Therefore, Grounds 1-2 teach limitation 10[b].

[10c] the second region comprising a plurality of sandboxes with user-definable programmable electronic circuits of the PLD; and
--

96. Limitation [10c] recites similar language to limitation [1c] and is satisfied for the same reasons explained previously. *Supra* §V.A.3.[1c] (incorporated by reference). Limitation [10c] additionally requires a plurality of sandboxes instead of just one or more sandboxes as recited in limitation [1c].

Ground 1: Seshadri teaches a plurality of sandboxes.

97. For example, Seshadri teaches sandboxing of multiple users' customer logics in a "multi-tenant cloud environment" (Ex-1004, 2:63-4:2) and refers to "client configurable circuits" in the plural (Ex-1004, 9:48-53, 10:8-14). Figure 1

I further note that Patent Owner alleged that various IC components "separately and together, constitute the claimed 'hardware-based or firmware-based router' and 'port.'" Ex-1021, 000020; *supra* §II.D (discussing Patent Owner's allegations as to limitation [1b]).

shows multiple client virtual machines corresponding to multiple “client configurable circuits.” Seshadri provides for “sandboxing of client configurable circuit 350 so that one client may not obtain secure information associated with the operation of another client's customer logic.” Ex-1004, 10:66-11:16. This makes sense because one users’ customer logic would have to be prevented from interfering with another user’s customer logic for security reasons, and Seshadri encapsulates the client-configurable circuits to “increase the security, performance, privacy, and/or availability of the computing resources” in the FPGA. Ex-1004, 3:41-64; *supra* §I.B (¶12 explaining background knowledge of FPGA security for servers).

Ground 2: Biederman further teaches a plurality of sandboxes.

98. For example, Biederman teaches having “multiple partitions” in the programmable fabric of an FPGA hosting a plurality of logic designs. Ex-1005, ¶129, Fig. 8:

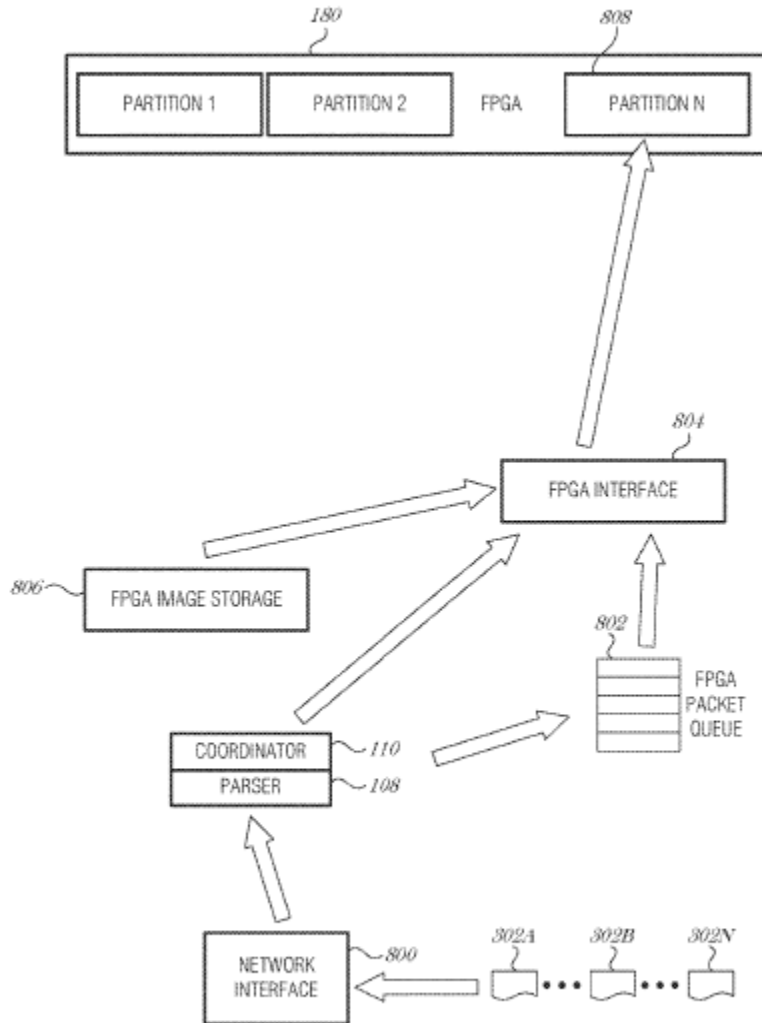


FIG. 8

Similar to functionality in Seshadri's shell logic, Biederman teaches a coordinator circuit that manages the partitions, determining which logic design is to be loaded in each partition. Ex-1005, ¶¶123-128 ("The packet parser 108 or coordinator circuit 110 determines the FPGA program needed to process the packet 302A-N. This information is sent to an FPGA interface controller 804."). "The FPGA

interface controller 804 then loads the program into a partition 808.” Ex-1005, ¶124.

99. Applying Biederman’s teaching of multiple partitions to Seshadri’s teaching of client-configurable circuits, the combination teaches dividing the client-configurable circuit into multiple partitions, each hosting customer logic from different users. *See* Ex-1005, ¶¶123-129, Fig. 8. Based on the combined teachings of the references, a POSITA would have found it obvious to partition the client-configurable circuit to allow multiple tenants to share a single FPGA, each using different areas in the FPGA, because Seshadri already teaches partitioning the configurable areas of a PLD into different areas for different logic circuits (e.g., static shell, reconfigurable shell, client-configurable circuit). Ex-1004, 10:36-49 (“Shell logic 340 and client configurable circuit 350...may be part of *one* or more FPGAs....[S]hell logic 340 and client configurable circuit 350 may be implemented on an FPGA 335....”), 10:7-11. Thus, the use of multiple sandboxes would have been obvious based on Seshadri’s teachings for sandboxing client-configurable circuits in FPGAs for multiple users combined with Biederman’s teaching of partitioning an FPGA.

100. **Motivation to Combine.** A POSITA would have been motivated to combine Seshadri’s teachings of a reconfigurable FPGA with a client-configurable circuit and Biederman’s teachings of partitioning an FPGA fabric because it would

have combined prior art elements according to known methods to yield predictable results. Biederman teaches a known technique to partition the programmable logic blocks of an FPGA. Ex-1005, ¶129. Seshadri teaches a known FPGA configuration where the programmable logic sectors are partitioned into a shell area and a client-configurable circuit area, which can be further partitioned. And it was widely known in the art to partition an FPGA's configurable circuit area into different portions for different users. *Supra* §I.A; Ex-1026, §II.C at 000003; *e.g.*, Ex-1007, §2 at 000003, Ex-1024, §III at 000003-4, Fig. 4, §II.A at 000002; Ex-1023, §3.2 at 000004, Fig. 2 at 000005, Fig. 1 at 000003. Therefore, the combination yields the predictable result of partitioning the client-configurable circuit into distinct areas for customer logic from different users.

101. A POSITA would have been further motivated because the combination provides a number of benefits. By splitting the programmable fabric of an FPGA into smaller partitions, more users can be served with a smaller number of FPGAs, providing substantial cost savings for cloud service providers by reducing the footprint of FPGA-equipped servers. This would also have increased the pool of potential customers by creating lower-cost options, allowing users wishing to test or develop smaller logic designs to share an FPGA, reducing the entry barrier for such users. A POSITA would have also recognized performance improvements from reduced assignment switching of FPGAs, thus

avoiding the overhead of more frequent unloading and reloading of customer logic designs.

102. A POSITA would have had a reasonable expectation of success when combining the teachings of Seshadri and Biederman. The combination would not have changed the principle of operation for either reference. For example, Seshadri's teachings of sandboxing client-configurable circuits in the programmable fabric of an FPGA would still be used to sandbox client-configurable circuits. Biederman's teachings of partitioning the programmable fabric to host different logics, would still be used to create multiple petitions. The operation of the programable logic sectors, interconnects, and other elements of the FPGA in the combination would perform in the same way as they do in Seshadri and in Biederman.

103. Seshadri and Biederman provide complementary teachings for using FPGAs in servers. Ex-1004, 10:36-47, 2:60-3:27, Fig. 3; Ex-1005, ¶¶27-30, ¶50, Fig. 1. The combination adds FPGA-partitioning details from Biederman, which naturally would have been compatible with Seshadri's FPGA teachings, which as explained above already partition the FPGA into several sections. Ex-1005, ¶129, Fig. 8; Ex-1004, 10:36-49, 10:7-14, 9:48-53. A POSITA would have found the combination obvious and would have had a reasonable expectation of success in doing so. Partitioning the programmable logic sectors would be straight forward to

implement because it was already known in the art to partition the programmable fabric in an FPGA. *Supra* §I.A. A POSITA with 2 years of experience in the field would have been fully capable of making the combination. *Supra* §III.

104. For at least the reasons explained above, Grounds 1-2 teach this limitation.

[10d] an internal communication network in the PLD, coupling the port of the bridge of the router and the one or more sandboxes for the data flow processing,

105. As I explained above, Seshadri’s shell logic (including the management, data path, and mailbox functionality) comprises a router having a bridge with a port.¹¹ *Supra* §V.A.4.[10b], §V.A.3.[1b], §V.A.3.[1d]. I also explained above how Seshadri teaches loading logic for data flow processing on the FPGA. *Supra* §V.A.3.1[pre]. As I explained above, the user’s customer logic is loaded to the sandboxes in the client-configurable circuits (*supra* §V.A.3.[1c]), and therefore the sandboxes are for the data flow processing.

¹¹ The ’943 patent uses router and bridge interchangeably. *Supra* n. 10. In district court, Patent Owner alleged various functional blocks “*separately and together, constitute*” the claimed router and port. *Supra* n. 10 (citing Ex-1021, 000020; *supra* §II.D).

106. Seshadri further teaches **an internal communication network in the PLD, coupling the port of the bridge of the router and the one or more sandboxes** (e.g., client-configurable circuits). For example, Figure 2 shows the data path serving as a network for internal communication between various modules within the FPGA, including the management module 214 and peripheral interface 212 in the shell logic and client-configurable circuit 240. Ex-1004, Fig. 2 (highlighted):

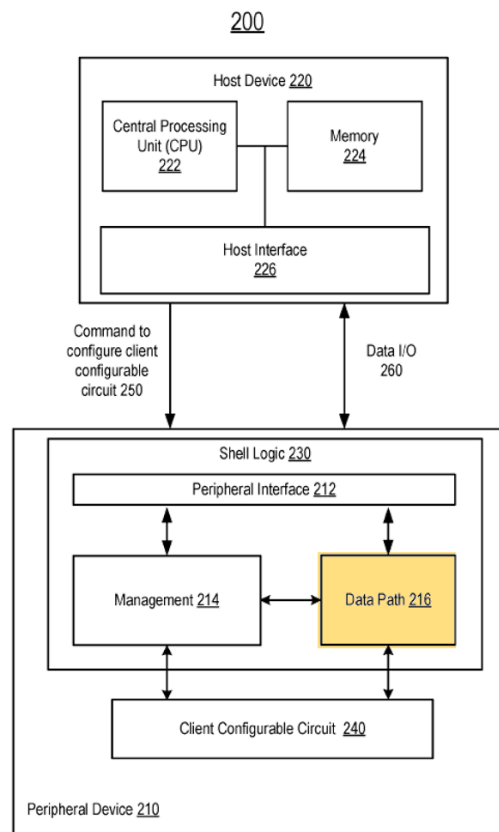


FIG. 2

As illustrated by the arrows to management 214, peripheral interface 212, and

client-configurable circuit 240, the data path module 216 couples the router in the shell logic (including the port and bridge) with the one or more sandboxes in the client-configurable circuit. The port of the bridge of the router includes the portions of shell logic that enable communication with the client-configurable circuit region. *Supra* §V.A.3[1b], §V.A.4[10b]. The data-path couples the port to the client-configurable circuits. Ex-1004, Fig. 2. *Id.*

107. Figure 2 depicts the data path as a connection fabric (a path for data), and Seshadri teaches that a fabric is a network, explaining that, “[i]n the context of a computer network, terms ‘fabric’ and ‘network’ may be used interchangeably herein.” Ex-1004, 26:1-3. The data path is a network that connects internal FPGA components, including peripheral interface 212, management 214, and client-configurable circuit 240, for communication between those modules in the FPGA. FPGAs and other PLDs comprise interconnect resources that serve as a fabric for relaying internal communication from one part of the IC to another; this was well-known in the art and a basic concept in FPGAs. *Supra* §I.A; *see, e.g.*, Ex-1010, 000004, 000176-177; Ex-1012, 000022; Ex-1006, ¶30, ¶33-34; Ex-1008, 1:22-29; Ex-1011, 000003-4; Ex-1009, ¶41-42. Thus, a POSITA with experience in FPGAs would have understood Seshadri’s teachings regarding a data path to encompass the interconnect resources that form an internal communication network of the FPGA.

108. For at least the reasons explained above, Grounds 1-2 teach this limitation.

[10e] wherein the bridge is lockable to prevent user access, and unlockable to enable user access, to the plurality of sandboxes and user-definable programmable electronic circuits therein.

109. Limitation [10e] recites similar language to limitation [1d] and is satisfied for the same reasons explained previously. *Supra* §V.A.3.[1d] (incorporated by reference). Seshadri further teaches the additional limitations of [10e], as explained below.¹²

110. Grounds 1-2 each teaches **a plurality of sandboxes** as I explained above for limitation [10c]. Ex-1004, 2:63-4:2, 10:36-49; Ex-1005, ¶¶123-129, Fig. 8; *supra* §V.A.4.[10c] (incorporated by reference). Seshadri's teachings apply equally to locking and unlocking a single sandboxed client-configurable circuit or multiple sandboxed partitions in the client-configurable region of the FPGA's programmable fabric. Seshadri contemplates the existence of multiple sandboxed client-configurable circuit areas in its cloud platform, as well as dividing their use between multiple users in a "multi-tenant cloud environment." Ex-1004, 2:63-4:2,

¹² I note that Patent Owner relied on the same support for limitations [1d] and [10e] in district court, copying and pasting content from claim 1, as I explained above. *Supra* §II.D; *Compare*, Ex-1021, 000099-108, *with*, Ex-1021, 000032-41.

Figs. 8, 1. Seshadri teaches checking user account permissions to lock or unlock the sandboxed client-configurable circuits for this reason. Therefore, the same teachings that I explained above for limitation [1d] equally apply to a plurality of sandboxes on the same FPGA.

111. For at least the reasons explained above, Grounds 1-2 teach this limitation.

5. Dependent Claim 2

2. The programmable device of claim 1, further comprising: the router communicative to translate between a first protocol and a second protocol, for messaging with system resources.

112. As I explained above for claims 1 and 10, Grounds 1-2 teach a PLD with a first region (e.g., “shell logic” area) comprising a router with a port and bridge, and a second region (e.g., “client configurable circuit” area) with sandboxes (e.g., “client configurable circuits”), where the port and bridge are lockable/unlockable to deny/allow user access to the sandboxes. *Supra* §§V.A.3-4. Seshadri teaches the additional limitations of claims 2 and 11.

113. Seshadri teaches **the router being communicative to translate between a first protocol and a second protocol, for messaging with system resources**. For example, Seshadri’s router functionality in the shell logic comprises a mailbox and data path that enables communications between the client-configurable circuit area and the PCIe interface, translating between the

FPGA's internal format and the PCIe bus protocol, which leads to the rest of the computer system. Ex-1004, 8:37-61 (“[D]ata path module 216 can be interposed between client configurable circuit 240 and the physical interconnect (e.g., the PCIe bus).”), Abstract (“[T]he shell logic also includes a mailbox configured to enable communication between the client virtual machine and a management virtual machine”). Seshadri explains the data path module functions “as a translation layer between peripheral interface 212 and client configurable circuit 240.” Ex-1004, 8:37-61. “[D]ata path module 216 may include an interface for receiving information from client configurable circuit 240, and data path module 216 may format the information for transmission from peripheral interface 212. Formatting the information can include generating control information for one or more commands and partitioning data into blocks that are sized to meet protocol specifications.” *Id.*

114. When messages are sent from the client-configurable circuits to the rest of the system, including the host processor and storage, this functionality in the shell logic translates between a first protocol used by the peripheral interface (e.g., PCIe) and the FPGA's internal protocol. The FPGA's internal communications are translated (e.g., using the data path module and mailboxes) into PCIe protocol “packets” or “bus transactions” for the protocol used on the local interconnect bus: “[t]he shell logic may also be responsible for forming packets or bus transactions

on the local interconnect [(PCIe interface)] and ensuring that certain protocol requirements are met.” Ex-1004, 6:11-41, 9:12-22 (“a mailbox function through which a client virtual machine can communicate with a management virtual machine (or a management hypervisor)”). PCIe is a protocol designed for connecting different chips and other components of desktop, laptop, and server computers. PCIe is a serial protocol with fewer lines to minimize routing space on printed circuit boards; it was not designed or used for internal chip communications, which typically used parallel buses. For this reason, the PCIe standard specifies that the physical layer implementation includes circuitry for “parallel-to-serial and serial-to-parallel conversion” and is responsible for translating between protocols (an implementation-specific format and the PCIe serialized format). Ex-1033, 000037 (“The Physical Layer exchanges information with the Data Link Layer in an implementation-specific format. This layer is responsible for converting information received from Data Link Layer in to an appropriate serialized format and transmitting it across the PCI Express Link at a frequency and width compatible with the remote device.”).

115. Seshadri teaches an FPGA with various internal components that communicate through internal buses. *E.g.*, Ex-1004, 14:54-63, Fig. 4 (“shell private fabric 470”). FPGAs included internal buses and fabrics to allow different portions of the FPGA to communicate with each other; this was a basic aspect of

IC design and had been well-known for FPGAs for decades. *Supra* §I.A; Ex-1010, 000004, 000176-177; Ex-1012, 000022; Ex-1006, ¶30, ¶33-34; Ex-1008, 1:22-29; Ex-1011, 000003-4; Ex-1009, ¶41-42. Here, Seshadri explains that the data path module translates between the protocol used by the FPGA's physical interface (e.g., PCIe) and the internal protocol used by the client-configurable circuits. Ex-1004, 8:37-61. Therefore, Seshadri teaches that the router functionality of the shell logic translates between a first protocol and a second protocol.

116. Seshadri teaches the router being communicative to translate between a first protocol and a second protocol, for messaging with system resources external to the FPGA. For example, Seshadri further teaches the distributed computing platform includes various system resources, such as “host systems, processor nodes, storage subsystems.” Ex-1004, 26:4-8, Fig. 8. Figure 7 illustrates other types of resources available on the host server device, such as “processing logic 702, a configuration module 704, a management module 706, a bus interface module 708, memory 710, and a network interface module 712.” Ex-1004, 23:24-42. Additionally, Seshadri's FPGAs are meant to allow “[c]lients [to] access the networked computing resources via a network, such as the Internet.” Ex-1004, 1:5-32. Indeed, Seshadri teaches the nodes hosting the customer logics on the FPGAs are connected over a network, including switches and routers. Ex-1004, 26:36-61.

117. For at least the reasons explained above, Ground 2 also teaches claim 2.

6. Dependent Claim 11

11. The programmable device of claim 10, further comprising: the router to translate between a first protocol used external to the second region and a second protocol used in the internal communication network in the PLD.

118. Claim 11 recites overlapping limitations with claim 2, which are satisfied for the same reasons explained above. *Supra* §V.A.5. Seshadri further teaches the additional language of claim 11. *Infra* Appx-3 (claim appendix showing claims side-by-side).

119. Seshadri teaches **a first protocol used external to the second region and a second protocol used in the internal communication network in the PLD**. As explained above for claim 2, the first protocol is used by the peripheral interface (e.g., PCIe); it is used for communications over the external PCIe interface. *See* Ex-1004, Fig. 4 (arrows between “PCIe RC 430” and “PCIe EP 450”). Therefore, it is external to the second region (the client-configurable circuits). The second protocol is the internal protocol used by the FPGA; Seshadri teaches that the “[d]ata path module 216 may also act as a *translation layer* between peripheral interface 212 and client configurable circuit 240.” Ex-1004, 8:37-61. Seshadri teaches translation between the internal protocol used by the client-configurable circuit (second region) and the first protocol, PCIe, for external

communications: “Specifically, data path module 216 may include an interface for receiving information from client configurable circuit 240, and data path module 216 may format the information for transmission from peripheral interface 212.”

Id. “Formatting the information can include generating control information for one or more commands and partitioning data into blocks that are sized to meet protocol specifications.” *Id.* Indeed, FPGAs include internal buses and fabrics (e.g., interconnects) to allow the different portions of the FPGA to communicate with each other; this had been well-known for FPGAs for decades. *Supra* §I.A; Ex-1010, 000004, 000176-177; Ex-1012, 000022; Ex-1006, ¶30, ¶33-34; Ex-1008, 1:22-29; Ex-1011, 000003-4; Ex-1009, ¶41-42.

120. For at least the reasons explained above, Ground 2 also teaches claim 11.

7. Dependent Claim 3

3. The programmable device of claim 1, wherein each of the sandboxes comprises user-programmable one or more of: analog circuitry, digital circuitry, memory, processors or controllers.
--

121. As I explained above for claims 1 and 10, Grounds 1-2 teach a PLD with sandboxes (e.g., client-configurable circuits). *Supra* §V.A.3.[1c], §V.A.4.[10c]. Seshadri further teaches the additional limitations of claim 3. Because claim 3 recites “one or more of” several options, it is my understanding that it is satisfied if any one of the listed elements is present.

122. Seshadri teaches **wherein each of the sandboxes comprises user-programmable digital circuitry**. For example, Seshadri explains that the client-configurable circuit area comprises “hardware that can be programmed or configured to perform a logic function specified by configuration data that is applied to or loaded on the configurable circuit.” Ex-1004, 2:60-3:11; *see also supra* §V.A.3.[1c] (explaining how the client-configurable circuits are user-definable programmable electronic circuits) (incorporated by reference). As explained above for limitation [1a], these user-programmable circuits are located on “programmable integrated circuits (ICs), such as field programmable gate arrays (FPGAs) or programmable array logics (PALs).” Ex-1004, 10:36-47. Therefore, Seshadri teaches that each sandbox comprises digital “logic” in an FPGA that is programmed by the user. *Id.*, 6:48-51, 7:22-36, 2:60-3:11, 3:65-4:9; *see also supra* §I.A. To the extent Patent Owner argues that this circuitry is somehow not digital, then the circuitry would be analog, which also satisfies this claim limitation.

123. Seshadri gives examples of the programmable digital circuitry in the client-configurable circuit. Ex-1004, 17:16-36:

Client configurable circuit 595 may include circuits that can be configured or programmed into the customer logic to perform a given function. For example, client configurable circuit 595 may include programmable logic blocks (e.g., FPGAs or PALs) comprising

combinational logic and/or LUTs and sequential logic elements (such as flip-flops and/or latches), programmable routing and clocking resources, programmable distributed and block RAMs, digital signal processing (DSP) bit slices, and/or programmable input/output pins.

124. For at least the reasons explained above, Ground 2 also teaches claim 3.

8. Dependent Claim 12

12. The programmable device of claim 10, wherein each of the plurality of sandboxes comprises user-programmable one or more of: analog circuitry, digital circuitry, memory, processors or controllers.
--

125. Claim 12 recites overlapping limitations with claim 3, which are satisfied for the same reasons explained above. *Supra* §V.A.7 (incorporated here by reference). Additionally, claim 12 recites “each of the *plurality* of sandboxes.” *Infra* Appx-3 (claim appendix showing claims side-by-side). Grounds 1-2 teach a plurality of sandboxes in the client-configurable circuits, as explained for claim 10, which comprise digital or analog circuitry for the same reasons explained for claim 3. *Supra* §V.A.4[10c] (incorporated here by reference).

9. Dependent Claims 4 and 13

- 4. The programmable device of claim 1, wherein the port comprises: a bridge being lockable to prevent reconfiguration of the bridge and prevent reconfiguration of the router while enabling reconfiguration of the one or more sandboxes.**
- 13. The programmable device of claim 10, wherein the bridge is lockable to enable reconfiguration of the plurality of sandboxes, prevent reconfiguration of the bridge and prevent reconfiguration of the router.**

126. As an initial matter, as I explained above for claims 1 and 10, Grounds 1-2 teach a PLD with a first region (e.g., “shell logic” area) comprising a router with a port and bridge, and a second region (e.g., “client configurable circuit” area) with sandboxes (e.g., “client configurable circuits”). *Supra* §§V.A.3-4. The port and bridge functionality in the first region are lockable/unlockable to deny/allow user access to the sandboxes. *Supra* §V.A.3[1d]. Seshadri further teaches the additional limitations of claims 4 and 13, which recite substantively similar language. *Infra* Appx-3 (claim appendix showing claims side-by-side).

127. Seshadri teaches the port comprising a bridge that prevents reconfiguration of the bridge and router¹³ functionality in the static shell logic

¹³ The router comprises a bridge for both claims. I also note that the '943 patent uses router and bridge interchangeably. Ex-1001, 4:52-59, 3:8-12, Fig. 3; *supra* n. 10.

while enabling reconfiguration of the sandboxes (e.g., client-configurable circuit). For reconfiguration, Seshadri describes how a configuration request is received, the corresponding user logic bitstream is downloaded, and the functionality in the shell logic controls configuration of the client-configurable circuit area. Ex-1004, 12:10-31:

Client-configurable circuit 350 may be reconfigured and controlled by a client associated with a client virtual machine 330 running on host 310. A client may upload a configuration file (e.g., firmware, hardware description language (HDL) file, bit streams, etc.) with customer defined logic repository service 312 for use by computing system 300. Through management virtual machine 332, the customer defined logic can be downloaded from logic repository service 312 and sent to peripheral device 315 for programming or configuring client configurable circuit 350. *The programming of client configurable circuit 350 may be controlled and managed by shell logic 340.*

See also Ex-1004, 20:46-21:14, 12:10-31, 11:16-22, 18:35-49, 20:21-22:67 (outlining programming details), Fig. 6. This functionality was explained previously. *Supra* §V.A.3[1d] (incorporated here by reference).

128. At other times, the shell logic leaves the configuration access port locked (“isolated”) so that the client-configurable circuit cannot be reconfigured. Ex-1004, 6:11-41. Thus, the shell logic unlocks the configuration access port to allow reconfiguration of the client-configurable circuits. *See id.* In addition to

Seshadri's teachings, this would have been obvious because the FPGA cannot be reconfigured when the access port is locked. *See id.*

129. During reprogramming of the client-configurable circuit, the router, bridge, and port functionality (including but not limited to the mailbox and datapath modules that bridge communications) is in the static shell. Ex-1004, 4:48-5:3:

When the reconfigurable shell is reconfigured during shell switching, *a circuit in the static shell may handle requests from the client virtual machine or a host, such as packet processing or memory access requests*, thus enabling the client virtual machine to be active during shell switching. *The static shell may also include circuits for isolating the reconfigurable shell from the client virtual machine, the static shell, and other circuits of the computing system (e.g., the host) and for bridging the communications between the client virtual machine and a management virtual machine on the host.*

See also Ex-1004, 18:4-24 (The static shell “*handle[s] signaling and packet processing during the reconfiguration of reconfigurable shell 590*.” For example, static shell 572 may include *a bus interface 574 for communicating with client virtual machine 520 and management virtual machine 530 on host 510*; a reconfiguration logic 586 for downloading shell configuration data and configuring the reconfigurable shell using the shell configuration data; *mailbox 588 for bridging the communications between management virtual machine 530 and client*

virtual machine 520...”), 18:25-33 (similar). “Packet processing logic 580 may be used to handle accesses (e.g., memory accesses) to peripheral device 560 from host 510 when reconfigurable shell 590 is isolated from static shell 572.” Ex-1004, 18:63-19:22.

130. The shell logic includes a static portion and a reconfigurable portion. The static shell is the portion that controls access to the configuration access port because the reconfigurable shell is disabled and isolated during reconfiguration of the client-configurable circuits, which can include reconfiguration of the reconfigurable shell. Ex-1004, 18:49-62 (“When activated, isolation logic 578 may gate the control or data signal to and from reconfigurable shell 590, or disconnect or block the interfaces between static shell 572 and reconfigurable shell 590, before reconfigurable shell 590 is reconfigured.”), 18:4-24 (“For example, static shell 572 may include...an isolation block 576 that can be activated to isolate reconfigurable shell 590 from static shell 572....”). Seshadri provides additional details of how the reconfigurable shell is isolated. Ex-1004, 19:23-63. Therefore, while the client-configurable circuits are being reconfigured, the shell logic functionality—including bridging functionality—is in the static portion of the shell, so that the shell logic can continue to operate and control the FPGA configuration process while reconfigurable portions are being changed. Ex-1004, 18:4-19:63.

131. As Seshadri explains, during reprogramming, the reconfigurable shell sends a signal to activate reconfiguration logic in the static shell “that is responsible for handling reconfigurable shell switching.” Ex-1004, 17:58-18:3, Fig. 5 (“Reconfiguration Logic 586”, highlighted):

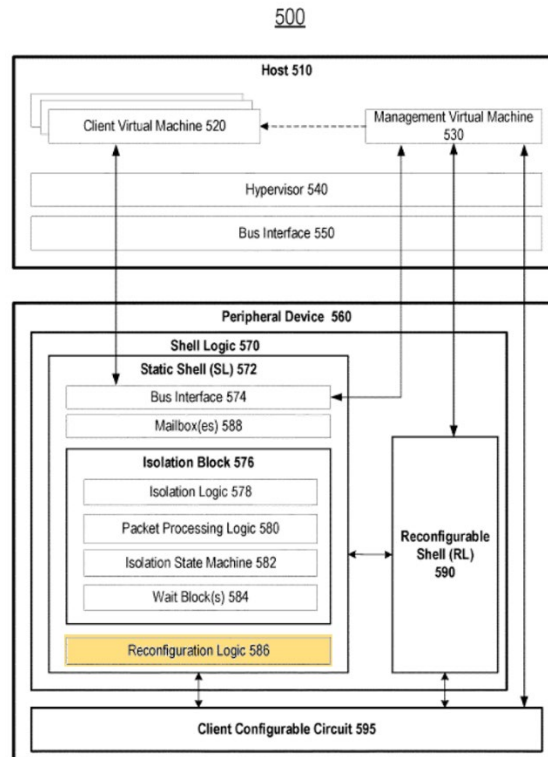


FIG. 5

Then, “the reconfiguration logic [] load[s] the shell configuration data (e.g., bit stream) for the new logic circuit for the reconfiguration shell to the programmable circuits allocated for the reconfigurable shell to configure the reconfigurable shell using the shell configuration data.” Ex-1004, 22:51-22:55.

132. Seshadri prevents reconfiguration of the bridge and router, which are in the static shell, while the client-configurable circuit area is reconfigured. As

Seshadri explains, the static shell can be unlocked for reconfiguration if an external reset or power cycle occurs, but is otherwise locked from reconfiguration during operation. Ex-1004, 17:37-57 (“Static shell 572 may include non-programmable circuits or programmable circuits that may be configured or initialized during a boot-up sequence, but may not be reconfigured during the operations of shell logic 570.”). While the static shell is implemented on a PLD (Ex-1004, 11:13-40), “[s]tatic shell 342 may not be modified unless an externally controlled reset signal or a power cycle occurs.” Ex-1004, 11:13-40. Therefore, the static shell, including the router, bridge, and port functions discussed above, is not allowed to be reconfigured when the client-configurable circuit and reconfigurable shell are being reconfigured. Ex-1004, 4:48-5:3 (“In some implementations, *the shell logic may include a static shell (SL) that may not change during shell switching*, and a reconfigurable shell (RL) that may be reconfigured during shell switching.”).

133. Claim 13 additionally requires **a plurality of sandboxes** instead of just one or more sandboxes as recited in claim 4. *Infra* Appx-3 (claim appendix showing claims side-by-side). I explained above how Grounds 1-2 each teach a plurality of sandboxes for limitation [10c]. *Supra* §V.A.4.[10c] (incorporated here by reference). The teachings of Seshadri that I just explained for claim 4 equally apply to the case where the client-configurable circuit area is divided into a

plurality of partitions, because the static shell would still control reconfiguration of the entire client-configurable circuit region as explained above.

134. For at least the reasons explained above, Grounds 1-2 teach claims 4 and 13.

10. Dependent Claim 5

5. The programmable device of claim 1, wherein the port comprises: a bridge having a security layer to authenticate a user or verify a permission, to reconfigure one or more of the one or more sandboxes.

135. As I explained above for claims 1 and 10, Seshadri teaches a PLD with a shell logic area comprising a router, a client-configurable circuit area, and a port with a bridge for locking/unlocking user access to sandboxes (e.g., client-configurable circuits). *Supra* §§V.A.3-4. It is my opinion that Seshadri teaches the additional limitations of claim 5.

136. Seshadri teaches **wherein the port comprises: a bridge having a security layer to authenticate a user or verify a permission, to reconfigure one or more of the one or more sandboxes**. For example, Seshadri discloses the shell logic carries out “security functionality.” Ex-1004, 10:66-11:16 (“Shell logic 340 on peripheral device 315 may perform security functionality....”). Specifically, the functionality in the shell logic verifies that the user’s customer account has permission to reconfigure the client-configurable circuits. Ex-1004, 9:48-10:15:

Thus, in some computing systems with client configurable circuits in a peripheral device (i.e., end point), in order to protect the host during a transition that replaces the first customer logic with the second customer logic, the following operations may be performed in response to a request from a client virtual machine to load a new customer logic. *First, the customer account permissions can be checked.*

After authenticating the user and checking the reconfigurable shell is ready, “[a] response can be generated to inform the client virtual machine that the load request is accepted.” Ex-1004, 9:48-10:15.

137. After the customer account permissions are verified, the shell logic then proceeds to reconfigure the requested one or more sandboxes (e.g., client-configurable circuits) by unlocking access to the configuration access port. *See* Ex-1004, 6:11-41 (“[T]he shell logic may isolate a configuration access port so that the customer logic cannot cause the client configurable circuit to be reprogrammed without using services provided by the computing service provider (e.g., the shell logic).”), 12:10-31 (“The programming of client configurable circuit 350 may be controlled and managed by shell logic 340.”). I explained Seshadri’s teachings of locking/unlocking and reconfiguring the client-configurable circuit above for limitation [1d] and claims 4/13. *Supra* §V.A.3.[1d], §V.A.9 (analysis incorporated by reference here).

11. Dependent Claim 14

14. The programmable device of claim 10, wherein the bridge has a security layer to authenticate a user or verify a permission, to reconfigure one or more of the plurality of sandboxes.

138. Claim 14 recites overlapping limitations with claim 5, which are satisfied for the same reasons explained above. *Supra* §V.A.10 (incorporated by reference).

139. Grounds 1-2 further teach the additional language of claim 14, which recites a **plurality of sandboxes** (*infra* Appx-3 (claim appendix showing claims side-by-side)). As explained above, Grounds 1-2 teach a plurality of sandboxes. *Supra* §V.A.4.[10c] (incorporated here by reference). The security layer was explained for claim 5, and the same analysis applies here because both claims require the security layer to authenticate a user or verify a permission to reconfigure “one or more” of the sandboxes.

12. Dependent Claims 9 and 18

9. The programmable device of claim 1, wherein the one or more sandboxes comprises a plurality of sandboxes that are logically or physically combinable to form a larger sandbox.

18. The programmable device of claim 10, wherein the plurality of sandboxes is logically or physically combinable to form a single, larger sandbox.

140. Claims 9 and 18 respectively depend from claims 1 and 10, and recite substantively similar language. Additionally, claim 18 recites a *single*, larger

sandbox. *Infra* Appx-3 (claim appendix showing claims side-by-side). Ground 2 teaches the additional limitations of claims 9 and 18. As I explained above for claims 1 and 10, a POSITA would have been motivated to combine Biederman's teachings of adjusting partition sizes between multiple partitions with Seshadri's teachings of client-configurable circuits. I explain this in more detail below.

Ground 2: Biederman teaches the additional limitation.

141. Biederman teaches **wherein the one or more sandboxes comprises a plurality of sandboxes that are physically combinable to form a single, larger sandbox**. For example, Biederman teaches reconfigurable logic in partitions (e.g., portions) of an FPGA. Ex-1005, ¶59 (“The coordinator circuit 110 may interface with an FPGA 180 to determine its status, request a reconfiguration (or partial reconfiguration), or provide data for the FPGA 180 to act upon. A partial reconfiguration is when a smaller portion of the FPGA 180 is reconfigured, as opposed to the entire FPGA 180.”). The size of the partitions is adjustable. Ex-1005, ¶134 (“Additionally, based on priority, the coordinator circuit 110 may increase or decrease the amount of...*partition* available for instructions or programs^[14].”). As I explained above for limitation [10c], Biederman teaches

¹⁴ Biederman refers to logic designs for FPGAs as “FPGA programs.” *E.g.*, Ex-1005, ¶128.

hosting a plurality of logic designs on “multiple partitions” in the programmable fabric of an FPGA. Ex-1005, ¶129, Fig. 8 (partitions 1-N in FPGA 180):

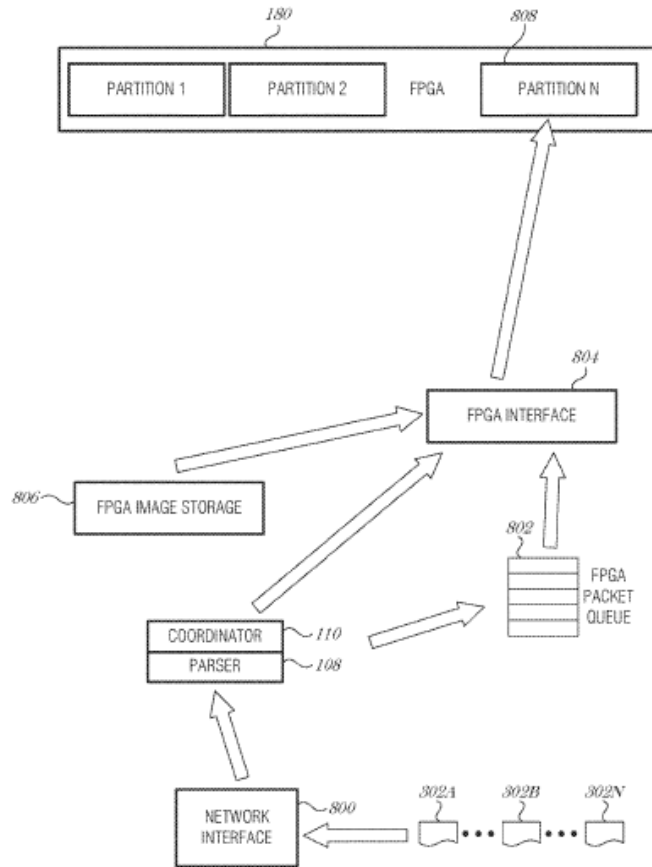


FIG. 8

142. The coordinator circuit manages the partitions, deciding which logic design to load to each partition and instructing the FPGA interface controller. Ex-1005, ¶¶123-126, ¶128 (“[T]he coordinator circuit 110 may use an FPGA interface controller 804 that is connected to FPGAs on the host system (or internal to the NID 102) and is operable to track FPGA programs in the FPGAs and load or unload FPGA programs in a *partition* that is controlled by the FPGA interface

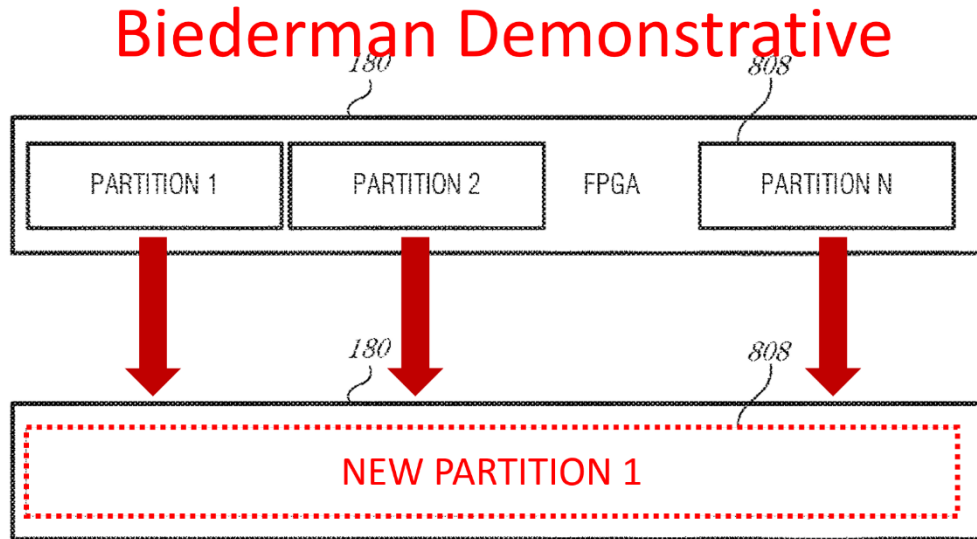
controller 804.’’). Biederman adjusts the partition size to match a logic design that is chosen based on the priority of tasks assigned to the partition, choosing larger or smaller logic designs to meet the processing needs of a task. Ex-1005, ¶134.

Biederman does this because “[u]sing a larger cache/partition space may act to reduce further cache/FPGA misses for complex processing, thereby ensuring faster processing and a better user experience.” *Id.* For example, “[h]igh-priority programs, such as a video conferences, may use complex processes to ensure real-time or near real-time audio and video presentation.” *Id.* Therefore, they would require more space (e.g., a larger partition with more programmable logic sectors) to load the user’s custom logic.

143. Biederman illustrates the operation of the coordinator circuit and FPGA partitions with an example of processing network packets. Ex-1005, ¶¶123-129, Fig. 8. Incoming packets are “parsed to determine a corresponding program to be run by the FPGA” and “[t]he packet parser 108 or coordinator circuit 110 determines the FPGA program needed to process [each of] the packet[s] 302A-N.” Ex-1005, ¶123. While the incoming packets are being processed, the logic designs on the partitions P1-PN are continuously being adapted to the packet data to be processed. Ex-1005, ¶¶130-131 (“Packets are received (operation 902) and then parsed to determine the best program (operation 904). The parsing (operation 904) may identify a set of instructions to load into a cache or a program to load into an

FPGA, for example”), Fig. 9. Logic designs that are not being used to process current packets are evicted and the space in the partition is reclaimed and reconfigured. Ex-1005, ¶129 (“In fact, once P1 is done processing, it may be reprogrammed to process a different data set.”).

144. When smaller partitions are evicted to make room for a larger logic design (Ex-1005, ¶¶129-31), the smaller partitions are combined to form a single, larger sandbox partition. This occurs because the multiple partitions divide up the limited amount of total space for customer logic on the FPGA. I have prepared the following demonstrative, based on the partitions shown in Biederman’s Figure 8, diagram to illustrate this concept:



For example, when the smaller logic in partitions 1 through N are replaced by a larger video transcoding logic—which would occur when the next packet to be processed by partition 1 requires heavy processing for video transcoding—

partitions 1 and 2, along with potentially other partitions, are combined into a single, larger partition to accommodate the resized partition 1. *See* Ex-1005, ¶134. Therefore, Biederman teaches physically combining partitions of an FPGA into a single, larger partition.

145. **Motivation to Combine.** Seshadri teaches an FPGA with client-configurable circuits, for use in a multi-user computing environment. Ex-1004, 3:41-64, 2:63-4:2, 2:50-3:11; *supra* §§V.A.3-4. Biederman teaches partitioning the programmable fabric of an FPGA and dynamically adjusting the partition sizes. A POSITA would have been motivated to combine Seshadri's teachings of a reconfigurable FPGA with a partitioned client-configurable circuit and Biederman's teachings of adjusting partition sizes because it would be combining prior art elements according to known methods to yield predictable results. Biederman teaches a known technique to partition the programmable logic blocks of an FPGA and to adjust partition size. Ex-1005, ¶134. Seshadri teaches a known configuration of an FPGA where the programmable logic sectors are partitioned into a shell area and a client-configurable circuit area, which can be further partitioned. The combination yields the predictable results of adjusting partition sizes to fit the logic loaded on the partition, as taught by Biederman.

146. A POSITA would additionally have been motivated because the combination provides a number of benefits. By sizing partitions to the logic to be

hosted, it adds flexibility to the partitioning of logic sectors in the FPGA. This allows for users to use larger logic designs, increasing user choice and satisfaction. This also increases utilization of client-configurable circuit sectors in the FPGA, which provides a benefit to cloud service providers and customers by allowing more customers to be served with a smaller amount of hardware. This reduces the cost of equipping, maintaining, and running data centers with FPGA-equipped servers. See Ex-1026, 000002-4 (discussing efficiency of sharing resources for cloud computing). Lowering costs and reducing the footprint of FPGA-equipped servers has the additional benefit of expanding the pool of potential customers increases because it creates lower-cost options and allows users developing smaller logic circuits to share an FPGA, reducing the entry barrier.

147. A POSITA would have had a reasonable expectation of success when combining the complementary teachings of Seshadri and Biederman. The combination would not have changed the principles of operation. For example, Seshadri's teachings of partitioning and reconfiguring client-configurable circuits in the programmable logic of an FPGA would still be used to partition and reconfigure client-configurable circuits. Biederman's teachings of adjusting partition sizes, including combining partitions, would still be used to adjust the size of partitions. The operation of the programable logic sectors, interconnects, and

other elements of the FPGA in the combination would perform in the same way as they do in Seshadri and in Biederman.

148. Seshadri and Biederman teach complementary teachings related to the composition and operation of FPGAs. The combination adds a few minor partitioning details from Biederman that are compatible with Seshadri's teachings. A POSITA would have found the combination obvious and would have had a reasonable expectation of success in doing so. Configuring a partition on logic sectors from more than one previous partitions would be straight forward to implement because it was already known in the art to reprogram portions of the programmable fabric in an FPGA—reprogrammability was a central feature of FPGAs. A POSITA with 2 years of experience in the field would have been fully capable of utilizing partitions on an FPGA, consistent with making the combination. *Supra* §III.

B. Grounds 3-4: the Challenged Claims are Obvious based on Atsatt-113 alone (Ground 3) and Atsatt-113 in view of Biederman (Ground 4).

149. It is my opinion that a POSITA would have found the Challenged Claims obvious based on Atsatt-113 (Ground 3). Biederman (*supra* §V.A.2) reinforces Atsatt-113's PLD teachings by explaining how PLDs were used for data flow architectures and providing additional details for dependent claims 9 and 18,

including techniques for resizing and combining partitions within the PLD (Ground 4).

150. A POSITA would have been motivated to combine Atsatt-113 with Biederman for several reasons. First, Atsatt-113 teaches the use of PLDs, including FPGAs, in a server environment, where the PLD is partitioned between multiple users. Biederman teaches additional details and use cases for a server with PLDs. *Infra* §V.B.3.10[pre]. Second, Atsatt-113 and Biederman both come from Intel and identify Intel as the applicant. Third, both use FPGAs to accelerate data processing for multiple users in the context of virtual machines. *Infra* §V.B.1, §V.A.2. I explain the motivation to combine and reasonable expectation of success in further detail for the individual limitations below. *Infra* §V.B.2.1[pre] (Motivation to Combine subsection), §V.B.8 (Motivation to Combine subsection).

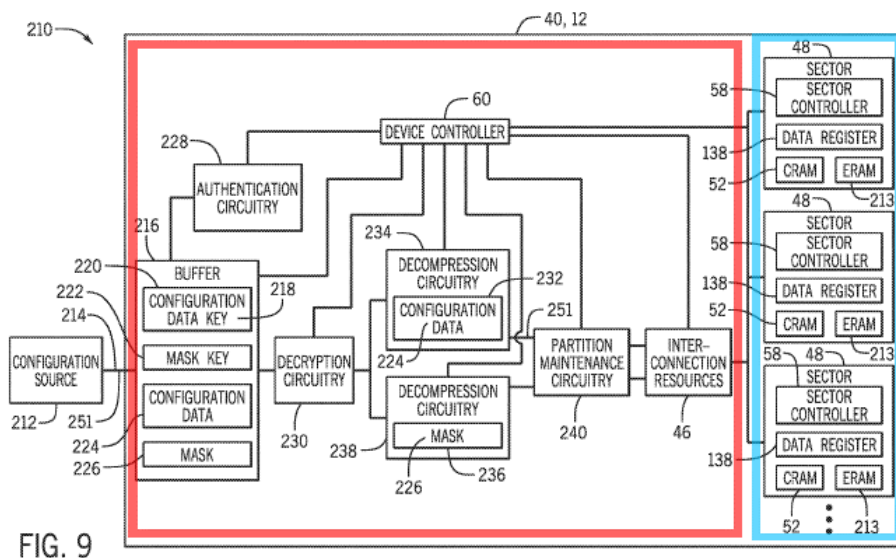
1. U.S. Patent Application Publication No. 2019/0095113A1 (“Atsatt-113”) (Ex-1006)

151. I have reviewed the Atsatt-113 reference. I understand that Atsatt-113 was not cited or considered during prosecution of the '943 Patent, based primarily on the fact that Atsatt-113 is not cited on the face of the patent, nor have I seen the reference discussed in the prosecution history.

152. Atsatt-113 is entitled “Maintaining reconfigurable partitions in a programmable device.” Ex-1006, (54) Title. Atsatt-113 was filed on September 28, 2017. Based on the priority date of July 15, 2018 of the '943 patent, I

understand Atsatt-113 is prior art to the '943 patent under at least post-AIA §102(a)(2).

153. Atsatt-113 is directed to “programmable logic devices” that are reconfigurable. Ex-1006, ¶27. Specifically, Atsatt-113 discloses “systems, methods, and devices for maintaining reconfigurable partitions in an integrated circuit (e.g., a programmable device such as an FPGA).” Ex-1006, ¶5. Atsatt-113 discloses configurations that include user programmable logic sectors on one hand, and a control area that includes various control circuitry and interconnection resources on the other. Ex-1006, ¶¶30-31, ¶39, Fig. 9 (annotated: control area surrounded in red to the left; programmable logic sectors surrounded in blue to the right):



154. Atsatt-113’s programmable logic sectors are divided into partitions that are assigned to different users and can be reconfigured to load different logic

during operation. Ex-1006, ¶¶40, ¶5, ¶3 (“situations where multiple users share a common device but distinct portions of the device”). Atsatt-113 provides security functions that “prevent partitions from utilizing resources that do not belong to them” using masks that specify permissions to access logic sectors and resources. Ex-1006, ¶¶41-42. Before applying masks (for example to reconfigure user partition), Atsatt-113 performs key-based authentication to lock and unlock the applicable masks. Ex-1006, ¶¶59-62, ¶¶66-69.

155. Atsatt-113 is analogous art to the '943 patent because it is in the same field of endeavor as the '943 patent, which is PLDs and data processing systems. *Supra* §III. Atsatt-113’s “disclosure relates generally to configuration of programmable devices.” Ex-1006, ¶1. Atsatt-113’s programmable devices are configured to carry out user-defined data processing in “a server having multiple virtual machines.” Ex-1006, ¶5. Atsatt-113 is reasonably pertinent to the problem addressed by the '943 patent’s inventor, because it directly addresses that problem in like manner to the '943 patent: Atsatt-113 presents a flexible solution to reconfigure partitions within a field programmable logic device. Ex-1006, (57) Abstract.

2. Independent Claim 1

1[pre]. A programmable device for data flow processing in a user-configurable server with swappable pods or cards, comprising:

156. To the extent the preamble is limiting, it is my opinion that Grounds 3-4 each teaches the preamble.

Ground 3: Atsatt-113 teaches the preamble.

157. Atsatt-113 teaches **a programmable device**. For example, Atsatt-113 states that its “disclosure relates generally to configuration of *programmable devices*, such as field programmable gate arrays (FPGAs).” Ex-1006, ¶1. Atsatt-113 further explains:

Modern electronics such as computers, portable devices, network routers, data centers, Internet-connected appliances, and more, tend to include at least one integrated circuit device. Integrated circuit devices take a variety of forms, including processors, memory devices, and programmable devices, to name only a few examples. A field programmable gate array (FPGA) is one type of programmable device utilizing integrated circuits. Programmable devices may include logic that may be programmed (e.g., configured) after manufacturing to provide a wide variety of functionality based on various design possible within the programmable devices. Thus, programmable devices contain programmable logic (e.g., logic blocks) that may be configured and reconfigured to perform a variety of functions on the devices, according to a configured design. The integrated circuit

devices may be configured in numerous differen[t] configurations. These configurations are generally loaded into the integrated circuit devices. However, partially reconfiguring the programmable device may be problematic for situations where multiple users share a common device but distinct portions of the device.

Ex-1006, ¶3.

158. Atsatt-113 teaches its PLD/FPGAs are used **for data flow processing in a user-configurable server with swappable pods or cards**. For example, Atsatt-113 applies its PLD/FPGA teachings to a “server having multiple virtual machines.” Ex-1006, ¶5. As Atsatt-113 states, “[m]odern electronics such as computers, portable devices, network routers, data centers, Internet-connected appliances, and more, tend to include at least one integrated circuit device,” such as Atsatt-113’s FPGAs. Ex-1006, ¶3. Thus, Atsatt-113 teaches the use of PLDs/FPGAs for data flow processing in servers, including for data center and network routing applications. *Id.*

159. Additionally, the incorporated How reference¹⁵ discloses the use of systems with FPGAs for “computer networking, data networking, instrumentation, video processing, digital signal processing, or any other application where the

¹⁵ U.S. Patent Pub. No. 2016/0049941A1 (“How”) was “incorporated by reference in its entirety for all purposes” in Atsatt-113. Ex-1006, ¶29.

advantage of using programmable or reprogrammable logic is desirable.” Ex-1031, ¶58. Therefore, the combined teachings of Atsatt-113 and its incorporated reference confirm the known and obvious use of FPGAs in data centers and network applications for data flow processing. *See id.*

160. Additionally, this would have been obvious to a POSITA because Atsatt-113 teaches using FPGAs in servers while also teaching the use of ICs, such as FPGAs, in “data centers.” Ex-1006, ¶3, ¶5. Servers having multiple virtual machines (Ex-1006, ¶5) were often used in data centers for data flow processing—e.g., for cloud computing—and it was well-known that accelerators, such as FPGAs, were used in servers to accelerating the processing of data flows, including for network data, multimedia data, big data, AI training data, and other common cloud computing workloads. *Supra* §I.B. By 2018, these workloads had been prevalent for many years—in many cases well over decade. *Id.* Thus, Atsatt teaches the transfer and processing of data as it flows through a data center. *Id.*

161. Moreover, Atsatt-113 teaches user-configurable FPGAs in servers. Ex-1006, ¶5. As explained below, users configured the server by loading custom logic onto the FPGA. *Infra* §V.B.2[1c]. Moreover, by 2018, desktop computers and servers typically included a mother board with various PCIe slots or other interfaces for inserting cards and connecting peripherals such as FPGAs, graphics processing units (GPUs), and network interface controllers (NICs). Thus, servers

in 2018 were configured with swappable cards to accelerate workloads with increased computing power or capabilities. *See* Ex-1006, ¶5. In data centers, virtual machines were commonly implemented using racks of servers with FPGA cards. *Supra* §I.B. Thus, Atsatt-113 teaches a user-configurable server with swappable cards. Additionally, this would have been obvious to a POSITA given Atsatt-113’s teachings of data centers and FPGAs in servers. Ex-1006, ¶3, ¶5.

162. For at least the reasons explained above, Atsatt-113 teaches the preamble.

Ground 4: Biederman further teaches the preamble.

163. Biederman provides additional teachings for data flow processing with swappable pods or cards. A POSITA would have been motivated to combine Atsatt-113’s teachings with Biederman’s usage model of FPGAs in configurable servers for data flow processing, as I explain in detail below.

164. Biederman teaches using FPGAs in a **user-configurable server with swappable pods or cards**. For example, Biederman teaches implementing “network interface card[s]” using FPGAs. Ex-1005, ¶25, ¶2 (“network cards”), ¶28 (“‘processing element’ refers to constructs able to process data, such as processes, threads, virtual machines, and FPGA programs”), ¶29 (“Example computing units include...an FPGA...”), Fig. 1 (Network Interface Device 102):

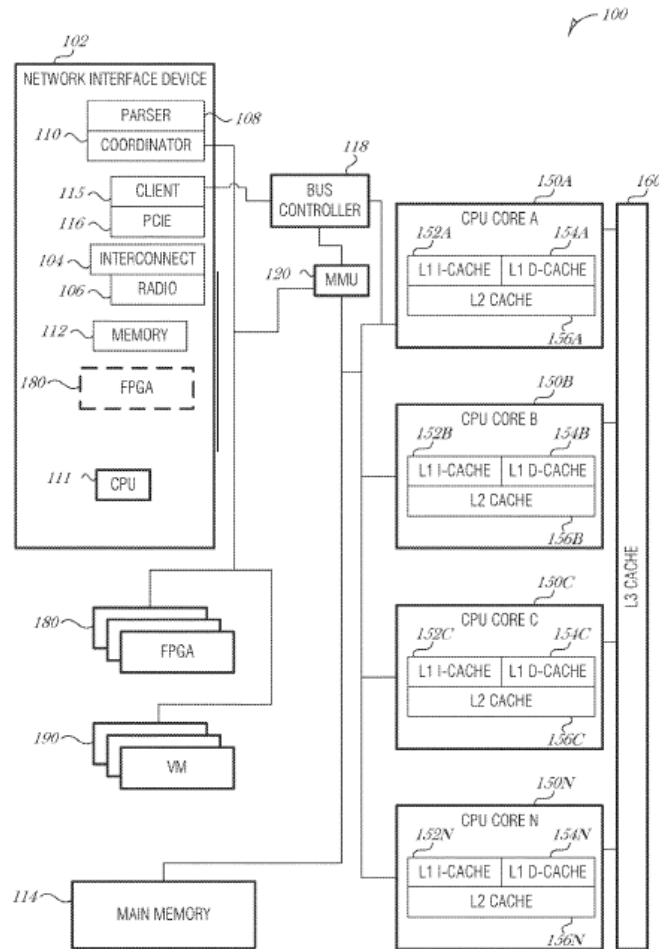


FIG. 1

165. In particular, Biederman teaches FPGAs on PCIe cards; and by 2010, PCIe was the prevalent interface standard for swappable cards in PCs and servers. Ex-1005, ¶52, ¶54, Fig. 1. Biederman teaches user-configurable network servers with multiple swappable PCIe cards. Ex-1005, ¶174-176, ¶50. As explained below, those PCIe cards include FPGAs that the user configures by loading custom logic. *Infra* §V.B.2[1c].

166. Biederman teaches using FPGAs for **data flow processing**, including for example “packet processing” of network data, where the FPGA processes packet data flowing through the network. Ex-1005, ¶2, ¶31:

In general, the NID receives a packet or packets over some L1/L2 interface, and uses a (possibly stateful) parser to decide what the next operation is for the packet, where “stateful” may rely on current or historical saved state metadata. The NID may perform on-NID processing using on-NID computing units, such as ASICs, FPGAs, or the like, and processing elements, such as FPGA programs, VMs, or the like. The NID is able to 1) do things related to the packet, such as encapsulate/decapsulate, en/decrypt, add/remove headers, aggregate/split, add timestamps, etc.; and 2) do things related to the state of the packet, such as save/update metadata, change internal or system configurations to handle packet processing, query/use stored metadata, query/use current or historical state of NID or system, request/schedule NID, and system-level state changes (e.g., pre-load caches, load FPGA code in either on-NID FPGA or host FPGA, or both, etc.

Biederman also teaches data flow processing for video data, including “video conferences” with flows of real-time audio and video data that require complex processing. Ex-1005, ¶134 (discussing an example processing logic for “video

conferences, [which] may use complex processes to ensure *real-time or near real-time audio and video presentation*”), ¶152 (“real-time video”).¹⁶

167. Biederman further discloses using its FPGA cards in data centers to process data, including for example image data for “Artificial Intelligence/Machine Learning/Neural Network.” Ex-1005, ¶182-184. In all these examples, data flowing through the data center is processed over a given path. *See id.*

168. **Motivation to Combine.** A POSITA would have been motivated to apply Biederman’s teachings to Atsatt-113 for several reasons. First, the references themselves suggest the combination: Atsatt-113 teaches the use of FPGAs and servers (Ex-1006, ¶3, ¶5), while Biederman uses FPGA cards in servers for data-flow processing. Ex-1005, ¶¶174-176, ¶2, ¶¶182-184. A POSITA would have found it obvious to apply Biederman’s well-known usage models to Atsatt-113’s teachings, which provides specific examples of how FPGA-equipped servers were used. This would have been a combination of prior art elements

¹⁶ The ’943 patent confirms that this kind of processing falls within the scope of the recited “data flow processing.” The specification provides the similar example of “video processing” over a “data flow path.” Ex-1001, 9:51-10:31, 2:27-39, 2:49-53, 7:45-46, Figs. 8 & 11.

according to known methods to yield the predictable results of using Atsatt-113's FPGA teachings to provide data flow services on cards in servers.

169. Second, both references are similarly directed to compositions of FPGAs and techniques for using FPGAs in servers, including networking and video processing. Atsatt-113 teaches techniques for partitioning an FPGA between multiple users without explaining particular use cases at length. Thus, a POSITA would naturally have looked to known uses of FPGAs in servers, including Biederman's teachings of using FPGA-equipped servers for processing network data, video data, and machine learning data flows. *See, e.g.*, Ex-1006, ¶¶3-5; Ex-1005, ¶2, ¶¶174-184. Indeed, Atsatt-113 provides motivation to look to Biederman by incorporating How (Ex-1006, ¶29), which applies Atsatt-113's sectorized FPGA architecture to applications in "computer networking, data networking, ...video processing" (Ex-1031, ¶58).

170. Third, a POSITA would have been motivated by the benefits of using Atsatt-113's FPGA teachings for Biedermann's use cases. Using Atsatt-113's reconfigurable FPGAs to process network packets, images, or for AI training would have increased the flexibility of the platform for common workloads. Indeed, these kinds of applications often used or swapped custom processing algorithms, and Atsatt-113's FPGA teachings allowed for reconfiguring individual partitions during operation (Ex-1006, ¶36), which would have been well-suited for

this purpose. Atsatt-113 loads customized user circuits into an FPGA (Ex-1006, ¶5), and the combination would further provide the well-known advantage of fast processing with low energy/thermal requirements, in comparison to using CPUs. *Supra* §I.B (¶17).

171. A POSITA would have had a reasonable expectation of success in combining these teachings because Biederman simply provides FPGA use cases for Atsatt-113's FPGA teachings. Thus, Atsatt-113's teachings would have been used in the same way that Atsatt-113 already teaches, applied to Biederman's context to perform packet processing, AI training, and machine learning, which Biederman teaches was performed by FPGAs. Ex-1005, ¶¶174-184. Furthermore, as explained above, the references themselves suggest the combination, which therefore would have provided a POSITA with a reasonable expectation of success. A POSITA would have been more than capable of combining the teachings of these two references. *See supra* §III.

172. For at least the reasons explained above, Ground 4 teaches the preamble.

[1a] a programmable logic device (PLD) having a first region and a second region;
--

173. Atsatt-113 teaches this limitation. For example, Figure 9 of Atsatt-113 discloses a block diagram for a “programmable logic device (e.g., FPGA 40).”

Ex-1006, ¶59, ¶29 (device 40 is “any suitable type of programmable logic device”). The PLD has a first region (e.g., the control area annotated with a red rectangle to the left in Fig. 9) and a second region (e.g., programmable logic sectors 48 annotated with a blue rectangle to the right).¹⁷ Ex-1006, Fig. 9:

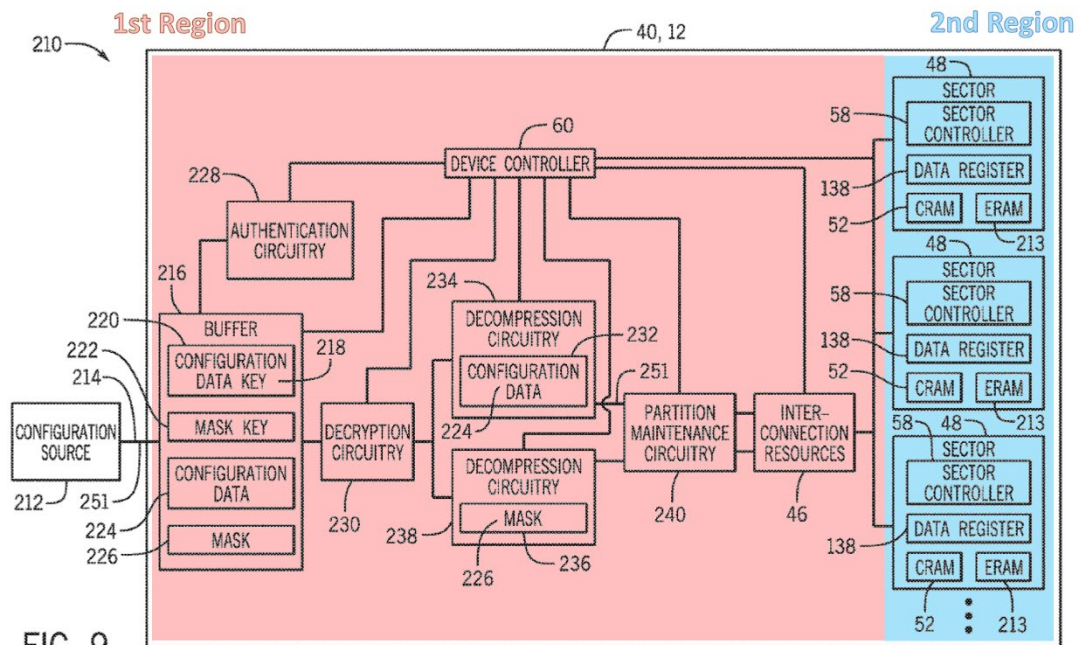


FIG. 9

174. Atsatt-113 also teaches a PLD under its proper construction: an IC where the user can alter the connection pattern of the circuits. *Supra* §IV.A. Atsatt-113 discloses a “programmable logic device” and further explains that it is an IC with circuitry programmed by the user after manufacturing to alter the logic pattern of the circuit,. For example, Atsatt-113 teaches a type of PLD called an

¹⁷ I note that Patent Owner took a broad view of the first and second regions recited in the claims, as I explained above. Ex-1021, 000002-15, 000027; *supra* §II.D.

FPGA and explains that it includes logic that is programmed after manufacturing.

Ex-1006, ¶3:

A field programmable gate array (FPGA) is one type of programmable device utilizing integrated circuits. Programmable devices may include logic that may be programmed (e.g., configured) after manufacturing to provide a wide variety of functionality based on various design possible within the programmable devices. Thus, programmable devices contain programmable logic (e.g., logic blocks) that may be configured and reconfigured to perform a variety of functions on the devices, according to a configured design.

When the user configures or reconfigures the FPGA logic circuits to perform a particular function, it alters the interconnection pattern of the logic circuits to implement a logic function provided by the user. *See id.* This is taught by Atsatt-113 and consistent with how FPGAs had been known and used in the art for decades. *Supra* §I.A.

175. In the primary example explained throughout Atsatt-113, Atsatt-113 uses an FPGA¹⁸ to explain its teachings, with the configurable logic areas in the

¹⁸ Atsatt-113 uses the term FPGA to explain its teachings but also applies its teachings to “any suitable type of programmable logic device.” Ex-1004, ¶29. For convenience, I will refer to an FPGA throughout this and subsequent sections to refer to Atsatt-113’s teachings explained in connection with an FPGA as well as its

second region divided into sectors. Ex-1006, ¶29 (“In one example, the FPGA 40 is a sectorized FPGA....”). Figure 2 shows a physical layout of the FPGA that can have “any suitable number of programmable logic sectors.” Ex-1006, ¶31 (“There may be any suitable number of programmable logic sectors 48 on the FPGA 40. Indeed, while 29 programmable logic sectors 48 are shown here, it should be appreciated that more or fewer may appear in an actual implementation (e.g., in some cases, on the order of 50 to 100 sectors or more).”), ¶38, Fig. 2:

teachings that are generally applicable to other types of PLDs. Ex-1004, ¶29; ¶27 (“While the techniques of this disclosure are described chiefly in the context of...programmable logic devices with field programmable gate array (FPGA) fabric, this is meant to be illustrative and not limiting. Indeed, the filtering circuitry of this disclosure may be implemented in other integrated circuit devices. For example, other types of integrated circuits, such as applicant-specific integrated circuits (ASICs), microprocessors, memory devices, transceivers, and the like, that may include regions of programmable logic.”).

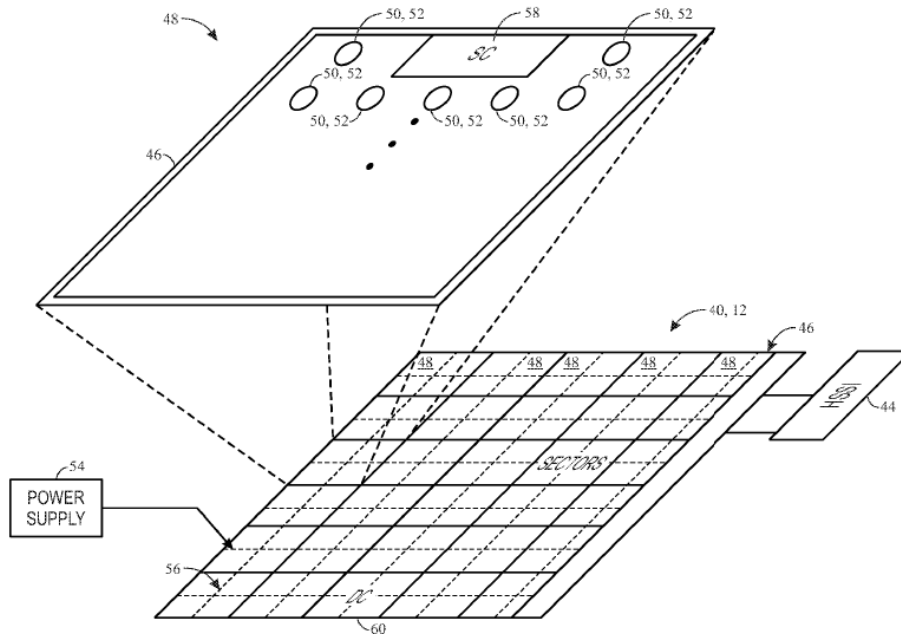


FIG. 2

176. Below, I explain Atsatt-113's teachings of the first and second regions in more detail for limitations [1b]-[1d].

177. For at least the reasons explained above, Ground 4 also teaches limitation [1a].

[1b] the first region comprising a hardware-based or firmware-based router with a port enabling communication between the router and the second region; and

178. Atsatt-113 teaches this limitation. I explained above for limitation [1a] that Atsatt-113 teaches a PLD (e.g., FPGA) having a first region (e.g., control area) and a second region (e.g., programmable logic sectors). *Supra* §V.B.2.[1a].

179. Atsatt-113 teaches the first region (e.g., control area) comprising a router with a port¹⁹ enabling communication between the router and the second region (e.g., programmable logic sectors). For example, the control area comprises functionality (e.g., in global resources, including interconnection resources and the device controller) to “route” data signals through the FPGA using interconnection resources (Ex-1006, ¶30), where the “signals may be transmitted as communication packets.” *Id.*, ¶33, Fig. 9. Atsatt-113 refers to FPGA “resources that are shared between...and [are] outside of the physical regions of the [programmable logic] sectors” as “global resources,” which includes interconnection resources. Ex-1006, ¶39 (“spanning resources that span physical partitions and/or sectors”), ¶55-56, Fig. 9 (annotated):

¹⁹ Patent Owner alleged in district court that various IC components “*separately and together*, constitute the claimed ‘hardware-based or firmware-based router’ and ‘port.’” Ex-1021, 000020; *supra* §II.D (discussing Patent Owner’s allegations as to limitation [1b]).

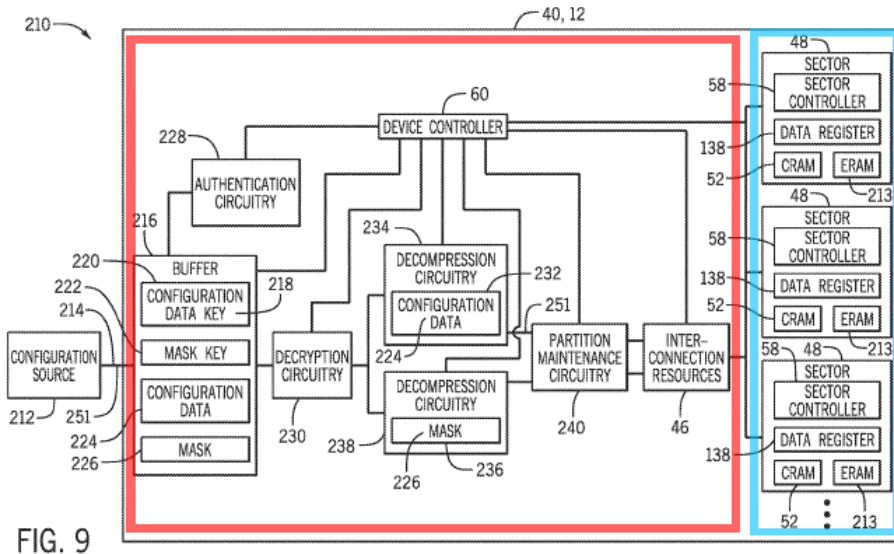


FIG. 9

180. The global resources include device controller 60 which enables communication by routing communication to the destination sector. Ex-1006, ¶33:

Each sector controller 58 thus may communicate with *the device controller 60, which may coordinate the operations of the sector controllers 58 and convey commands initiated from outside the FPGA device 40.* To support this communication, the interconnection resources 46 may act as a network between the device controller 60 and each sector controller 58. The interconnection resources may support a wide variety of signals between the device controller 60 and each sector controller 58. In one example, these signals may be transmitted as communication packets.

181. As seen in Figure 2, the interconnection resources provides a pathway to convey external communication to/from the HSSI interface to the logic sectors as directed by the routing functionality of the control area. Ex-1006, ¶33 (“To support this communication, the interconnection resources 46 may act as a network

between the device controller 60 and each sector controller 58.”), Fig. 2 (highlighted):

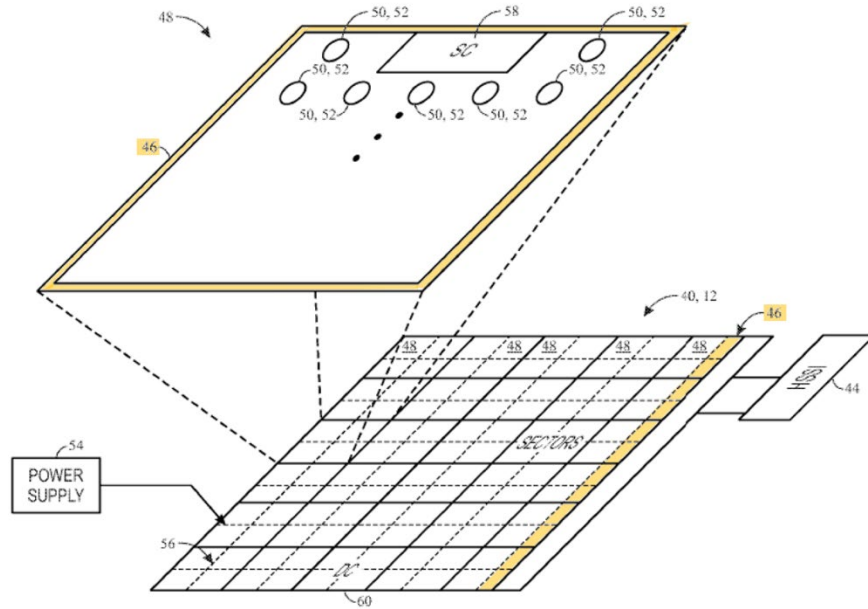


FIG. 2

182. In summary, Atsatt-113 includes router functionality in global resources to route communications to the second region, which further includes port functionality to enable communication between the global resources and the second region. For example, interconnection resources 46 includes functionality serving as a port for communication between circuits in the control area and the programmable logic sectors. Commands and data that are processed or authenticated in the first region (e.g., at authentication circuitry, decryption circuitry, decompression circuitry) are forwarded to their destination, such as the configuration RAM (CRAM) in programmable logic sectors, as packets. Ex-1006,

¶59-66. Atsatt-113 further describes that interconnection resources include ports for input/output of communication between different blocks of the FPGA. Ex-1006, ¶55 (“input/output ports (e.g., IO48)”), ¶39 (“input-output (IO) circuitry”). Figure 9 shows the logical connections established through the port from the device controller 60 and interconnection resources 46 in the first region to the programmable logic sectors 48 in the second region. Ex-1006, Fig. 9:

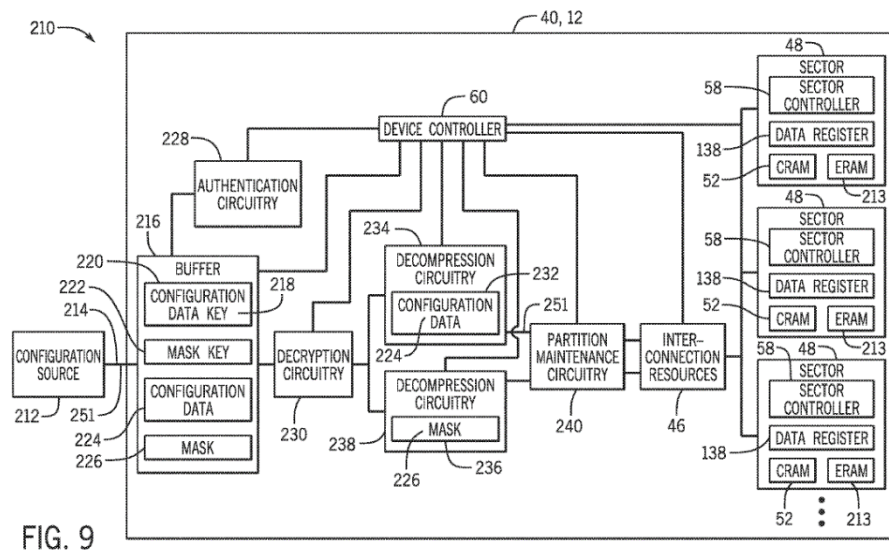


FIG. 9

Figure 3 shows the physical layout of some of the global resources (112, 114, 116, 118) in relation to the programmable logic sectors. Ex-1006, Fig. 3 (highlighted):



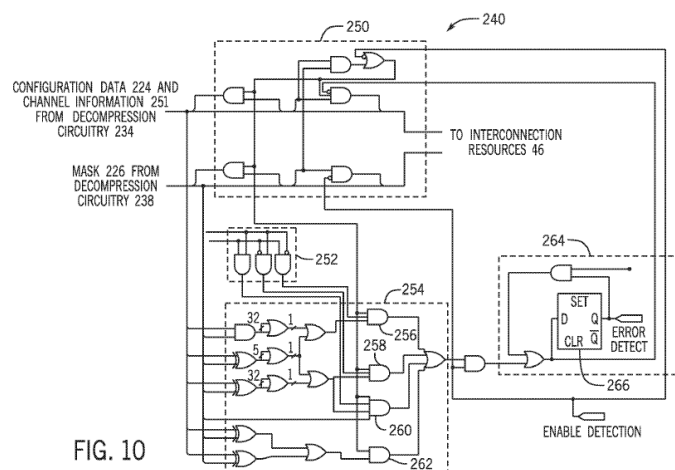
FIG. 3

183. Atsatt-113 provides an example of how configuration data is routed from the device controller in the control area to the correct sectors for a partition via the interconnection resources. Ex-1006, ¶81:

If the partition maintenance circuitry 240 does not generate or detect an error as illustrated in Table 1, then the configuration data 224 has been validated or verified to not perform unauthorized or improper modifications to CRAM cells 52 and/or ERAM cells 213 of other reconfigurable partitions of the FPGA 40. *Turning back to FIG. 9, the validated or verified configuration data 224 is sent to the interconnection resources 46 to be distributed to one or more sectors 48 of the FPGA 40.* In some embodiments, the mask 226 may also be sent to the interconnection resources 46. On arriving at the one or more sectors 48, the configuration data 224 may be stored in one or more respective data registers 138, from which one or more sector

controllers 58 modify contents of respective CRAM cells 52 and/or ERAM cells 213.

184. Atsatt-113 explains that when the user's circuit in a partition of the programmable logic sectors is updated, channel information is used to route (e.g., direct) the circuit configuration data by “inform[ing] the interconnection resources 46 where to send” it. Ex-1006, ¶71 (“The channel information 251 may inform the interconnection resources 46 where to send and how to format the configuration data 224.”), Ex-1006, ¶100 (same), Fig. 10:



185. Communications with external devices are sent to/from the programmable logic sectors. Functionality in the global resources acts as an intermediary for receiving inbound communication and sending outbound communication to the host or other external devices. Ex-1006, ¶¶30-33. These external communications travel between the router functionality in global resources and the programmable logic sectors. *Id.* The global resources include

various interfaces for external communication with other components in the system, including “Peripheral Component Interconnect Express (PCIe) circuitry” (Ex-1006, ¶39) and “high-speed serial interface—HSSI” (Ex-1006, ¶55]). *See also*, Ex-1006, Fig. 2 (“HSSI 44”).

186. Atsatt-113 teaches the router is hardware-based. The above-explained teachings are implemented on the hardware-programmable fabric of the FPGA, where functionality is implemented as hardware, rather than software. Ex-1006, ¶39:

The programmable fabric 100 also includes global resources 112, 114, 116, and 118. The global resources 112, 114, 116, and 118 include resources that are shared between the sectors 102A-110H and outside of the physical regions of the sectors 102A-110H and/or partitions of the programmable fabric 100. These global resources 112, 114, 116, and 118 may include clock routing to each sector, phase lock loop (PLL) circuitry, input-output (IO) circuitry, Peripheral Component Interconnect Express (PCIe) circuitry, hardware platform interface (HPI), a hard processor system (HPS), and/or other suitable circuitry. The programmable fabric 100 may also include spanning resources that span physical partitions and/or sectors. These spanning resources may include a horizontal wire and a vertical wire utilized by the partition to transmit and/or receive data and/or output data. The spanning resources may also include scanning chains used for register readback from the partitions.

187. For at least the reasons explained above, Ground 4 also teaches limitation [1b].

[1c] the second region comprising one or more sandboxes with user-definable programmable electronic circuits of the PLD,
--

188. Atsatt-113 teaches this limitation. As I explained above for limitation [1a], Atsatt-113 teaches a PLD (e.g., FPGA) having a first region (e.g., control area) and a second region (e.g., programmable logic sectors). *Supra* §V.B.2.[1a].

189. Atsatt-113 teaches **the second region** (e.g., programmable logic sectors) **comprising one or more sandboxes**. For example, the programmable logic sectors comprise logical groupings within the programmable fabric of the FPGA.²⁰ Ex-1006, ¶36 (“The programmable fabric 100 includes...sectors 102A-110H, may include a logical grouping of logical elements in the programmable fabric 100.”). Atsatt-113 teaches grouping the sectors into partitions for different users to load dedicated logic, performing different functions, into their particular partitions. Ex-1006, ¶40 (“*The programmable fabric 100 may also include*

²⁰ For the '943 patent's embodiments, sandboxes are designated areas of an FPGA. Ex-1001, 1:37-44, 5:19-49 (“Each of one or more sandboxes 132 on the FPGA 122 is designated for programmable electronic circuits....”).

partitions that may be directed to different functions, users, machines, permissions, or other divisions of the programmable device.”).

190. Figure 3 illustrates the allocation of groupings, called partitions, across programmable logic sectors. Ex-1006, ¶40 (“[S]ome partitions, such as a partition 120, are entirely within and fully encompass a sector.... .The programmable fabric 100 may also include partitions, such as partition 126, that spans multiple sectors.... .Some partitions, such as partition 131, extend beyond and/or do not fully encompass sectors in which the partition is located.”), Fig. 3:

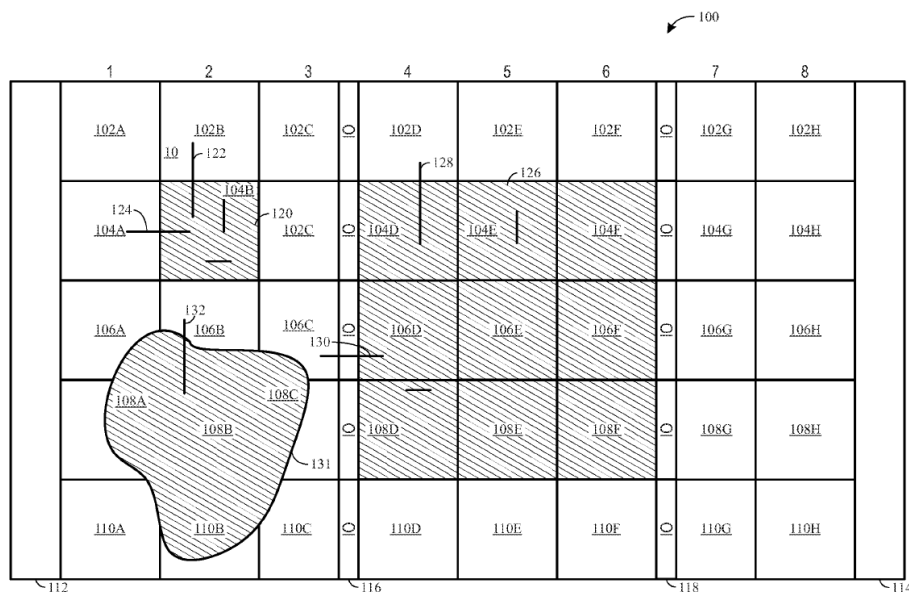


FIG. 3

191. Atsatt-113 teaches that the partitions are sandboxes where the partitions and their related resources are isolated from other partitions owned by other users. For example, partitions and their related resources are isolated from partitions assigned to other users “to ensure they are not modified by partitions that

do not own them.” Ex-1006, ¶41 (“As discussed below, resources within and/or spanning into a partition are managed by the system to ensure they are not modified by partitions that do not own them.”).

192. Atsatt-113 teaches various mechanisms for isolating partitions. For example, maintaining an “authenticated list” of addresses to control access to the embedded RAM associated with a sector. Ex-1006, ¶54 (“To prevent unauthorized overwrites, an authenticated list of allowed ERAM address for each region may be loaded into the configuration system as an ordered list. The configuration system processor (e.g., SDM, local/sector managers) writes the packet header from the ordered list into the ERAM packet to ensure that writing only occurs to only authorized addresses.”). Atsatt-113 also explains how a “violation of resource separation” can be prevented during configuration of user partitions. Ex-1006, ¶53 (“In some embodiments, the configuration system (e.g., SDM) may compare allowed masks of each region and report a violation of resource separation. In some embodiments, when the configuration system detects a violation, the configuration may halt configuration of the programmable device.”). Atsatt-113 also discusses the use of masks to enforce isolation by defining the contours of each sandbox (e.g., the user partitions) and the resources a partition can or cannot access. Ex-1006, ¶¶41-42 (“To provide isolation between partitions, an ‘allowed’ masks set for each partition may be generated to cover all

resources that are owned by a partition. The masks identify both the sector and the resources used in that sector by a particular partition. For each sector there is then a set of masks that define the resources allocated and allowed for each partition.”).

193. Atsatt-113 teaches that the partitions of the programmable logic sectors comprise **user-definable programmable electronic circuits of the PLD**. Atsatt-113’s partitions are portions of the programmable fabric of an FPGA. Ex-1006, ¶40, ¶36. As Atsatt-113 explains, “[a] field programmable gate array (FPGA) is one type of programmable device utilizing integrated circuits. Programmable devices may include logic that may be programmed (e.g., configured).” Ex-1006, ¶3. Thus Atsatt-113 explains how its FPGA contains physical circuits where the logical interconnections of the actual electronic circuits are programmable and reconfigurable (*id.* ¶3), and Atsatt-113 further teaches how the logic loaded to the user partitions is reconfigurable by the user, and therefore user-definable. Ex-1006, ¶5 (“For instance, a server having multiple virtual machines may enable a user to change a region allocated to the user....”), ¶36 (“For example, each sector may include logic elements that would be typically be used in the programmable fabric 100 by each user or function.”). Atsatt-113 confirms this by describing configuration sources for the reconfigurable partitions to include “configuration data 224 (e.g., *used to implement a user design on the partially reconfigurable region*).” Ex-1006, ¶59. In short, Atsatt-113 teaches the

user defining the logic in each partition by programming and reconfiguring the logic of electronic circuits in each FPGA partition. *Id.*, ¶¶3-5, ¶36, ¶59.

Additionally, integrated circuits are electronic circuits, and the configurable logic elements in PLDs, including FPGAs, were implemented as electronic circuits.

This was well-known because integrated circuits are electric circuits, powered by the flow of electricity through the integrated circuit, and PLDs, including FPGAs, were implemented as ICs, as Atsatt-113 teaches. Ex-1006, ¶3, ¶5.

194. For at least the reasons explained above, Atsatt-113 teaches limitation [1c], which applies to Grounds 3 and 4.

[1d] wherein the port comprises: a bridge being lockable to prevent user access, and unlockable to enable user access, to each of the sandboxes and the user-definable programmable electronic circuits therein.
--

195. Atsatt-113 teaches this limitation. As I explained above for limitation [1a], Atsatt-113 teaches a PLD (e.g., FPGA) having a first region (e.g., control area) and a second region (e.g., programmable logic sectors). *Supra* §V.B.2.[1a]. I also explained, for limitation [1b], how the control area of Atsatt-113's FPGA is a first region that comprises a router with a port. *Supra* §V.B.2.[1b].

196. Atsatt-113 teaches **wherein the port comprises: a bridge being lockable to prevent user access, and unlockable to enable user access, to each of the sandboxes (e.g., user partitions) and the user-definable programmable**

electronic circuits therein.²¹ For example, the interconnection resources further includes bridge functionality that is lockable or unlockable to prevent (by locking) or enable (by unlocking) user access to configure/reconfigure each partition. Ex-1006, ¶¶66-81. Before a mask is used to configure/reconfigure the FPGA, the mask is authenticated²² using keys. Ex-1006, ¶59 (“For example, the data stream 214 may include configuration data used to configure the FPGA 40, a mask used to maintain the reconfigurable partitions of the FPGA 40, an authentication key (or keys) used to authenticate the configuration data, and *an authentication key (or keys) used to authenticate the mask.*”). The device controller in the FPGA, alone

²¹ In district court, Patent Owner alleged this limitation is satisfied by logging in to a development tool kit to copy firmware to a device because the device “does not allow user[s] to directly access” it and “the user must log in” to verify account permissions to manage the programming of the accused card. Ex-1021, 000034-35, 000055-56; *supra* §II.D. Patent Owner further based its allegations on “various hardware security features to prevent unauthor[iz]ed access.” Ex-1021, 000037-41.

²² Atsatt uses keys to verify identity, consistent with known authentication in the art. *See, e.g.*, National Information Systems Security (INFOSEC) Glossary, NSTISSI No. 4009 September 2000 at p. 4.

or in conjunction with the authentication circuitry or engine, authenticates “the mask 226 with the mask authentication key 222.” Ex-1006, ¶¶62, ¶¶61-62. This is then applied to lock/unlock access to different resources and programmable logic sectors based on the “mask for a partition that indicates resources to which the partition is associated.” Ex-1006, ¶5 (“For instance, a server having multiple virtual machines may enable a user to change a region allocated to the user without enabling the user to change other regions. A configuration controller, before or during runtime, may generate a mask for a partition that indicates resources to which the partition is associated. The configuration controller may use the mask to reconfigure the partition without changing other partitions.”), ¶¶41-42. The masks “describe this ownership (block 204). The masks may include allowed and not allowed masks to indicate resource ownership for the partition.” Ex-1006, ¶56. In short, the mask mechanism is used to enable or disable user access to each partition. Ex-1006, ¶¶41-42, ¶56.

197. Figure 8 shows the flow of mask generation and use. First, a base partition is set for global resources on initialization. Ex-1006, ¶55 (“Global resources are often configured once and not updated by partition updates. ...Often, they are defined by a base partition at the time the set of partitions is defined.”). Resources, including the programmable logic sectors to host user logic, are assigned to a partition. Ex-1006, ¶56, Fig. 8 (step 202). Masks are then generated

to record the assignments. Ex-1006, ¶56, Fig. 8 (step 204). When user partitions are updated and reconfigured, the appropriate masks are applied to unlock the appropriate partition while leaving other partitions locked. Ex-1006, ¶56, Fig. 8 (step 202).

198. For example, for embodiments that use a mask, the mask is sent to the interconnection resources to unlock the partitions where the user has access, allowing FPGA configuration data to be sent to those partitions, while leaving other partitions locked. Ex-1006, ¶81:

Turning back to FIG. 9, the validated or verified configuration data 224 is sent to the interconnection resources 46 to be distributed to one or more sectors 48 of the FPGA 40. *In some embodiments, the mask 226 may also be sent to the interconnection resources 46.* On arriving at the one or more sectors 48, the configuration data 224 may be stored in one or more respective data registers 138, from which one or more sector controllers 58 modify contents of respective CRAM cells 52 and/or ERAM cells 213.

See also Ex-1006, ¶30 (“Each programmable logic sector 48 may include a number of programmable logic elements 50 having operations defined by configuration memory 52 (e.g., configuration random access memory (CRAM)).”), ¶34 (“In one example, the configuration memory 52 may be implemented as configuration random-access-memory (CRAM) cells.”), ¶¶43-45, ¶54 (“To prevent unauthorized overwrites, an authenticated list of allowed ERAM address for each

region may be loaded into the configuration system as an ordered list.”). A POSITA would have been motivated to use masks as taught by Atsatt-113 because they increase the security and simplicity of authenticating and managing users’ partition permissions while preventing users from altering partitions they do not have access to. Ex-1006, ¶26. The interconnection resources are responsible for sending configuration data to the programmable logic sectors. Ex-1006, ¶71, ¶100, Fig. 10.

199. Therefore, functionality in the interconnection resources acts as a bridge for accessing specific programmable logic sectors and associated resources. For example, transactions that pass over the interconnection resources, including reading/writing to programmable logic sectors, are modified based on the applicable mask state. Ex-1006, ¶41 (“The fundamental mechanism relies on the fact that configuration of a partition relies on a read/modify/write sequence that uses mask images that define the exact resources that are to be left un-modified.”). This mechanism locks the interconnection resources for the associated user and commands to prevent modification of partitions when the user’s key has not been authenticated. *Id.*, ¶81, ¶59. A POSITA would have understood this and found it obvious from Atsatt-113’s teachings because the interconnection resources distributes configuration data to partitions, and the mask is authenticated to confirm whether the user has access to specific partitions. *Id.*, ¶81, ¶71, ¶51, ¶57,

¶¶66-67. That is why Atsatt-113 teaches sending the mask to the interconnection resources—to enforce the mask. *Id.*, ¶81. In like manner, Atsatt-113 teaches the interconnect resources are unlocked for access to authorized partitions and resources. Ex-1006, ¶26 (“[E]mbodiments of the present disclosure relate generally to partial reconfiguration of only a portion of a programmable fabric using a mask to determine which resources may be changed by the partition.”).

200. As explained above, the FPGA partitions comprise user-definable programmable electronic circuits. *Supra* §V.A.2[1c].

201. For at least the reasons explained above, Ground 4 also teaches limitation [1d].

3. Independent Claim 10

202. I explained above how the limitations of claim 10 recite similar language to claim 1. *Supra* §V.A.4; *infra* Appx-3 (claim appendix showing claims side-by-side). Grounds 3-4 teach the overlapping portions of claim 10 for the same reasons explained above for claim 1. *Supra* §V.B.2. Grounds 3-4 also teach the additional language of claim 10, as explained below.

[10pre]-[10a]

203. The limitations of [10pre] and [10a] are also recited in [1pre] and [1a]. Therefore, these limitations are satisfied for the same reasons that I explained

above for [1pre] and [1a], respectively. *Supra* §V.B.2[1pre]-[1a] (incorporated by reference).

[10b] the first region comprising a hardware-based or firmware-based router having a bridge with a port enabling communication between the router and the second region;
--

204. Limitation [10b] recites similar language to limitation [1b], further specifying the router “having a bridge” with a port. Limitation [1d] recites a similar limitation: “the port: comprises a bridge.” Therefore, for both claims, the first region comprises a router with a bridge and port.²³

205. As explained above, Atsatt-113 teaches the first region comprising a hardware-based router (e.g., functionality in global resources, including interconnection resources and the device controller, to route signals) with a bridge and port that enables communication between the router and the programmable logic sectors. Ex-1006, ¶¶30-33, ¶¶59-66, ¶55, ¶39; *supra* §V.B.2.[1b] (incorporated here by reference), §V.B.2[1d] (incorporated here by reference). Therefore, Grounds 1-2 teach limitation 10[b].

²³ The '943 patent uses router and bridge interchangeably. *Supra* n. 10.

[10c] the second region comprising a plurality of sandboxes with user-definable programmable electronic circuits of the PLD; and

206. Limitation [10c] recites similar language to limitation [1c] and is satisfied for the same reasons explained previously. *Supra* §V.B.2.[1c] (incorporated by reference). Limitation [10c] additionally requires a plurality of sandboxes instead of just one or more sandboxes as recited in limitation [1c].

Ground 3: Atsatt-113 teaches a plurality of sandboxes.

207. For example, Atsatt-113 explains its PLD includes a plurality of partitions for different users. Ex-1006, ¶40 (“*The programmable fabric 100 may also include partitions that may be directed to different functions, users, machines, permissions, or other divisions of the programmable device.*”). Because the PLD has partitions (plural) for different users, rather than a single user, Atsatt-113 teaches a more than one partition (sandbox) for the more than one user. I explained the plurality of sandboxed partitions for claim 1. *Supra* §V.B.2[1c] (incorporated by reference).

208. For at least the reasons explained above, Grounds 3-4 teach this limitation.

Ground 4: Biederman further teaches **a plurality of sandboxes**.

209. I previously explained how Biederman teaches a plurality of sandboxes; those same teachings apply here. *Supra* §V.A.4.[10c] (incorporated by reference).

210. **Motivation to Combine.** Atsatt-113 provides express teaching, suggestion, and motivation to combine by teaching partitions (plural) for different users. Ex-1006, ¶40. Biederman provides mechanisms for managing and resizing multiple partitions on an FPGA. Ex-1005, ¶134. Therefore, a POSITA would have been motivated to combine Atsatt-113's FPGA teachings with Biederman's teachings of using multiple partitions and adjusting their size. The motivation to combine Atsatt-113 and Biederman is further explained below for claim 9. *Infra* §V.B.8 (Motivation to Combine Subsection incorporated by reference).

[10d] an internal communication network in the PLD, coupling the port of the bridge of the router and the one or more sandboxes for the data flow processing,

211. It is my opinion that Grounds 3-4 each teaches this limitation.

Ground 3: Atsatt-113 teaches limitation [10d].

212. As I explained above for limitation [1b], Atsatt-113's global resources includes functionality for a router with a port that enables communication between the global resources and the programmable logic sectors. Ex-1006, ¶55, ¶39, Fig. 9; *supra* §V.B.2.[1b] (incorporated here by reference). Atsatt-113's

interconnection resources includes functionality that serves as an internal communication network in the FPGA that couples the programmable logic sectors to the port. Ex-1006, ¶33 (“To support this communication, the *interconnection resources 46* may act as a network between the device controller 60 and each sector controller 58.”).

213. Atsatt-113 teaches the network in the interconnection resources couples the port of the interconnection resources to the sandboxes (user partitions), because the interconnection resources couple global resources to the sector controllers for the logic sectors comprising each user partition. As seen in Figure 2, interconnection resources 46 form a network that couples programmable logic sectors to each other and global resources, such as an HSSI port. Ex-1006, Fig. 2 (highlighted):

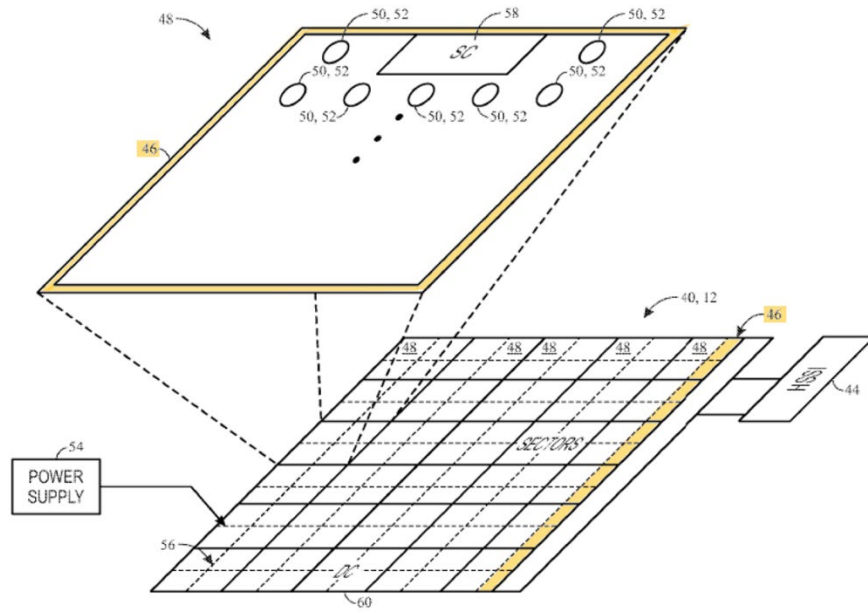
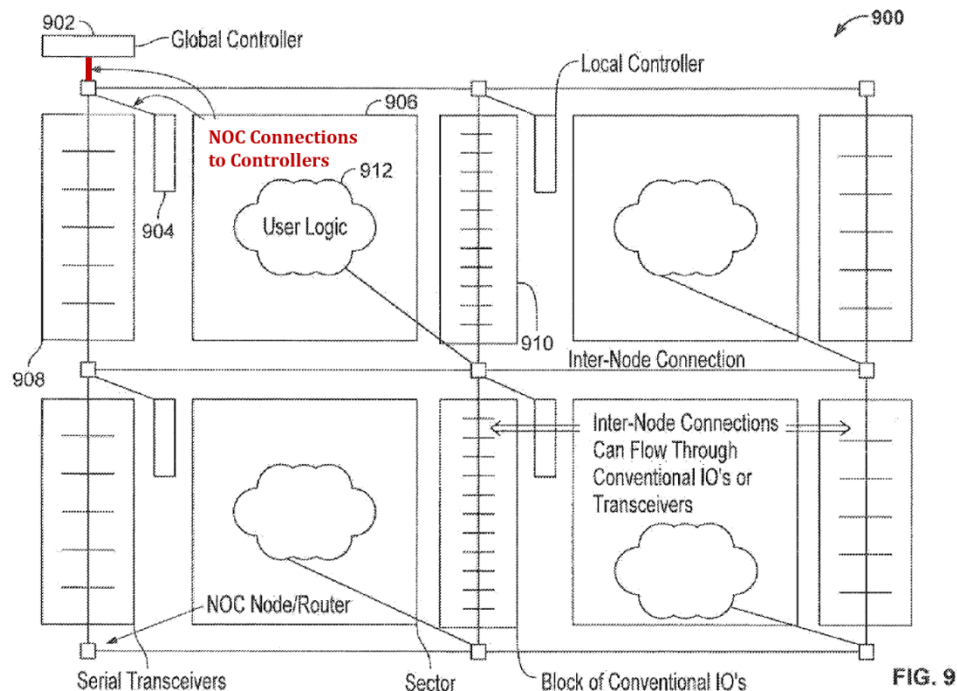


FIG. 2

214. Atsatt-113 teaches further details of the port and network in the interconnection resources. For example, the incorporated How reference²⁴ teaches “a network of routing wires and programmable switches which are configured by SRAM bits or other suitable means to implement routing connections between the

²⁴ How was “incorporated by reference in its entirety for all purposes” in Atsatt-113. Ex-1006, ¶29. In particular, a POSITA would have understood Atsatt-113 to incorporate How’s disclosures regarding the structure, configuration, and operation of its FPGA, because Atsatt-113 states that “the FPGA 40 is a sectorized FPGA of the type described in” How, in connection with its discussion of integrated circuit 12. *Id.*

blocks.” Ex-1031, ¶21, ¶39. Figure 9 of How depicts a port that connects global controller 902 to a Network-On-Chip, which in turn connects to the programmable logic sectors. Ex-1031, ¶39 (“a mesh-based network-on-chip (NoC) between sectors 906 and I/O-oriented blocks 908 and 910”), Fig. 9 (highlighted):



Therefore, the “NOC Connection” to the global controller enables communication between the FPGA global resources and the programmable logic sectors. *See id.*

215. Atsatt-113 states that its sectorized FPGA implementation is described in How. Ex-1006, ¶29. Both are Altera references with the same overall architecture, and share a common co-inventor Atsatt. Atsatt-113’s control area (including the device controller) in global resources corresponds to How’s global controller because both are described as controlling the operations of the FPGA

globally, including controlling the local sector controllers and managing configuration of the FPGA. Ex-1006, ¶33, ¶¶61-66; Ex-1031, ¶¶3-4, ¶¶25-26, ¶¶32-37, ¶29. How's NoC falls within the scope of Atsatt-113's interconnection resources because it interconnects the various resources in the FPGA, including the global/device controller and the programmable logic sectors. Ex-1031, ¶¶37-38 (“network of *connections* between the global controller and each sector's local controller.”), ¶39.

216. Reading Atsatt-113's sectorized FPGA teachings in combination with How's details of FPGA sectors, Atsatt-113's interconnection resources includes a port (which includes the NOC Connection) and a network-on-chip (Ex-1031, ¶39, ¶21, Fig. 9) that couples the port to the programmable logic sectors, in the manner described by How. This is clear from the teachings, and a POSITA would also have found it obvious because Atsatt-113 explains how the interconnection resources “may act as a network” (Ex-1006, ¶33), while How provides details regarding that network (Ex-1031, ¶39).

217. For at least the reasons explained above, Ground 3 teaches this limitation.

Ground 4: Biederman further teaches this limitation.

218. As explained for claim 1, Biederman provides additional teachings for using FPGA partitions for **data flow processing**, including for example, “packet

processing” of network data, processing image data, and AI data. Ex-1005, ¶2, ¶¶182-184; *supra* §V.B.2.[1pre] (incorporated here by reference).

219. **Motivation to Combine.** The motivation was explained for claim 1. *Supra* §V.B.2.1[pre] (motivation to combine). The same analysis applies here and is incorporated by reference.

220. For at least the reasons explained above, Ground 4 teaches this limitation.

[10e] wherein the bridge is lockable to prevent user access, and unlockable to enable user access, to the plurality of sandboxes and user-definable programmable electronic circuits therein.

221. Limitation [10e] recites similar language to limitation [1d] and is satisfied for the same reasons explained previously. *Supra* §V.B.2.[1d] (incorporated by reference).

222. Grounds 3-4 further teach the additional limitation of [10e], **a plurality of sandboxes**. Atsatt-113 teaches this alone (Ground 3) and in combination with Biederman (Ground 4). *Supra* §V.B.2.[10c] (incorporated by reference). Because Atsatt-113 teaches multiple partitions, its teachings for locking and unlocking the bridge apply to a plurality of sandboxes. Therefore, the same teachings that I explained above for limitation [1d] equally apply here.

223. For at least the reasons explained above, Grounds 3-4 teach this limitation.

4. Dependent Claim 2

2. **The programmable device of claim 1, further comprising: the router communicative to translate between a first protocol and a second protocol, for messaging with system resources.**

224. As I explained above for claims 1 and 10, Grounds 3-4 each teaches a PLD with a control region with a router, a programmable logic region with sandboxes, and a port with a bridge for locking/unlocking user access to the sandboxes. *Supra* §§V.B.2-3.

225. Atsatt-113 teaches **the router** (e.g., functionality in global resources) **communicative to translate between a first protocol and a second protocol, for messaging with system resources**. For example, Atsatt-113 teaches that functionality in the global resources utilizes various data and signal formats to communicate with external clients (e.g., using HSSI) or with the host device (e.g., using PCIe) which is then routed to/from the appropriate sandbox. Ex-1006, ¶39 (“These global resources 112, 114, 116, and 118 may include...Peripheral Component Interconnect Express (PCIe) circuitry...”), ¶55 (“input/output ports (e.g., 1048), interfaces (e.g., high-speed serial interface HSSI)”). Therefore, the routing functionality in the global resources translates between a first protocol (the external protocol, e.g., PCIe or HSSI) and a second protocol (the internal protocol of the FPGA).

226. For example, PCIe is a protocol for connecting different chips, using a serial protocol with fewer lines to minimize routing space on printed circuit boards. PCIe was designed for external inter-chip communications, not internal chip communications. *See* Ex-1033, 0000017, 000029-32. Likewise, HSSI was used for communications between chips and other components; it is a serial transmission protocol primarily used in networking equipment and uses differential emitter-coupled logic (ECL) to achieve high data rates and low noise level. *See* Ex-1032, 0000002, 000011-13. HSSI utilizes a simple control protocol using two control signals: TA (Data Terminal Equipment Available) and CA (Data Communication Equipment Available). Ex-1032, 0000002, 0000004.

227. In contrast to serial PCIe and HSSI protocols, internal communication within an FPGA typically occurred in parallel, where multiple signals (e.g., bitstreams) traverse processing paths in parallel. As known in the art, this was true for the interconnect resources that are programmed to connect the different parts of the FPGA. The incorporated How²⁵ reference discloses other communication protocols such as “general purpose I/Os” and “DDR memory interfaces.” Ex-1031, ¶21, Fig. 1. Atsatt-113 at least teaches an internal protocol that is different from these various internal and external protocols and translating between the

²⁵ How was incorporated by reference. *Supra* n. 24.

internal protocol and the external protocols, because Atsatt-113's FPGA interfaces with a variety of protocols that require translation for interoperability. *See also* Ex-1031, ¶36, Fig. 7.

228. Thus, Atsatt-113 teaches functionality in the global resources that translates from the internal protocol of the FPGA (over the internal interconnect to internal system resources) to external protocols (e.g., PCIe and HSSI).

229. Atsatt-113 teaches the router performing these translations for messaging with system resources external to the FPGA. Atsatt-113 teaches using FPGAs in networked servers that contain a multitude of resources. When processing data on distributed infrastructure in this manner, it was conventional to store data to be processed on external resources, including memory elsewhere in the server or remote storage, because FPGAs had limited memory capacity. Thus, the communications translated over the PCIe/HSSI or other external interfaces usually included such communications for the purpose of messaging with other system resources.

230. Biederman further teaches examples of such external resources, including other FPGAs, processors, co-processors, memory, storage devices, etc. Ex-1005, ¶¶53-54, ¶59, ¶¶193-196, Fig. 1, Fig. 18:

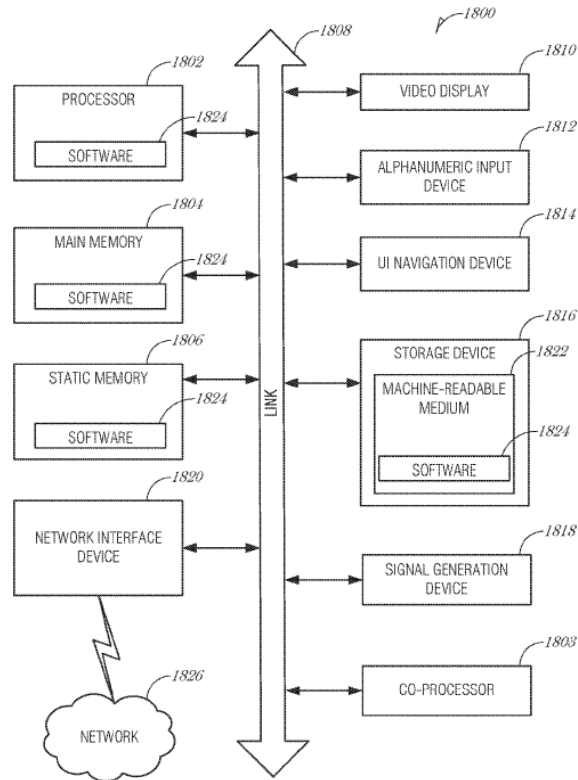


FIG. 18

231. Biederman teaches that the network interface device 1820, which includes the FPGAs, “may query or otherwise interface with various components of the system 1800 to inspect cache memory; trigger or cease operations of a virtual machine, process, or other processing element; or otherwise interact with various computing units or processing elements that are in the system 1800 or external from the system 1800.” Ex-1005, ¶198.

232. For at least the reasons explained above, Grounds 3-4 teach claim 2.

5. Dependent Claim 11

11. The programmable device of claim 10, further comprising: the router to translate between a first protocol used external to the second region and a second protocol used in the internal communication network in the PLD.

233. Claim 11 recites overlapping limitations with claim 2, which are satisfied for the same reasons explained above. *Supra* §V.B.4. Atsatt-113 further teaches the additional language of claim 11. *Infra* Appx-3 (claim appendix showing claims side-by-side). Atsatt-113 further teaches the first protocol is **used external to the second region** and the second protocol is **used in the internal communication network in the PLD**. As I just explained above for claim 2, Atsatt-113 uses a first external protocol, such as HSSI or PCIe. Ex-1006, ¶39, ¶55, Fig. 2. Since it is used external to the FPGA, the first protocol is not used in the second region (programmable logic sectors). I also explained how the FPGA uses a second protocol as the internal protocol within the FPGA, to communicate over the internal FPGA interconnect. *Supra* §V.B.4. As I explained above for limitation [10d], the interconnect resources is an internal communication network in the FPGA. Ex-1006, ¶33; *supra* §V.B.3.[10d] (incorporated here by reference).

234. For at least the reasons explained above, Ground 4 also teaches claim 11.

6. Dependent Claims 3 and 12

3. **The programmable device of claim 1, wherein each of the sandboxes comprises user-programmable one or more of: analog circuitry, digital circuitry, memory, processors or controllers.**
12. **The programmable device of claim 10, wherein each of the plurality of sandboxes comprises user-programmable one or more of: analog circuitry, digital circuitry, memory, processors or controllers.**

235. As I explained above for claims 1 and 10, Grounds 3-4 each teaches a PLD with a control region with a router, a programmable logic region with sandboxes, and a port with a bridge for locking/unlocking user access to the sandboxes. *Supra* §§V.B.2-4. It is my opinion that Atsatt-113 teaches the additional limitations of claims 3 and 12.

236. Atsatt-113 teaches **wherein each of the sandboxes comprises user-programmable digital circuitry**. For example, Atsatt-113 discloses the user partitions each comprise user-programmable digital circuitry (e.g., programmable logic sectors of the programmable fabric of the FPGA). Ex-1006, ¶3 (“A field programmable gate array (FPGA) is one type of programmable device utilizing integrated circuits. Programmable devices may include logic that may be programmed (e.g., configured) after manufacturing to provide a wide variety of functionality based on various design possible within the programmable devices. Thus, programmable devices contain programmable logic (e.g., logic blocks) that may be configured and reconfigured to perform a variety of functions on the

devices, according to a configured design....[There are] situations where multiple users share a common device but distinct portions of the device.”). As I explained above for limitation [1c], Atsatt-113 teaches the user partitions (sandboxes) comprise user-definable programmable electronic circuits. *Supra* §V.B.2.[1c]. That analysis applies here and is incorporated by reference.

237. The digital circuitry for each partition is specifically programmable by each owner-user. Ex-1006, ¶5 (“Some designs may be programmed in the integrated circuit that target partial reconfiguration regions without changing configurations of adjacent regions. For instance, a server having multiple virtual machines may enable a user to change a region allocated to the user without enabling the user to change other regions.”). Atsatt-113 confirms each partition is separately programmed with configuration data for that partition. Ex-1006, ¶59 (“FIG. 9 is a block diagram of a system 210 that maintains reconfigurable partitions of the programmable logic device (e.g., FPGA 40) of FIG. 2, according to an embodiment of the present disclosure....Specifically, the configuration data may reconfigure at least a partition of the FPGA 40 (e.g., partially reconfigure the FPGA 40) when programmed in the FPGA 40....[T]he configuration data 224 [is] used to implement a user design on the partially reconfigurable region[.]”).

238. Atsatt-113 further explains that each of the programmable logic sectors comprise **digital circuitry**, because they are a collection of digital logic

elements such as combinational logic, sequential logic, and lookup tables. Ex-1006, ¶30:

Each programmable logic sector 48 may include a number of programmable logic elements 50 having operations defined by configuration memory 52 (e.g., configuration random access memory (CRAM)). The programmable logic elements 50 may include combinational or sequential logic circuitry. For example, the programmable logic elements 50 may include look-up tables, registers, multiplexers, routing wires, and so forth.

Atsatt-113's teachings refer to digital circuitry, because such logic elements operate digitally—they receive digital signals and output discrete values as a function of their logical operations. It was well-known that digital circuitry operated in this manner, in comparison to analog circuitry that processed analog signals and returned results over a continuum. To the extent Patent Owner argues that this circuitry is somehow not digital, then the circuitry would be analog, which also satisfies this claim limitation.

239. Regarding claim 12, I explained above how the each of the sandboxes is part of a plurality of sandboxes. *Supra* §V.B.3.[10c] (incorporated here by reference).

240. For at least the reasons explained above, Ground 4 also teaches claims 3 and 12.

7. Dependent Claims 4 and 13

4. **The programmable device of claim 1, wherein the port comprises: a bridge being lockable to prevent reconfiguration of the bridge and prevent reconfiguration of the router while enabling reconfiguration of the one or more sandboxes.**
13. **The programmable device of claim 10, wherein the bridge is lockable to enable reconfiguration of the plurality of sandboxes, prevent reconfiguration of the bridge and prevent reconfiguration of the router.**

241. As I explained above for claims 1 and 10, Grounds 3-4 each teaches a PLD with a control region with a router, a programmable logic region with sandboxes, and a port with a bridge for locking/unlocking user access to the sandboxes. *Supra* §§V.B.2-4. It is my opinion that Atsatt-113 teaches the additional limitations of claims 4 and 13. *Infra* Appx-3 (claim appendix showing claims side-by-side).

242. Atsatt-113 teaches **wherein the port comprises: a bridge being lockable to prevent reconfiguration of the bridge and prevent reconfiguration of the router while enabling reconfiguration of the one or more sandboxes.**

For example, Atsatt-113 teaches the global resources are assigned to a base partition that is usually “configured once and not updated by partition updates” such as reconfiguration of the user partitions. Ex-1006, ¶55. Thus, the base partition, including the routing and bridge functionality in the global resources, is set and configured during initialization and then locked to prevent reconfiguration.

Ex-1006, ¶55 (“Often, they are defined by a base partition at the time the set of partitions is defined.”).

243. Because the base partition is not owned by any user, it is locked by setting a “not allowed mask[]” that prevents reconfiguration during configuration of the user partitions. Ex-1006, ¶56. I explained above how the masks are used by Atsatt-113 to lock each partition from and unlock each partition for reconfiguration. *Supra* §V.B.2.[1d] (incorporated here by reference). That analysis applies here and is incorporated by reference. Because the base partition is locked after initial setup, it is locked when user partitions are unlocked for reconfiguration. Therefore, the bridge is lockable to prevent reconfiguration of the base partition, which includes the router and bridge, while enabling reconfiguration of the sandboxes.

244. As to claim 13, I explained above how Grounds 3-4 teach a plurality of sandboxes for limitation [10c]. *Supra* §V.B.3.[10c] (incorporated here by reference). The teachings of Atsatt-113 explained for claim 4 equally apply to the case where there is a plurality of partitions, because Atsatt-113 teaches more than one user partition.

245. For at least the reasons explained above, Ground 4 also teaches claims 4 and 13.

8. Dependent Claims 5 and 14

5. **The programmable device of claim 1, wherein the port comprises: a bridge having a security layer to authenticate a user or verify a permission, to reconfigure one or more of the one or more sandboxes.**
14. **The programmable device of claim 10, wherein the bridge has a security layer to authenticate a user or verify a permission, to reconfigure one or more of the plurality of sandboxes.**

246. Claims 5 and 14 respectively depend from claims 1 and 10, and recite substantively similar language apart from claim 14 referring to the plurality of sandboxes recited in claim 10. *Infra* Appx-3 (claim appendix showing claims side-by-side). As I explained above for claims 1 and 10, Grounds 3-4 each teaches a PLD with a control region with a router, a programmable logic region with sandboxes, and a port with a bridge for locking/unlocking user access to the sandboxes. *Supra* §§V.B.2-4. It is my opinion that Atsatt-113 teaches the additional limitations of claims 5 and 14.

247. Atsatt-113 teaches **wherein the port comprises: a bridge having a security layer to authenticate a user or verify a permission, to reconfigure one or more of the one or more sandboxes**. For example, Atsatt-113 discloses a security layer (e.g., device controller 60 and authentication circuitry or engine 228) in the port and bridge functionality of the global resources that authenticates configuration data and masks to verify permission to configure the partitions. Ex-1006, ¶62 (“The device controller 60 may use authentication circuitry or engine

228, which may be communicatively coupled to the device controller 60, to authenticate the configuration data 224 with the configuration data authentication key 220 and the mask 226 with the mask authentication key 222.”). Atsatt-113 explains that “[t]he authentication circuitry 228 may use any suitable techniques to authenticate the configuration data 224 and the mask 226.” Ex-1006, ¶62. As an example, Atsatt-113 suggests using “a secure hash algorithm (e.g., SHA-1).” Ex-1006, ¶62.

248. Figure 16 shows the flow of the authentication process, from receiving configuration data and keys (step 382) up to reconfiguring the designated user partition (step 398). Ex-1006, ¶93 (“FIG. 16 is a flow diagram illustrating a process 380 for partially reconfiguring a programmable fabric while maintaining reconfigurable partitions using the system 210 of FIG. 9, according to an embodiment of the present disclosure.”), Fig. 16:

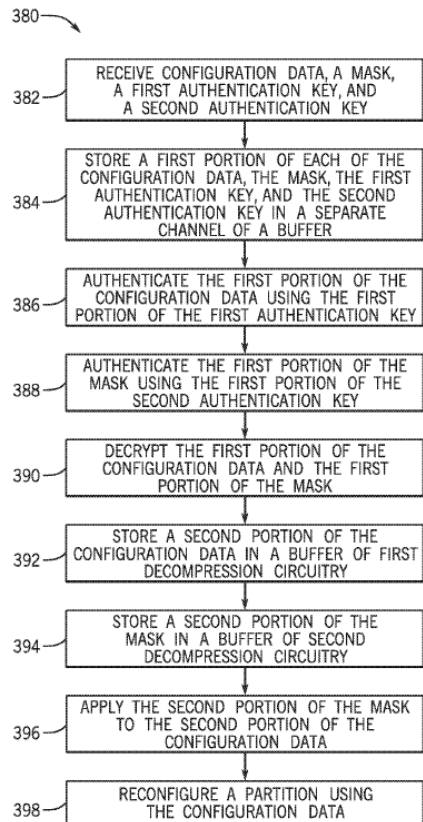


FIG. 16

249. First, “[t]he device controller 60 may receive (process block 382) configuration data 224, a mask 226, a first authentication key (e.g., a configuration data key 220) and a second authentication key (e.g., a mask key 222). The device controller 60 may store (process block 384) at least a first portion of each of the configuration data 224, a mask 226, the first authentication key 220, and the second authentication key 222 in a separate channel 218 of a buffer 216.” Ex-1006, ¶94.

250. At steps 386-388, “[t]he device controller 60 may then authenticate (process block 386) at least the first portion of the configuration data 224 using at

least the first portion of the first authentication key 220” (Ex-1006, ¶95) and “may also authenticate (process block 388) at least the first portion of the mask 226 using at least the first portion of the second authentication key 222” (Ex-1006, ¶96). While Atsatt-113 refers to the device controller 60 performing these steps in connection with Figure 16, the teachings equally apply to authentication circuitry 228 because Atsatt-113 explains that the device controller uses the authentication circuitry to that end. Ex-1006, ¶62. Indeed, Atsatt-113 states that this security layer functionality can “be implemented by the device controller 60 and/or one or more sector controllers 58 of the FPGA 40.” Ex-1006, ¶93.

251. Functionality in the interconnection resources use Atsatt-113’s security functionality to enable and disable access to particular programmable logic sectors. As explained for claim 1, the authentication of the mask is part of the process that the interconnection resources use to lock or unlock access to programmable logic sectors. *E.g.*, Ex-1006, ¶81, ¶¶41-42, ¶¶59-69; *supra* §V.B.2.[1d] (incorporated here by reference).

252. As to claim 13, I explained above how Atsatt-113 teaches a plurality of sandboxes for limitation [10c]. *Supra* §V.B.3.[10c] (incorporated here by reference). The teachings of Atsatt-113 explained for claim 5 equally apply to the case where there is a plurality of partitions, because Atsatt-113 teaches more than one user partition.

253. For at least the reasons explained above, Ground 4 also teaches claims 5 and 14.

9. Dependent Claims 9 and 18

- 9. The programmable device of claim 1, wherein the one or more sandboxes comprises a plurality of sandboxes that are logically or physically combinable to form a larger sandbox.**
- 18. The programmable device of claim 10, wherein the plurality of sandboxes is logically or physically combinable to form a single, larger sandbox.**

254. Claims 9 and 18 respectively depend from claims 1 and 10, and recite substantively similar language. Additionally, claim 18 recites a *single*, larger sandbox. *Infra* Appx-3 (claim appendix showing claims side-by-side). Ground 4 teaches the additional limitations of claims 9 and 18. As I explained above for claims 1 and 10, a POSITA would have been motivated to combine Biederman's teachings of adjusting partition sizes with Atsatt-113's teachings of sandboxed partitions.

255. Biederman teaches the limitations of claims 9 and 18, as explained above for Ground 2. Those same teachings apply here for Ground 4. *Supra* §V.A.12 (incorporated by reference).

256. **Motivation to Combine.** A POSITA would have been motivated to combine Atsatt-113's teachings of a reconfigurable FPGA with separate user partitions and Biederman's teachings of adjusting partition sizes because it would

apply a known technique to a known device ready for improvement to yield predictable results. Biederman teaches a known technique to adjust the size of partitions on an FPGA. Ex-1005, ¶134. Atsatt-113 is ready for improvement because it already teaches an FPGA with multiple partitions that are reconfigurable, and the partitions are loaded with various circuits that may be smaller or larger than the initially assigned partition size. Ex-1006, Abstract, ¶¶36-38, ¶¶57-59, Fig. 3. The combination yields the predictable results of adjusting partition sizes to fit the logic loaded on the partition.

257. A POSITA would also have been motivated because the combination provides a number of benefits. By sizing partitions to the logic to be hosted, it adds flexibility to the partitioning of logic sectors in the FPGA. This allows for users to use larger logic designs, increasing user choice and satisfaction. This also contributes to maximizing the utilization of logic sectors in the FPGA, which provides a benefit to the cloud service provider. This reduces the cost of equipping, maintaining, and running data centers with FPGA-equipped servers. See Ex-1026, 000002-4 (discussing efficiency of sharing resources for cloud computing). Lowering costs and reducing the footprint of FPGA-equipped servers has the additional benefit of expanding the pool of potential customers increases because it creates lower-cost options and allows users developing smaller logic circuits to share an FPGA, reducing the entry barrier.

258. A POSITA would have had a reasonable expectation of success when combining the complementary teachings of Atsatt-113 and Biederman. The combination would not have changed the principles of operation. For example, Atsatt-113's teachings of partitioning and reconfiguring partitions in the programmable logic of an FPGA would still be used to partition and reconfigure user partitions. Biederman's teachings of adjusting partition sizes, including combining partitions, would still be used to adjust the size of partitions. The operation of the programable logic sectors, interconnects, and other elements of the FPGA in the combination would perform in the same way as they do in Atsatt-113 and in Biederman.

259. Atsatt-113 and Biederman teach complementary teachings related to the composition and operation of FPGAs. The combination adds a few minor details from Biederman that are compatible with Atsatt-113's teachings. A POSITA would have found the combination obvious and would have had a reasonable expectation of success in doing so. Indeed, a POSITA would have understood Biederman to teach the obvious result of Atsatt-113's teachings of reconfiguring partitions with different logic that may be larger than the previously loaded logic. A POSITA with 2 years of experience in the field would have been fully capable of making the combination. *Supra* §III.

C. Ground 5

260. Tarafdar discloses analytics and management tools for FPGAs and teaches the limitations of dependent claims 6 and 15. Ground 5 adds Tarafdar's teachings to each of Grounds 1-4. The motivation to combine is explained in detail below.

1. Tarafdar et al., "Enabling Flexible Network FPGA Clusters in a Heterogeneous Cloud Data Center" ("Tarafdar") (Ex-1007)

261. I have reviewed the Tarafdar reference. I understand that Tarafdar was not cited or considered during prosecution of the '943 Patent, based primarily on the fact that Tarafdar is not cited on the face of the patent, nor have I seen the reference discussed in the prosecution history.

262. Tarafdar is a conference paper that was published February 2017 in Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. Ex-1022, ¶¶9, 11. Based on the priority date of July 15, 2018 of the '943 patent, I understand Tarafdar is prior art to the '943 patent under at least post-AIA §102(a)(1).

263. A declaration from the publisher ACM confirms that Tarafdar is a printed prior art publication. See Ex-1022. Tarafdar was published in the conference proceedings and also online on the ACM Digital Library by February 22, 2017. Ex-1022, ¶7, ¶10. The print copy was publicly accessible because it was

published and conference proceedings are customarily distributed to the interested public during the conference for which it is prepared. The digital copy was also publicly accessible on the Internet. The ACM Digital library was and is searchable using keywords. Ex-1022, ¶10. In my own research, I have for many years used the search functionality of the ACM Digital Library to locate relevant papers based on subject matter or authors. It is a well-known depository of publications that is frequented by the interested public, including professors, researchers, and students around the world.

264. Tarafdar presents “a framework for creating network FPGA clusters in a heterogeneous cloud data center.” Ex-1007, Abstract. Tarafdar teaches the use of management tools to monitor and orchestrate virtual machines and their underlying processing resources in data centers. Ex-1007, §2.2 at 000002.

Tarafdar presents analytics generated from Tarafdar’s “own orchestration tool” (Ex-1007, §2.2 at 000002) that evaluate the performance of FPGA cluster designs in terms of resource overhead and throughput. Ex-1007, §7 at 000007-9:

Table 1: Resource Overhead of our System

Hardware Setup	LUTS	Flip-Flops	BRAM
SDAccel Base	53346 (12.3 %)	64550 (7.45 %)	228 (15.5 %)
SDAccel Base with Ethernet Support	62344 (14.4 %)	76124 (8.79 %)	228 (15.5 %)
Input Module			
Input Bridge	87 (0.02 %)	170 (0.019 %)	2 (1.36 %)
Input Demultiplexer with 16 outputs	82 (0.019 %)	124 (0.014 %)	0 (0 %)
Output Module			
Ethernet FIFO Controller	26 (0.006 %)	12 (0.014 %)	2 (1.36 %)
Output Switch with 16 inputs	517 (0.119 %)	138 (0.016 %)	0 (0 %)
Packet Formatter (one per network output stream)	230 (0.053 %)	252 (0.029 %)	2 (1.36 %)
Total Available	433200	866400	1470

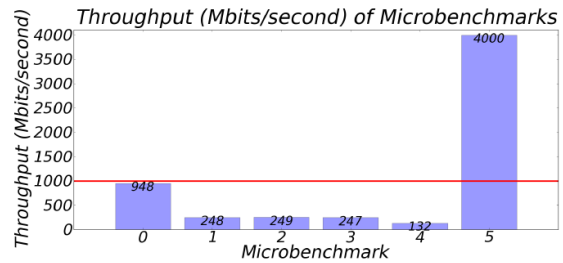


Figure 17: Throughput of the four microbenchmarks with two additional datapoints.

Ex-1007, Table 1 at 000008.

Ex-1007, Fig. 17 at 000009.

265. Tarafdar is analogous art to the '943 patent because it is in the same field of endeavor as the '943 patent, which is PLDs and data processing systems. *Supra* §III. Tarafdar explores FPGA-based solutions for data processing in heterogeneous data centers. Ex-1007, Abstract. In particular, Tarafdar tests its techniques on a database acceleration application. *Id.* Tarafdar is reasonably pertinent to the problem addressed by the '943 patent's inventor because it seeks to optimize and reconfigure programmable clusters using PLDs such as FPGAs. Ex-1007, §1 at 000001.

2. Dependent Claims 6 and 15 are Rendered Obvious in View of Tarafdar in combination with Grounds 1-4.

- 6. The programmable device of claim 1, further comprising: the router configured to perform analytics on resources of the programmable device, on a per module basis, including determining communication bandwidth and resource utilization.**
- 15. The programmable device of claim 10, further comprising: the router configured to perform analytics on resources of the programmable device, on a per module basis, including determining communication bandwidth and resource utilization in the plurality of sandboxes.**

266. Claims 6 and 15 each respectively depend from claims 1 and 10, but otherwise recite substantively similar language. *Infra* Appx-3 (claim appendix showing claims side-by-side).

267. Tarafdar teaches claim 6. The overlapping limitations of claim 15 are satisfied for the same reasons. Tarafdar further teaches the additional limitation of claim 15 regarding the “each of the plurality” of sandboxes. I explained above how Grounds 1-4 teach the plurality of sandboxes for claim 10. *Supra* §V.A.4 (incorporated by reference), §V.B.3 (incorporated by reference).

268. As I explained above for claims 1 and 10, Grounds 1-4 each teaches a PLD with a first region with a router, a second region with sandboxes, and a port with a bridge for locking/unlocking user access to the sandboxes. *Supra* §§V.A.3-4, §§V.B.2-3.

269. Tarafdar teaches **the router configured to perform analytics on resources of the programmable device, on a per module basis**. For example,

Tarafdar discloses “Cloud Cluster Management Tools” that perform analytics on FPGA resources to adjust the deployment of FPGA modules in cloud clusters. Ex-1007, §2.2 at 000002. Tarafdar builds on prior art techniques such as OpenStack that analyze “metrics such as performance, *resource utilization*, and CPU usage.” *Id.* Tools for “orchestration and load balancing using the aforementioned metrics” were known and used in cloud computing environments. *Id.* Tarafdar performs analytics by determining the utilization of different resources and making decisions to adjust the configuration of the cluster based on these metrics.²⁶

270. Tarafdar teaches that the router (in the first region) is configured to perform these analytics. First, like Seshadri and Atsatt-113, Tarafdar discloses a wrapper region, which Tarafdar calls a “hypervisor,” between the FPGA device interface (e.g., PCIe) and the user-configurable region of the FPGA, for routing

²⁶ The ’943 patent confirms that monitoring resource utilization and making decisions based on the monitored statistics falls within the scope of the recited “perform[ing] analytics.” The specification states that “the MMI router 124 gathers statistics such as communication bandwidth, possibly on a per module basis for analysis, telemetry, decision making, etc.” Ex-1001, 5:11-18. The patent provides examples of analytics that include “determining how close the system is to full capacity of the resources or full capacity of the router itself.” *Id.*

communication to and from the user-configurable application region and managing the region. Ex-1007, §3.2 at 000003, Fig. 2:

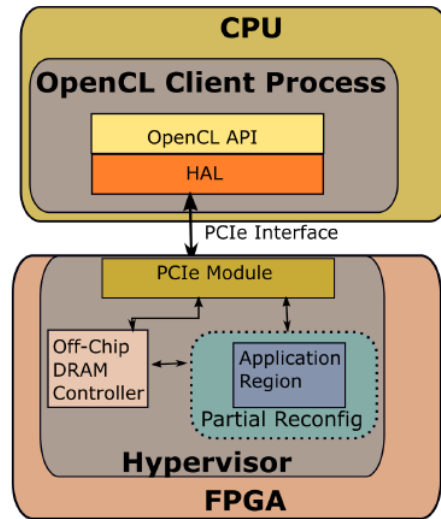


Figure 2: System diagram of the SDAccel platform.

271. Tarafdar monitors and analyzes the resources at the router, in this Hypervisor region, because Tarafdar “monitor[s] and updat[es] our clusters using FPGA metrics.” Ex-1007, §2.2 at 000002. Thus, Tarafdar teaches monitoring and analyzing metrics on the FPGA, which naturally would have been implemented in the Hypervisor region. A POSITA would have found this obvious because metrics are used for managing the FPGAs, and the Hypervisor region carries out the OpenCL API calls received from the host to manage the FPGAs, such as for programming the hosted logic. Ex-1007, §3.2 at 000003. Moreover, a POSITA would have recognized benefits in carrying out such analytics in a management region on the FPGA, with the most direct and immediate access to overall FPGA

metrics and resources, thereby allowing orchestration to be based on the most accurate and current data. And Tarafdar teaches the collection of metrics internal to the FPGA. Ex-1007, §7.3.2 at 000009 (microbenchmark 5 collecting “internal FPGA throughput”). This would also reduce potential overhead of transmitting a continual stream of statistical data over the network for every FPGA in the platform, allowing network traffic to be reduced to aggregate or key statistics for each module, or triggers that prompt further action at managing nodes, to increase the overall efficiency of the platform. Moreover, distributed computing servers are complex systems with a large number of CPUs, coprocessors (e.g., vector/SIMD, GPU), and FPGAs. Measuring and optimizing performance of such large, complex, expensive systems requires access to utilization, communication bandwidth, and other performance data from throughout the system. This information was conventionally available from CPUs and coprocessors, and a POSITA would look for similar data from FPGAs.

272. Tarafdar teaches performing monitoring and analytics on a per-module basis. For example, Tarafdar monitors network throughput and resource overhead (Ex-1007, §7.3.2 at 000009, §7.1 at 000008) for the “orchestration of clusters” (Ex-1007, §2.2 at 000002). Therefore, Tarafdar teaches, or at least suggests, performing this on a per-module basis because the availability of network throughput and resource overhead for each module facilitates orchestration of

modules within the clusters by allowing modules to be assigned tasks based on their current status, which is one of Tarafdar’s stated goals. Ex-1007, §1 at 000001. This granularity, in comparison to system-wide analysis, would have allowed for more efficient use of FPGA resources in the data center environment of Atsatt-113, Seshadri, and Biederman. Additional details are explained below for particular metrics.

273. Tarafdar teaches **determining communication bandwidth**, including network throughput determined for each microbenchmark in Figure 17. Ex-1007, Fig. 17 at 000009:

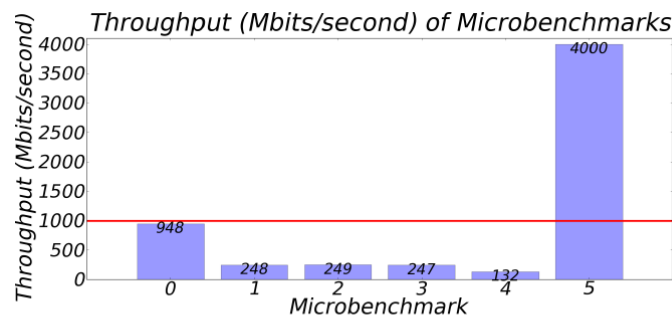


Figure 17: Throughput of the four microbenchmarks with two additional datapoints.

Tarafdar demonstrates measuring the throughput using the “iperf tool.” Ex-1007, §7.3.2 at 000009. Iperf is a popular open-source utility that is available for Linux, Unix, and Window systems, among others. As Tarafdar states, iperf “is a network tool used to measure throughput of network connections.” *Id.*

274. Tarafdar “test[s] the latency and throughput of the input and output modules using a set of microbenchmarks.” Ex-1007, §7 at 000008. The

“microbenchmarks consist of an application that is a direct connection between the Input Module and the Output Module of an Application Region.” Ex-1007, §7.2 at 000009. The Application Region is the user-configurable area of the FPGA that is apart from and managed by the hypervisor. Ex-1007, 3.2 at 000005, Fig. 2. Microbenchmarks are carried out for different combinations of modules. Ex-1007, §7.3 at 000009.

275. Microbenchmarks 4 and 5 determine communication bandwidth for each FPGA. Microbenchmark 4 “is the throughput between two virtual machines in the SAVI network”; microbenchmark 5 is the “calculated throughput within the FPGA.” Ex-1007, §7.3 at 000009. Tarafdar explains that “the internal FPGA throughput is calculated by using the bus width, which is 4-bytes wide and multiplying that by the clock speed, which is 125 MHz.” *Id.* Therefore, since microbenchmark 5 calculates throughput for each FPGA, it calculates communication bandwidth on a per-FPGA module basis. *Id.*

276. Additionally, Microbenchmark 1 provides measurements for a single FPGA module, while Microbenchmarks 2 and 3 includes measurements of multiple FPGA modules. Ex-1007, Fig. 15(b) at 000009:

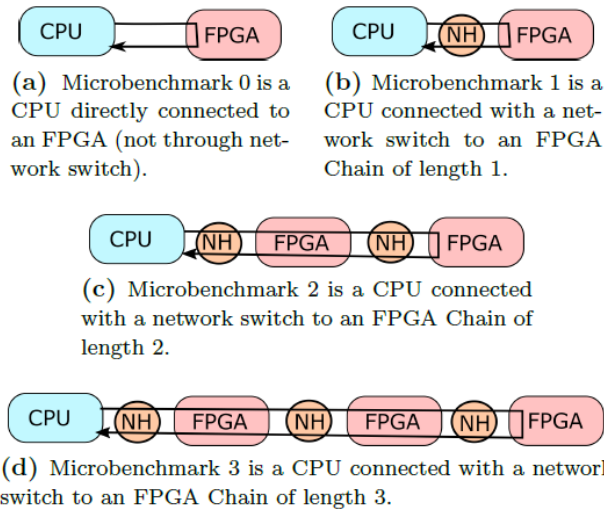


Figure 15: Microbenchmarks 1 to 3 have a network hop (NH). Each network hop travels to the network switch connected to all the FPGAs. Microbenchmark 0 does not use a virtualized CPU, where as the others use virtual CPUs provisioned in SAVI.

277. Because microbenchmark 1 includes a single network hop between the CPU and the FPGA module, microbenchmark 1 teaches measuring the communication bandwidth for a single module. The microbenchmarks use a host CPU as an intermediary as part of an experiment measuring the efficiency gains of chaining multiple FPGAs together. Nonetheless, these portions of Tarafdar teach the measurement of communication bandwidth, and Tarafdar also includes more general teaching and suggestion to monitor and analyze metrics on the FPGA, as I explained above. *Supra* ¶271. Thus, in an actual deployment beyond experimental tests, Tarafdar suggests, carrying out the determination of network throughput at the FPGA (for example in the Hypervisor); this would have been obvious for the reasons I explained above.

278. Tarafdar teaches **determining resource utilization**, including “resource overhead” in Table 1. Tarafdar, Table 1 at 000009:

Table 1: Resource Overhead of our System

Hardware Setup	LUTS	Flip-Flops	BRAM
SDAccel Base	53346 (12.3 %)	64550 (7.45 %)	228 (15.5 %)
SDAccel Base with Ethernet Support	62344 (14.4 %)	76124 (8.79 %)	228 (15.5 %)
Input Module			
Input Bridge	87 (0.02 %)	170 (0.019 %)	2 (1.36 %)
Input Demultiplexer with 16 outputs	82 (0.019 %)	124 (0.014 %)	0 (0 %)
Output Module			
Ethernet FIFO Controller	26 (0.006 %)	12 (0.014 %)	2 (1.36 %)
Output Switch with 16 inputs	517 (0.119 %)	138 (0.016 %)	0 (0 %)
Packet Formatter (one per network output stream)	230 (0.053 %)	252 (0.029 %)	2 (1.36 %)
Total Available	433200	866400	1470

As seen in Table 1, Tarafdar determines the “absolute numbers” for each metric as well as “the percentage of the device total” for metrics including utilization of resources such as LUTs (lookup tables), flip-flops, and BRAM (block RAM) in the FPGA for different logic functions and partitions, such as “SDAccel Base” or “Ethernet FIFO Controller,” implemented on the FPGA module. *See* Ex-1007, §7.1 at 000009.

279. As to claim 15, Tarafdar further teaches performing analytics for the resources **in the plurality of sandboxes**. For example, Tarafdar teaches partitioned FPGAs like that of Seshadri, Atsatt-113, and Biederman—partitioning the programmable logic into “four smaller regions allowing multiple users to share

a single FPGA device.” Ex-1007, §2 at 000003. When the Application Region is partitioned into four smaller user regions, performing the above-described analytics on the FPGA includes doing so in the plurality of user regions. Ex-1007, §7 at 000007-10.

280. **Motivation to Combine with Seshadri Grounds (1-2).** As I explained above, Seshadri and Biederman teach known techniques and uses of PLDs, including FPGAs, in data centers. *Supra* §V.A. A POSITA would have been motivated to further apply Tarafdar’s teachings because Seshadri suggests “various modifications and changes” with an “intention to cover all modifications” to its teachings (Ex-1004, 29:62-30:11) and Tarafdar similarly teaches techniques for use in data center/cloud computing architectures and virtual machines. Tarafdar teaches known techniques for monitoring and analyzing the resource utilization and network bandwidth of FPGAs in cloud computing data centers. Ex-1007, §2 at 000002, §7 at 000007-10. Applying Tarafdar’s monitoring and analysis to the FPGA architecture of Seshadri (alone or in view of Biederman) would have yielded the predictable result of monitoring and analyzing resource utilization for each FPGA module. Thus, the combination of Tarafdar with either Ground 1 or 2 would have been a simple combination of prior art elements according to known methods to yield predictable results.

281. A POSITA would have been motivated to apply Tarafdar's teachings to Seshadri alone and in combination with Biederman to take advantage of several benefits. For example, Tarafdar "provide[s] an easy way to orchestrate large FPGA clusters" (Ex-1007, §1 at 000001) where "monitoring is used for user statistics as well as fault tolerance within the cluster." *Id.*, §2.2 at 000002. The use of such techniques "are helpful for getting reliable performance on a cluster as well as debugging a cluster." *Id.* A POSITA would have recognized such benefits and would have been motivated to apply Tarafdar's teachings to distributed computing systems using FPGAs, such as those taught by Seshadri and Biederman, in order to better administrate the nodes in the cloud and to increase FPGA usage efficiency. *See id.*

282. A POSITA would have had a reasonable expectation of success in combining Seshadri's and Biederman's teachings with those of Tarafdar. The combinations would not have changed the principle of operation of any of the references because it simply applies Tarafdar's FPGA-monitoring and analyzing teachings to Seshadri alone and in combination with Biederman, which likewise teach FPGAs. Therefore, the combination simply uses each reference for its intended purpose.

283. A POSITA would have found the combinations obvious and would have had a reasonable expectation of success in doing so, because the

combinations are relatively simple as I just explained. Tarafdar demonstrates its teachings applied to an FPGA, and a POSITA would have understood how to apply it to other FPGAs. Thus, a POSITA with 2 years' experience (*supra* §III) would have been fully capable of combining the teachings of the different references in the manner I explained above.

284. **Motivation to Combine with Atsatt Grounds (3-4).** As I explained above, Atsatt-113 and Biederman teach known techniques and uses of PLDs, including FPGAs, and their combination applies those teachings to data centers. *Supra* §V.B. A POSITA would have been motivated to further apply Tarafdar's teachings because Tarafdar teaches known techniques for monitoring and analyzing the resource utilization and network bandwidth of FPGAs in cloud computing data centers. Ex-1007, §2 at 000002, §7 at 000007-10. Applying Tarafdar's monitoring and analysis to the FPGA architecture of Atsatt-113 (alone or in view of Biederman) would have yielded the predictable result of monitoring and analyzing resource utilization for each FPGA module. Thus, the combination with either Ground 3 or 4 would have been a simple combination of prior art elements according to known methods to yield predictable results.

285. A POSITA would have been motivated to apply Tarafdar's teachings to Atsatt-113 alone and in combination with Biederman to take advantage of several benefits. For example, Tarafdar "provide[s] an easy way to orchestrate

large FPGA clusters.” Ex-1007, §1 at 000001. Tarafdar states “monitoring is used for user statistics as well as fault tolerance within the cluster.” *Id.*, §2.2 at 000002. The use of such techniques “are helpful for getting reliable performance on a cluster as well as debugging a cluster.” *Id.* A POSITA would have recognized such benefits and would have been motivated to apply Tarafdar’s teachings to other distributed computing systems, such as those of Atsatt-113 and Biederman, using FPGAs in order to better administrate the nodes in the cloud and increase FPGA usage efficiency.

286. Beyond the cloud, Tarafdar’s teachings would have been advantageous for optimizing FPGA utilization within a computer system such as a server. Since FPGAs were often provided on PCIe cards, multiple FPGA cards were easily added to a single computer. Ex-1007, §5 at 000004, Fig. 4; Ex-1005, ¶52, ¶54, Fig. 1; Ex-1006, ¶39, ¶55, Fig. 2. Tarafdar’s analytics include internal FPGA metrics that report internal throughput and utilization, both of which were useful for optimizing FPGA utilization and workloads within a single computer system. Ex-1007, §2.2 at 000002. Thus, a POSITA would have been motivated to apply Tarafdar’s teachings to Atsatt-113 alone and in combination with Biederman in data centers, for reasons explained above, but also for single-computer applications.

287. A POSITA would have had a reasonable expectation of success in combining Atsatt-113's and Biederman's teachings with those of Tarafdar. The combinations would not have changed the principles of operation of any reference because it simply applies Tarafdar's FPGA-monitoring and analyzing teachings to Atsatt-113 alone and in combination with Biederman, which likewise teach FPGAs. Therefore, the combination simply uses each reference for its intended purpose.

288. A POSITA would have found the combinations obvious and would have had a reasonable expectation of success in doing so, because the combinations are relatively simple as I just explained. Tarafdar demonstrates its teachings applied to an FPGA, and a POSITA would have understood how to apply it to other FPGAs. Thus, a POSITA with 2 years' experience (*supra* §III) would have been fully capable of combining the teachings of the different references in the manner I explained above.

D. Ground 6

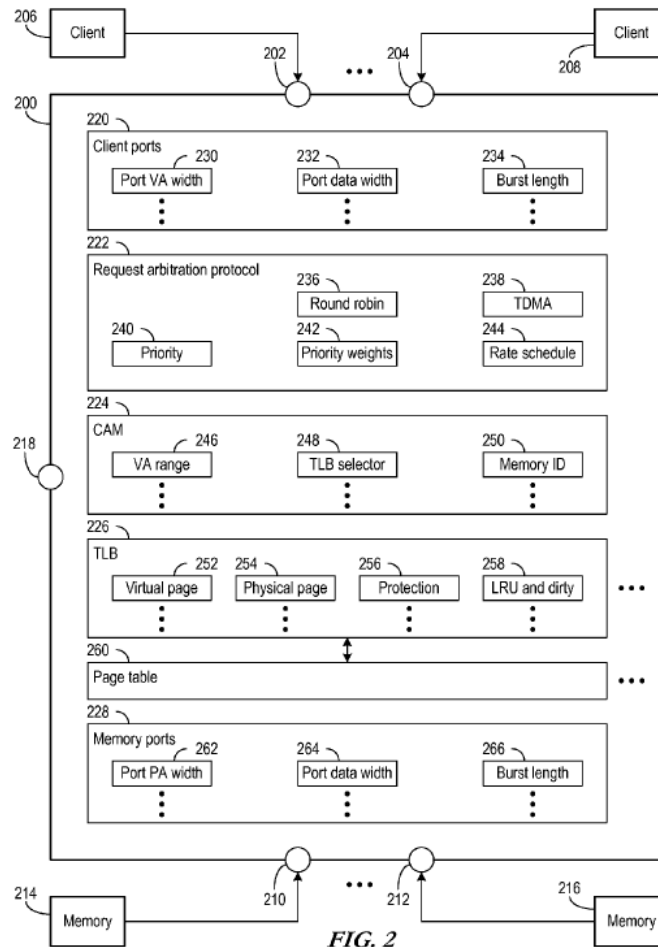
289. Kulkarni discloses configurable ports and teaches the limitations of claims 7 and 16. Ground 6 adds Kulkarni's teachings to each combination of Grounds 1-4. The motivation to combine is explained in detail below.

1. U.S. Patent No. 8,015,386 (“Kulkarni”) (Ex-1008)

290. I have reviewed the Kulkarni reference. I understand that Kulkarni was not cited or considered during prosecution of the '943 Patent, based primarily on the fact that Kulkarni is not cited on the face of the patent, nor have I seen the reference discussed in the prosecution history.

291. Kulkarni is entitled “Configurable memory manager.” Ex-1008, (54) Title. Kulkarni was filed on March 31, 2008 and published on September 6, 2011. Based on the priority date of July 15, 2018 of the '943 patent, I understand Kulkarni is prior art to the '943 patent under post-AIA §§102(a)(1)-(2).

292. Kulkarni discloses “[a] configurable memory manager [that] is configurable with various configuration parameters.” Ex-1008, (57) Abstract. The memory manager is intended to be used in PLDs such as FPGAs, to allow client logic in the PLD to access shared memory. Ex-1008, 1:12-54, Fig. 2:



293. Various parameters of the ports are configurable, including the data width. Ex-1008, 2:25-37 (“Various parameters, such as the data widths of ports 102 through 104 and 106 through 108, are configurable in the configurable memory manager 100.”).

294. Kulkarni is analogous art to the ’943 patent because it is in the same field of endeavor as the ’943 patent, which is PLDs and data processing systems. *Supra* §III. Kulkarni aims to improve the “performance of multiprocessing system[s]” implemented in a PLD. Ex-1008, 1:39-47. Kulkarni is reasonably

pertinent to the problem addressed by the '943 patent's inventor, because its techniques are applicable to the problem of interfacing communication between partitions within a field programmable logic device.

2. Dependent Claims 7 and 16 are Rendered Obvious in View of Kulkarni in combination with Grounds 1-4.

- 7. The programmable device of claim 1, wherein the one or more sandboxes each comprise port interfaces that are configurable as to data width or speed.**
- 16. The programmable device of claim 10, wherein the each of the plurality of sandboxes comprises one or more port interfaces configurable as to data width or speed.**

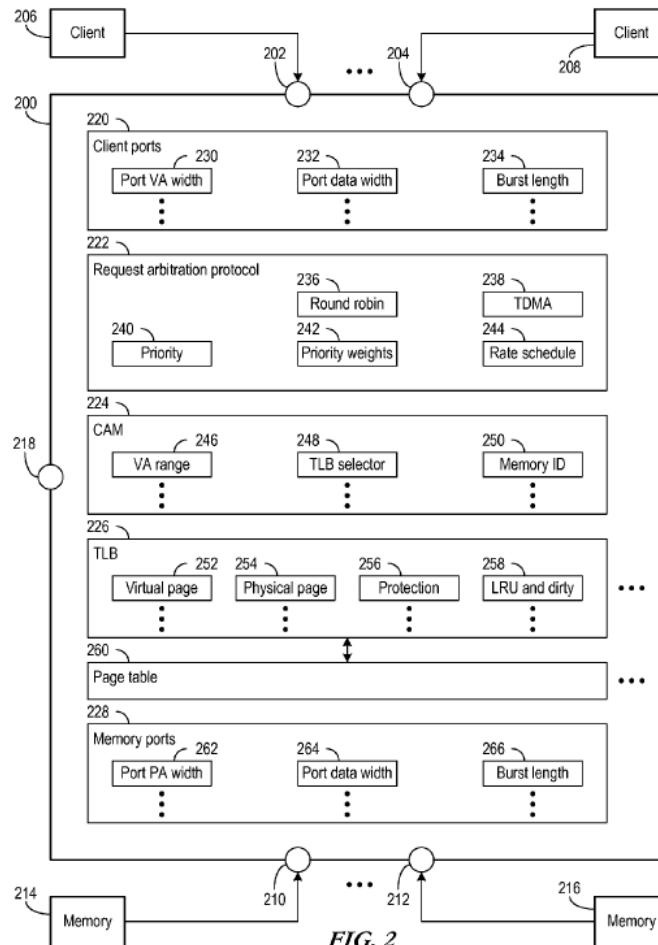
295. Claims 7 and 16 each respectively depend from claims 1 and 10, but otherwise recite substantively similar language. *Infra* Appx-3 (claim appendix showing claims side-by-side).

296. Kulkarni teaches claim 7. The overlapping limitations of claim 16 are satisfied for the same reasons. Kulkarni further teaches the additional limitation of claim 16 regarding “each of the plurality” of sandboxes. I explained above how Grounds 1-4 teach the plurality of sandboxes for claim 10. *Supra* §V.A.4, §V.B.3 (incorporated by reference). Kulkarni teaches ports for multiple client sandboxes as I explain below, and therefore the teachings discussed for claim 7 equally apply to each of a plurality of sandboxed user partitions. Ex-1008, Fig. 2.

297. As I explained above for claims 1 and 10, Grounds 1-4 each teaches a PLD with a first region with a router, a second region with sandboxes, and a port

with a bridge for locking/unlocking user access to the sandboxes. *Supra* §§V.A.3-4, §§V.B.2-3.

298. Kulkarni teaches **wherein the one or more sandboxes each comprise port interfaces that are configurable as to data width**. For example, Kulkarni discloses a configurable memory manager to interface between the programmable logic areas and memory in an FPGA. Ex-1008, Abstract (“A configurable memory manager is configurable with various configuration parameters. The configurable memory manager has client ports for receiving requests for accessing memories and memory ports for accessing respective memories.”), Fig. 2. Kulkarni teaches providing each client-configurable area of the FPGA (which Kulkarni calls “clients”) with a port so the-client configurable areas each comprise a port interface to the configurable memory manager:



299. Kulkarni teaches including ports on an FPGA (including memory manager ports and the corresponding client ports on the other side of the link) that are “independently configurable to specify the parameter of a data width of the port.” Ex-1008, Abstract, 2:25-37 (“Various parameters, such as the data widths of ports 102 through 104 and 106 through 108, are configurable in the configurable memory manager 100.”).

300. Figure 1 is a block diagram showing the memory manager (and the connecting ports from the client-configurable areas) in closer detail. Ex-1008, 2:25-37, Fig. 1:

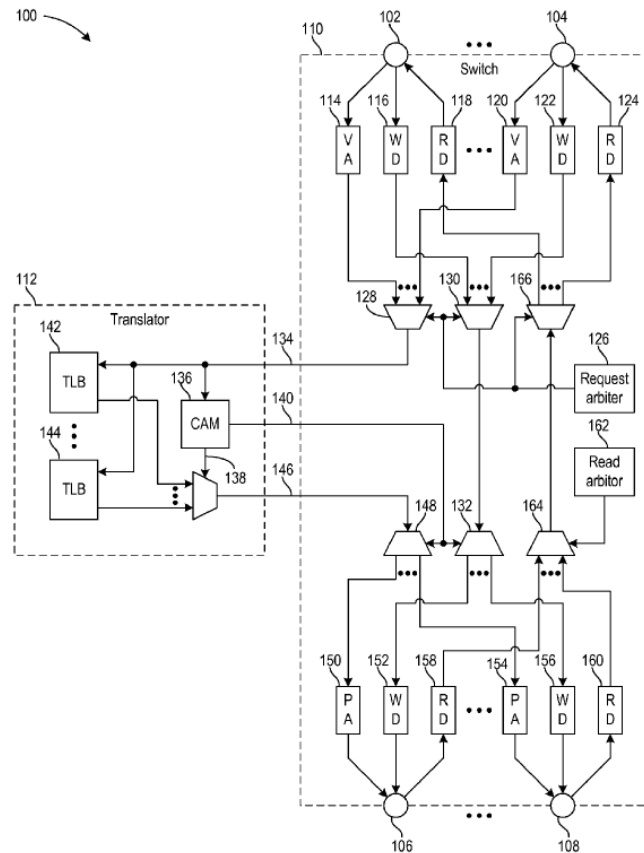


FIG. 1

Kulkarni explains “processing elements” in programmable logic sectors are “connected to client ports 102 through 104 to access storage in memories connected to memory ports 106 through 108.” Ex-1008, 2:25-37. Switches are used to “transfer[] memory access requests between the client ports 102 through 104 and...the appropriate one of memory ports 106 through 108.” Ex-1008, 2:25-37, 1:58-2:3 (“The configurable memory manager includes a switch and a

translator. The translator translates a virtual address in each of the requests into an identifier of one of the memories and a physical address in the memory. The switch transfers each request from the client port receiving the request to the memory port for accessing the memory identified by the identifier for the virtual address in the request.”).

301. With respect to Figure 2, the corresponding client ports 202-204 are configured with parameters to configure “the width of read data and write data” and the “the burst length of read data and write data transferred.” Ex-1008, 4:26-42 (“For example, parameter 234 sets the number of consecutive data cycles needed to transfer read and write data and parameter 232 sets the number of data bits transferred in each data cycle.”). Similarly, parameters can be used to configure the data width and burst length for memory ports 210-212. Ex-1008, 6:22-27 (“Parameter 262 configures the width of the physical address for memory port 210, parameter 264 configures the data width for memory port 210, and parameter 266 configures the data burst length for memory port 210. Memory port 212 is similarly configurable by parameters 228.”). Since the ports on the configurable memory manager are connected to a corresponding port on the client, when Kulkarni adjusts the width of the port on one side, it adjusts the width of the port for the other. *See id*, Ex-1008, 4:26-42, Fig. 2. Therefore, each client sandbox port has a configurable data width. *See id*.

302. **Motivation to Combine with Seshadri Grounds (1-2).** As I

explained above, Seshadri and Biederman teach known techniques and uses of PLDs, including FPGAs. *Supra* §V.A. A POSITA would have been motivated to further apply Kulkarni's teachings because Seshadri suggests "various modifications and changes" with an "intention to cover all modifications" to its teachings (Ex-1004, 29:62-30:11) and Kulkarni suggests using its configurable memory manager in PLDs, including FPGAs. Ex-1008, 1:11-54. In particular, Kulkarni notes that "[i]t may be time consuming and expensive to design [custom] protocols for exchanging data between the processors of a multiprocessing system" implemented on a PLD. Ex-1008, 1:48-54. Therefore, Kulkarni's teachings were readily applicable to the PLD teachings of Seshadri and Biederman.

303. A POSITA would have been motivated to use Kulkarni's teachings of a configurable memory manager and ports in other PLDs, such as those taught by Seshadri and Biederman, because Kulkarni provides a powerful and flexible solution for organizing communication between different parts of a PLD. Kulkarni's configurable memory manager is particularly well-suited for FPGAs hosting a variable number of user partitions that can be configured with circuitry of varying specifications because it allows communication bandwidth to be dynamically tailored to the capacity and needs of each user's custom logic. Ex-1008, 2:25-37, 4:26-42, 6:22-27. Therefore, a POSITA would have been

motivated to apply Kulkarni's teachings to adjust the data width of ports in each sandbox taught by Seshadri alone and in combination with Biederman based on the capacity and needs of the user's custom logic. *See id.*

304. Moreover, Seshadri and Kulkarni both teach an FPGA with an area that is client-configurable. Therefore, a POSITA would have been motivated to apply Kulkarni's teachings regarding "client" areas in the FPGA to Seshadri's teachings regarding client-configurable circuits because it would have used Kulkarni's configurable memory manager to improve similar devices in the same way. Ex-1008, 2:25-37, 4:26-42, 6:22-27.

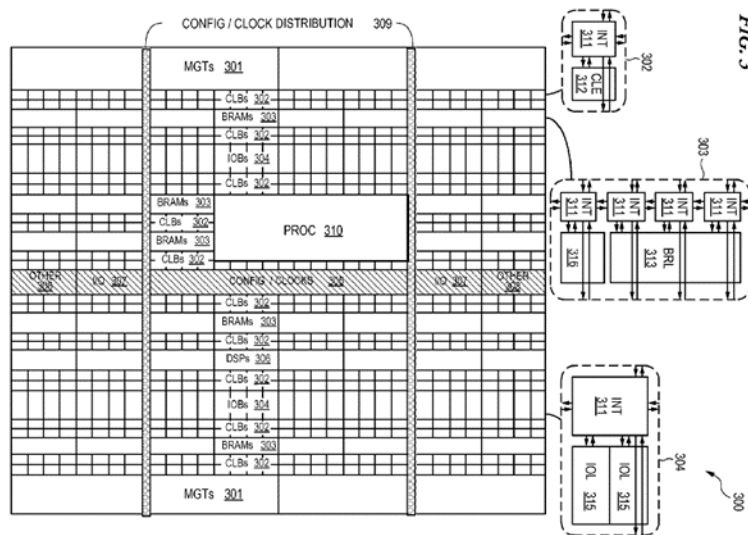
305. Additionally, Seshadri provides motivation to combine its teachings with hard macros for communication ports. Ex-1004, 10:48-65 ("Hard macros for other types of communication ports"). Macros were used to import a design into an IC. Kulkarni teaches a hard macro for communication ports that is operable with various configurations of interconnect resources in FPGAs, without the need to replace or modify them. Ex-1008, 1:56-3:5, 2:58-3:12. Seshadri teaches the use of additional circuitry to carry out specific functions, known in the art as hard macros, and thus Seshadri is designed to support additional circuitry, for example as taught by Kulkarni, for communication ports. Ex-1004, 10:48-65; *supra* §I.A.

306. Indeed, a POSITA looking at communication ports would have looked to Kulkarni because Kulkarni is a patent filed by Xilinx, which was the first

company to introduce commercial FPGAs and has long commanded around half of the global market for FPGAs. A POSITA would have looked to publications embodying Xilinx's research efforts, such as patents like Kulkarni, for advanced technologies to improve other FPGAs. Xilinx was widely recognized as a pioneer and market leader in FPGAs, and a POSITA reading the FPGA teachings of Seshadri and Biederman would have naturally looked to Xilinx patents for additional implementation details and features.

307. A POSITA would have had a reasonable expectation of success in applying Kulkarni's teachings to Seshadri's and Biederman's teachings. First, the combinations would not have changed the principle of operation of any reference. As explained above, Kulkarni's teachings could have been applied to Seshadri by adding a configurable communication manager, and Seshadri teaches using such hard macros for communication ports. Ex-1004, 10:48-65 ("Hard macros for other types of communication ports"). Therefore, the combination simply uses each reference for its intended purpose.

308. A POSITA would have found the combinations obvious and would have had a reasonable expectation of success in doing so, because Kulkarni provides the details of how to implement its configurable memory manager on an FPGA. *E.g.*, Ex-1008, 6:28-7:41, Fig. 2, Fig. 3:



As I explained above, the combinations do not require extensive modification or a change of principle. A POSITA would have understood Kulkarni's teachings to be additions that could be implemented with routine engineering, and a POSITA would have been fully capable of combining the teachings of the different references in the manner I explained above. *Supra* §III.

309. **Motivation to Combine with Atsatt-113 Grounds (3-4).** As I explained above, Atsatt-113 and Biederman teach known techniques and uses of PLDs, including FPGAs (*supra* §V.B), to which Kulkarni's teachings could be readily applied. A POSITA would have been motivated to further apply Kulkarni's teachings because Kulkarni suggests using its configurable memory manager in PLDs, including FPGAs. Ex-1008, 1:11-54. In particular, Kulkarni notes that "[i]t may be time consuming and expensive to design [custom] protocols

for exchanging data between the processors of a multiprocessing system” implemented on a PLD. Ex-1008, 1:48-54.

310. A POSITA would have been motivated to use Kulkarni’s teachings of a configurable memory manager and ports in other PLDs, such as those taught by Atsatt-113 and Biederman, because Kulkarni provides a powerful and flexible solution for organizing communication between different parts of a PLD. Kulkarni’s configurable memory manager is particularly well-suited for FPGAs hosting a variable number of user partitions that can be configured with circuitry of varying specifications because it allows communication bandwidth to be dynamically tailored to the capacity and needs of each user’s custom logic. Ex-1008, 2:25-37, 4:26-42, 6:22-27. Therefore, a POSITA would have been motivated to apply Kulkarni’s teachings to adjust the data width of ports in each sandbox taught by Atsatt-113 and Biederman based on the capacity and needs of the user’s custom logic. *See id.*

311. Moreover, Atsatt-113 and Kulkarni both teach an FPGA with an area that is client-configurable. Therefore, a POSITA would have been motivated to apply Kulkarni’s teachings regarding “client” areas in the FPGA to Atsatt-113’s teachings regarding programmable logic sectors because it would have used Kulkarni’s configurable memory manager to improve similar devices in the same way. Ex-1008, 2:25-37, 4:26-42, 6:22-27. Additionally, Atsatt provides

motivation to “include transceiver circuitry.” Ex-1006, ¶30, ¶104 (“[T]he disclosure is not intended to be limited to the particular forms disclosed.”).

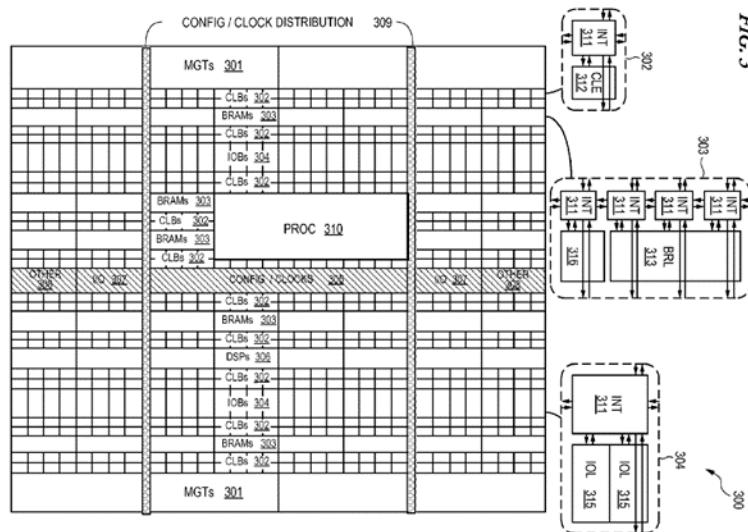
Kulkarni teaches an optimization for I/O circuitry, which a POSITA would have been motivated to combine with Atsatt-113. Ex-1008, 1:56-3:5, 2:58-3:12.

312. Additionally, Kulkarni is a patent filed by Xilinx, which was the first company to introduce commercial FPGAs and has long commanded around half of the global market for FPGAs. A POSITA would have looked to publications embodying Xilinx’s research efforts, such as patents like Kulkarni, for advanced technologies to improve other FPGAs. Xilinx was widely recognized as a pioneer and market leader in FPGAs, and a POSITA reading the FPGA teachings of Seshadri and Biederman would have naturally looked to Xilinx patents for additional implementation details and features.

313. A POSITA would have had a reasonable expectation of success in applying Kulkarni’s teachings to Atsatt-113 alone and in combination with Biederman. First, the combinations would not have changed the principles of operation of any of the references. Kulkarni’s teachings are meant to be an addition to well-known configurations of interconnect resources in FPGAs, rather than replacing or modifying them. Indeed, the use of additional circuitry to carry out specific functions was well-known, and Atsatt-113 is designed to support such additional circuitry. Ex-1006, ¶30 (“[T]he FPGA 40 may include transceiver

circuitry....”); *supra* §I.B. Each reference’s teachings would have been used the way they are taught in each reference, to provide the functionality and benefits as taught by each reference.

314. A POSITA would have found the combinations obvious and would have had a reasonable expectation of success in doing so, because Kulkarni provides the details of how to implement its configurable memory manager on an FPGA. *E.g.*, Ex-1008, 6:28-7:41, Fig. 2, Fig. 3:



As I explained above, the combinations do not require extensive modification or a change of principle. A POSITA would have understood Kulkarni’s teachings to be additions that could be implemented with routine engineering, and a POSITA with 2 years’ experience (*supra* §III) would have been fully capable of combining the teachings of the different references in the manner I explained above.

E. Ground 7

315. Atsatt-923 discloses monitoring temperatures to adjust the operation frequency of different portions of an FPGA and teaches the limitations of claims 8 and 17. Ground 7 adds Atsatt-923's teachings to each combination of Grounds 1-4. The motivation to combine is explained in detail below.

1. U.S. Patent Application Publication No. 2019/0041923A1 ("Atsatt-923") (Ex-1009)

316. I have reviewed the Atsatt-923 reference. I understand that Atsatt-923 was not cited or considered during prosecution of the '943 Patent, based primarily on the fact that Atsatt-923 is not cited on the face of the patent, nor have I seen the reference discussed in the prosecution history.

317. Atsatt-923 is entitled "Systems and methods for integrating power and thermal management in an integrated circuit." Ex-1009, (54) Title. Atsatt-923 was filed on December 27, 2017 and published on February 7, 2019. Based on the priority date of July 15, 2018 of the '943 patent, I understand Atsatt-923 is prior art to the '943 patent under post-AIA §§102(a)(2).

318. Atsatt-923 discloses thermal and power management on PLDs such as FPGAs. Ex-1009, ¶1. Atsatt-923 presents IC configurations that use voltage regulators to "provide power, voltage frequency changes, action voltage recovery margin operations, dynamic voltage frequency scaling, and other voltage signals." Ex-1009, ¶32, Fig. 1:

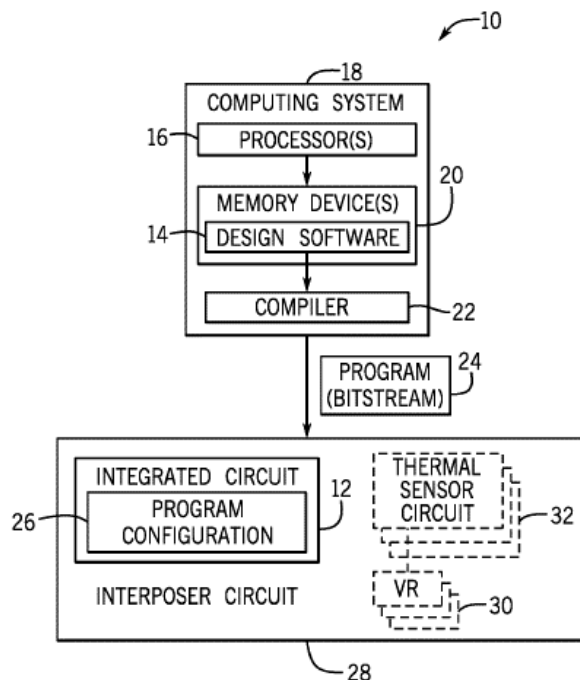


FIG. 1

319. Frequency scaling is carried out on “sector groups” of the FPGA. Ex-1009, ¶18-24. Thermal sensors are used to “measure a temperature or an amount of heat generated from a sector or a portion of the integrated circuit” upon which a controller send “commands to the voltage regulator 30 to cease operations, increase voltage frequency, decrease voltage frequency, and the like.” Ex-1009, ¶33.

320. Atsatt-923 is analogous art to the ’943 patent because it is in the same field of endeavor as the ’943 patent, which is PLDs and data processing systems. *Supra* §III. Atsatt-923 presents techniques for use with “programmable logic devices [that] are increasingly used as accelerators for machine learning, video processing, voice recognition, image recognition, and many other highly

specialized tasks, particularly those that would be too slow or inefficient in software running on a processor.” Ex-1009, ¶4. Atsatt-923 is reasonably pertinent to the problem addressed by the ’943 patent’s inventor, because it solves technical challenges that arise when large numbers of FPGAs are concentrated in data centers.

2. Dependent Claims 8 and 17 are Rendered Obvious in View of Atsatt-923 in combination with Grounds 1-4.

- 8. The programmable device of claim 1, wherein the one or more sandboxes each comprise clock cycle utilization regulation for controlling power consumption or thermal characteristics.**
- 17. The programmable device of claim 10, wherein each of the plurality of one or more sandboxes comprises clock cycle utilization regulation for controlling power consumption or thermal characteristics.**

321. Claims 8 and 17 each respectively depend from claims 1 and 10, but otherwise recite substantively similar language. *Infra* Appx-3 (claim appendix showing claims side-by-side).

322. Atsatt-923 teaches claim 8. The overlapping limitations of claim 17 are satisfied for the same reasons. Atsatt-923 further teaches the additional limitation of claim 17 regarding “each of the plurality” of sandboxes. I explained above how Grounds 1-4 teach the plurality of sandboxes for claim 10. *Supra* §V.A.4 (incorporated by reference), §V.B.3 (incorporated by reference).

323. As I explained above for claims 1 and 10, Grounds 1-4 each teaches a PLD with a first region with a router, a second region with sandboxes, and a port

with a bridge for locking/unlocking user access to the sandboxes. *Supra* §§V.A.3-4, §§V.B.2-3.

324. Atsatt-923 teaches **the one or more sandboxes each comprise clock cycle utilization regulation for controlling power consumption or thermal characteristics**. For example, Atsatt-923 regulates the power consumption and operating temperature for groups of logic sectors in an FPGA by dynamically adjusting the voltage and frequency. Ex-1009, ¶¶21-23, ¶26 (“As a result, the integrated circuit may have improved thermal management by ensuring that different sectors or sector groups operate within a desired temperature range.”). Supporting circuits, including voltage regulators and thermal sensors, are placed below the programmable fabric. Ex-1009, ¶¶46-48, Figs. 4-9. Atsatt-923 regulates clock cycle utilization by adjusting clock cycle frequency;²⁷ this controls power consumption because frequency scaling allows FPGA sectors to “use more power (e.g., operate at higher voltage or *frequency*) [or] operate with less power.” Ex-1009, ¶62. Clock frequency is a direct function of the clock cycle (it is the number of clock cycles per second). For a given workload, scaling clock frequency up decreases clock cycle utilization, while scaling clock frequency down increases

²⁷ In district court, Patent Owner alleged that “scaling down processing speed/frequency” satisfies this limitation. Ex-1021, 000058; *supra* §II.D.

clock cycle utilization. For example, if a workload utilizes 1,000,000 clock cycles per second, and the clock frequency is 2 MHz (2,000,000 cycles per second), the clock cycle utilization is 50%. For the same workload, the clock utilization becomes 33% at a 3 MHz clock speed but increases to 100% at a 1 MHz clock speed.

325. Atsatt-923 teaches that each sandbox includes this functionality for regulating clock cycle utilization. For example, Atsatt-923 teaches applying the frequency scaling to groups of logic sectors inside an FPGA (Ex-1009, ¶¶21-23, ¶26, ¶¶63-64), which aligns with the sandboxed partition teachings of Atsatt-113 and Seshadri. Based on the combined teachings of the reference, it would have been obvious to include Atsatt-923's clock regulation teachings for each partition in the FPGA because one of Atsatt-923's stated goals is to provide this feature for distinct sectors or portions of an FPGA. *Id.* Moreover, it would have been obvious to control the voltage and frequency on a partition-basis because each partition hosts a custom user-defined logic circuit, where activity and utilization will typically differ between partitions. Therefore, clock cycle utilization would need to be adjusted on a per-partition basis to meet power consumption or thermal constraints for the distinct workloads and custom logic in each partition.

326. **Motivation to Combine with Seshadri Grounds (1-2).** As I explained above for Grounds 1-2, Seshadri and Biederman teach known techniques

and uses of PLDs. *Supra* §V.A. A POSITA would have been motivated to further apply Atsatt-923's teachings because Atsatt-923 suggests improving power management of "programmable logic device[s], such as a field programmable gate array (FPGA)," which are taught by Seshadri and Biederman. Ex-1009, ¶1. Thus, applying Atsatt-923's teachings to Seshadri alone and in combination with Biederman would have been a combination of prior art elements according to known methods to yield the predictable results of monitoring operational temperatures and adjusting the clock cycle utilization (e.g., frequency) of user partitions in PLDs. Moreover, Seshadri suggests "various modifications and changes" with an "intention to cover all modifications" to its teachings (Ex-1004, 29:62-30:11).

327. A POSITA would have been further motivated because Seshadri and Biederman both contemplate the use of PLDs in high density data center servers. The need to manage temperature levels to keep ICs within operating parameters, or alternatively to adjust performance to maintain operability at a given temperature was well-known. In fact, management of Thermal Design Power (TDP) budget had been a primary consideration for computer systems since the mid-2000s and one of the main constraints on performance. *E.g.*, Ex-1034, ¶¶3-4. A POSITA would have realized that Atsatt-923's teachings provide a solution to this problem, conveniently embedded in the PLD, for providing more efficient utilization and

reduce wasted power and heat. Atsatt-923's teachings were advantageous in terms of saving time developing external circuitry or solutions to include on the cards or in the server, and would have provided a fail-safe mechanism if such external solutions failed. Moreover, at the scale used in cloud computing servers, adjusting internal clock cycle utilization could also provide meaningful reductions in energy use. *See* Ex-1009, ¶5.

328. A POSITA would have had a reasonable expectation of success in applying Atsatt-923's teachings to Seshadri alone and in combination with Biederman. The combinations would not have changed the principle of operation of any reference, as Atsatt-923's teachings are simply used to adjust the voltage and speed of an IC without otherwise changing the ways in which it operates. As Atsatt-923 explains, implementing its disclosures would "be a routine undertaking of design, fabrication, and manufacture for those of ordinary skill having the benefit of this disclosure." Ex-1009, ¶16. And the management of power, voltage, and frequency was well-known in data centers and for FPGAs. Seshadri's and Biederman's teachings would be used in the way they are taught as I explained above for Grounds 1-4, with minor additions to monitor temperatures and control the operating frequency and voltage of different logic sectors.

329. A POSITA would have found the combinations obvious and would have had a reasonable expectation of success in doing so. As I explained above,

the combinations do not require extensive modification or a change of principle. A POSITA would have understood Atsatt-923's teachings to be additions that could be implemented with routine engineering, and a POSITA would have been fully capable of combining the teachings of the different references in the manner I explained above. *See supra* §III.

330. **Motivation to Combine with Atsatt-113 Grounds (3-4).** As I explained above for Grounds 3-4, Atsatt-113 and Biederman teach known techniques and uses of PLDs, including FPGAs. *Supra* §V.B. A POSITA would have been motivated to further apply Atsatt-923's teachings because Atsatt-923 suggests improving power management of "programmable logic device[s], such as a field programmable gate array[s] (FPGA)." Ex-1009, ¶1. Applying Atsatt-923's teachings to Atsatt-113 alone and in combination with Biederman would have been a combination of prior art elements according to known methods to yield the predictable results of monitoring operational temperatures and adjusting the clock cycle utilization (e.g., frequency) of user partitions in PLDs.

331. A POSITA would also have been motivated to combine because Atsatt-113 and Atsatt-923 both come from Intel. Both are patent publications that identify Intel as the applicant, as well as common inventors Atsatt and Weber. Ex-1006, (71)-(72); Ex-1009, (71)-(72). Atsatt-923's teachings were meant to be used in similar FPGA configurations as those taught by Atsatt-113. *Compare* Ex-1006,

Fig. 2, *with* Ex-1009, Fig. 3. Biederman does not change this analysis because its teachings simply apply known use cases, such as using FPGAs for data flow processing in distributed computing systems, and dynamically adjusting partition sizes.

332. A POSITA would have been further motivated by the advantages of Atsatt-923 that I explained above for the combination of Atsatt-923 with Grounds 1-2. The same reasons apply here and are incorporated by reference.

333. A POSITA would have had a reasonable expectation of success in applying Atsatt-923's teachings to Atsatt-113 alone and in combination with Biederman. The combinations would not have changed the principle of operation of any reference, as Atsatt-923's teachings simply adjust the voltage and speed at which ICs operate without otherwise changing the ways in which they operate. As Atsatt-923 explains, implementing its disclosures would "be a routine undertaking of design, fabrication, and manufacture for those of ordinary skill having the benefit of this disclosure." Ex-1009, ¶16. Atsatt-113's, and Biederman's teachings would be used in the way they are taught as I explained above for Grounds 3-4, with minor additions to monitor temperatures and control the operating frequency and voltage of different logic sectors.

334. A POSITA would have found the combinations obvious and would have had a reasonable expectation of success in doing so. As I explained above,

the combinations do not require extensive modification or a change of principle. A POSITA would have understood Atsatt-923's teachings to be additions that could be implemented with routine engineering, and a POSITA with 2 years' experience (*supra* §III) would have been fully capable of combining the teachings of the different references in the manner I explained above.

VI. BACKGROUND AND QUALIFICATIONS

335. This section contains a summary of my educational background, career history, publications, and other relevant qualifications. My curriculum vitae, including my publications and patents, is attached as Appendix 1 to this declaration.

I am an independent consultant with Camelback Computer Architecture, LLC. My residence and place of business is at 7 Farm Road, Los Altos CA, 94024. I am over the age of eighteen, and I am a citizen of the United States.

I have over 45 years of academic and industrial experience in applying, designing, studying, teaching, and writing about microprocessors and computer systems. I received an Electrical Engineering Ph.D. degree in 1984 from Stanford University. I earlier received an Electrical Engineering B.S. degree from MIT in 1973 and an Electrical Engineering M.S. degree from Stanford University in 1978. I have taught classes in computer architecture at Stanford, Tel Aviv, and Arizona State Universities.

From 1976 to 1977, I worked at Burroughs Corporation, where I designed peripheral interface controllers, including those for serial data communications based on Intel 8080 microprocessor components. From 1980 to 1989, I was the lead architect for the design of three high-performance microprocessors at Zilog and National Semiconductor. Later, at Intel, I was the lead architect of the Pentium® Processor from 1989 to 1992 and of the 815 chipset from 1999 to 2000, both of which became the most widely used PC components of their time. The 815 chipset comprised two components: (1) a memory controller hub (MCH) that included a graphics controller and memory controller with interfaces to the CPU, 133 MHz SDRAM system memory modules, an optional, external graphics controller and (2) an I/O controller hub (ICH) that included various I/O controllers (e.g., network, hard drive, USB) for system peripheral devices. Additionally, I served as co-manager for the Itanium processor design from 1993-1997.

I am a Senior Member of the Institute of Electrical and Electronics Engineers (IEEE), and served as the chair of the IEEE Technical Committee on Microprocessors and Microcomputers from 1999 to 2000. I was the keynote speaker at the first Cool Chips conference, dedicated to the study of low-power microprocessors and systems. I have given invited lectures at several universities, and published ten papers in various professional journals and conference proceedings. My paper entitled “Architecture of the Pentium Processor,” was

selected as best paper in IEEE Micro for 1993. I am a named inventor on over 30 U.S. patents that pertain to microprocessors, computer systems, and related technology.

336. I have reviewed the '943 Patent, and I am familiar with the patent's subject matter, which is within the scope of my education and professional experience. For example, during development of the Pentium Processor I am familiar with design verification using a large emulation system of FPGAs. Based at least on my background in academia and industry, I am familiar with the issues and technology relating to processors, chipsets, memory, and peripheral devices for computer systems, including those for scalable, multiprocessor systems. I have personally analyzed, developed, and tested such computer components and systems.

A. Compensation

337. For my efforts in connection with the preparation of this declaration I have been compensated at my standard rate of \$695 per hour for this type of consulting activity. My compensation is in no way contingent on the results of these or any other proceedings relating to the above-captioned patent.

B. Materials and Other Information Considered

338. I have considered information from various sources in forming my opinions. I have reviewed and considered each of the exhibits listed in the attached

Appendix 2 (Materials Considered in the Preparation of This Declaration) in forming my opinions.

VII. UNDERSTANDING OF THE LAW

339. I am not an attorney. In forming my opinions in this Declaration, I applied the relevant legal principles provided to me by counsel, which are summarized in Appendix 4.

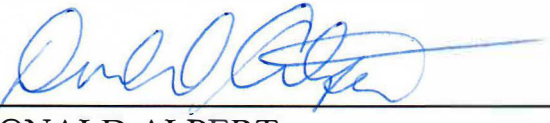
VIII. RESERVATION OF RIGHTS

340. My opinions are based upon the information that I have considered to date. I am unaware of any evidence of secondary considerations with respect to the '943 Patent that would render any of the challenged claims non-obvious. I reserve the right, however, to supplement my opinions in the future to respond to any arguments raised by the owner of the '943 Patent and to take into account new information that becomes available to me.

341. I declare that all statements made herein of my knowledge are true, and that all statements made on information and belief are believed to be true, and that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Executed on 12/18, 2024.

By:



DONALD ALPERT

APPENDIX 1: CURRICULUM VITAE OF DONALD ALPERT



Donald Alpert, Ph.D.
Camelback Computer Architecture, LLC
2310 Homestead Road
Suite C1 #411
Los Altos CA 94024
(650) 386-5301

Donald Alpert, Ph.D., is an independent consultant, specializing in microprocessors and computer systems. For more than a decade in his role as president of Camelback Computer Architecture LLC, Dr. Alpert has provided clients with expert services for patent litigation, technology assessment, and product development. He has over 40 years of wide-ranging academic and industry experience that enable him to provide expert consulting services in the following areas:

- Computers, Computer Systems
- Computer architecture
- Microprocessors
- Semiconductors
- PC Systems
- Embedded Systems
- Design Validation and Debug
- Performance Evaluation
- Cache Memory
- Instruction Sets
- Media Processing
- Power Management

Patent Analysis: Dr. Alpert has served as an expert on dozens of patent litigation cases, handling both infringement and invalidity issues. Dr. Alpert has worked as both a consulting expert and testifying expert assisting attorneys representing major high-tech corporations. He has substantial experience testifying in depositions and at trials before U.S. Federal Courts and the International Trade Commission. Dr. Alpert also consults to assess patents for acquisition and for pre-litigation review. See [here](#) for a summary of testifying experience.

Product Development: Dr. Alpert is a widely recognized innovator with over 30 patents. Dr. Alpert has worked at some of the most ground-breaking technology firms, including Zilog, National Semiconductor, and Intel, where he was the principal architect of the Pentium® Processor. See [here](#) for more information and example projects.

Employment Experience

Camelback Computer Architecture, LLC, Sacramento, CA

Consultant

2/02-Present

Computer architecture consulting services to assess technology, develop products, and protect IP.

Intel Corporation, Chandler, Arizona

Principal Engineer

12/99-1/02

Led architecture team of 10 responsible for definition and performance evaluation of Itanium™ Processor Family microprocessors.

Principal Engineer

5/98-12/99

Led architecture team of 4 responsible for technical strategy and product definition of PC graphics processors and chipsets.

Stanford University, Stanford, California

Consulting Assistant Professor

9/97-4/98

Responsible for teaching undergraduate and graduate classes in computer architecture and participating in processor-design research projects.

Intel Corporation, Santa Clara, California

Engineering Manager

10/93-6/97

Responsible for microarchitecture definition and architecture validation of the Itanium™ microprocessor, which was the first implementation of Intel's 64-bit extension architecture. Led a team of 60 engineers.

Architecture Manager

1/89-10/93

Responsible for the Pentium® Processor technical definition. Worked with customers and Intel marketing to identify product requirements and with the design engineering team to evaluate design options. Led a team of seven engineers responsible for creating technical specifications and for modeling and simulating performance using traces from DOS, Windows, and UNIX execution. After completing the Pentium Processor, led an interdisciplinary team of 30 engineers to define a new 64-bit instruction set architecture.

National Semiconductor Corporation, Herzlia, Israel

Architecture Manager

5/85-1/89

Responsible for defining new microprocessors in the Series 32000 family, including the NS32532 and "Swordfish" high-performance microprocessors. Led a team of five engineers working in the following areas: modeling and simulating high-performance microprocessors, evaluating novel computer architectures, and writing technical specifications for new products.

Zilog, Incorporated, Campbell, California

Component Architect

6/80-4/85

Worked on the Z80,000 CPU, a 32-bit microprocessor that integrated an instruction pipeline, cache, and memory management unit. Responsible for specifying the non-privileged instruction set, evaluating performance, and writing the reference manual for the microprocessor.

Burroughs Corporation, Pasadena, California

Associate Engineer

7/76-7/77

Designed computer peripheral interface controllers using MSI-TTL and 8080 microprocessors.

Alpx Computer Corporation, Danbury, Connecticut

8/75-6/76

Consulting engineer working on design of video games.

Action, Washington, DC

6/73-6/75

Peace Corps volunteer teaching mathematics and physics in a secondary school in Ghana, West Africa.

Donald Alpert Camelback Computer Architecture, LLC (650) 386-5301

Education

Stanford University 9/77-6/84
Graduate study in computer design. National Science Foundation Graduate Fellowship.
Awarded MSEE in 1978 and PhDEE in 1984.

M.I.T. 9/69-6/73
Member of Tau Beta Pi and Eta Kappa Nu. Awarded BSEE in 1973.

Professional Activities

Member of Microprocessor Report editorial board (2002-present)

Anthem College Commencement Address (18-Dec-2009)

Panel session about history of microprocessor development at National Semiconductor Corporation, Computer History Museum (2008).

[\[http://archive.computerhistory.org/resources/access/text/Oral_History/102658246.05.01.acc.pdf\]](http://archive.computerhistory.org/resources/access/text/Oral_History/102658246.05.01.acc.pdf)

IEEE Technical Committee on Microprocessors and Microcomputers, Chair (1999-2000) and Vice-Chair (2001-present).

Lectures about the Itanium™ Processor Family at U.C. Irvine (1999, 2000).

Cool Chips I, Keynote speaker (Tokyo, 1998)

IEEE Hot Chips Symposium Program Co-chair (1994), member of program committee (1993, 1995).

Lectures about the Pentium® Processor at Stanford University, University of Illinois, and Cornell (1993-1994).

Appeared in Stanford UVC Educational Video production about the Pentium Processor (1993).

IEEE International Symposium on Computer Architecture member of program committee (1996) and Session Chair for Cache Design (1989).

Teaching Experience

Faculty Associate at Arizona State University teaching undergraduate class on assembly language programming (Spring/2001) and computer architecture (Spring/2002).

Consulting Associate Professor at Stanford University teaching undergraduate and graduate classes in computer architecture (Fall/1997, Winter/1998).

Consulting Professor at Tel Aviv University teaching graduate class in computer architecture (Spring/1987)

Peace Corps volunteer in a secondary school in Ghana, West Africa teaching mathematics and physics (1973-1975).

Don Alpert, "How Microprocessors Upset the Computer Industry," Microprocessor Report, December 11, 2006.

Don Alpert, "Evolving PC Chip-Set Topologies," Microprocessor Report, April 17, 2006.

Don Alpert, "Will Microprocessors Become Simpler?" Microprocessor Report, November 17, 2003.

Don Alpert, "Itanium Processor Status Report," Microprocessor Report, July 28, 2003.

Don Alpert, "Scalable MicroSupercomputers," Microprocessor Report, March 17, 2003.

Don Alpert, "ColdFire Goes Fully Superscalar," Microprocessor Report, October 28, 2002.

Donald Alpert and Alan Jay Smith, "Guest Editor Introduction," IEEE Micro special issue on Hot Chips VI, April 1995.

Donald Alpert and Dror Avnon, "Architecture of the Pentium Processor," IEEE Micro, June 1993, pp. 11-21. Selected best paper in IEEE Micro for 1993.

D. Alpert, A. Averbuch, and O. Danielli, "Performance Comparison of Load/Store and Symmetric Instruction Set Architectures," Proceedings, 18th International Symposium on Computer Architecture, June 1990, pp. 172-181.

Benjamin Maytal, Sorin Iacobovici, Donald Alpert, et al, "Design Considerations for a General-Purpose Microprocessor," Computer, January 1989, pp. 66-76.

Donald Alpert and Michael Flynn, "Performance Trade-offs for Microprocessor Cache Memories," IEEE Micro, August 1988, pp. 44-54.

D. Alpert, J. Levy, and B. Maytal, "Architecture of the NS32532 Microprocessor," Proceedings, 1987 IEEE International Conference on Computer Design, October 1987, pp. 168-172. Awarded best presenter at Design and Test sessions.

D. Alpert, D. Biran, L. Epstein, et al, "Trends in VLSI Microprocessor Design," Proceedings, First Annual Conference on Computer Technology, Systems and Applications (CompEuro '87), May 1987, pp. 564-567.

Y Sidi, D. Alpert, D. Biran, et al, "Design Considerations of an Advanced 32-Bit Microprocessor," Proceedings, 15th Conference of Electrical and Electronic Engineers in Israel, April 1987.

D. Alpert, "Trends in VLSI Microprocessor Design," Proceedings, 15th Conference of Electrical and Electronic Engineers in Israel, April 1987.

Donald Alpert, Michael J. Flynn, and Scott Wakefield, "Directly Executed Languages for VLSI Processor Design," Proceedings, 1983 IEEE International Conference on Computer Design, October 1983, pp. 609-612.

Donald Alpert, "Powerful 32-Bit Micro Includes Memory Management," Computer Design, October 1983, pp. 213-220.

Don Alpert, Dean Carberry, Mike Yamamura, et al, "32-Bit Processor Chip Integrates Major System Functions," Electronics, July 14, 1983, pp. 113-119.

Patents

4,802,085, Apparatus and method for detecting and handling memory-mapped I/O by a pipelined microprocessor

5,249,286, Selectively locking memory locations within a microprocessor's on-chip cache

5,263,153, Monitoring control flow in a microprocessor

5,416,913, Method and apparatus for dependency checking in a multi-pipelined microprocessor

5,442,756, Branch prediction and resolution apparatus for a superscalar computer processor

5,475,824, Microprocessor with apparatus for parallel execution of instructions

5,479,652, Microprocessor with an external command mode for diagnosis and debugging

5,481,751, Apparatus and method for storing partially-decoded instructions in the instruction cache of a CPU having multiple execution units

5,559,986, Interleaved cache for multiple accesses per clock cycle in a microprocessor

5,606,676, Branch prediction and resolution apparatus for a superscalar computer processor

5,617,554, Physical address size selection and page size selection in an address translator

5,621,886, Method and apparatus for providing efficient software debugging

5,638,525, Processor capable of executing programs that contain RISC and CISC instructions

5,657,253, Apparatus for monitoring the performance of a microprocessor

5,659,679, Method and apparatus for providing breakpoints on taken jumps and for providing software profiling in a computer system

5,669,011, Partially decoded instruction cache

5,671,435, Technique for software to identify features implemented in a processor

5,675,825, Apparatus and method for identifying a computer microprocessor

5,692,167, Method for verifying the correct processing of pipelined instructions including branch instructions and self-modifying code in a microprocessor

5,729,724, Adaptive 128-bit floating point load and store operations for quadruple precision compatibility

5,740,413, Method and apparatus for providing address breakpoints, branch breakpoints, and single stepping

5,764,959, Adaptive 128-bit floating point load and store instructions for quad-precision compatibility

5,774,686, Method and apparatus for providing two system architectures in a processor

5,790,834, Apparatus and method using an ID instruction to identify a computer microprocessor

5,802,605, Physical address selection and page size selection in an address translator

5,958,037, Apparatus and method for identifying features and the origin of a computer microprocessor

5,991,874, Conditional move using a compare instruction generating a condition field

6,052,801, Method and apparatus for providing breakpoints on a selectable address range

6,219,774, Address translation with/bypassing intermediate segmentation translation to accommodate two different instruction set architecture

6,385,718, Computer system and method for executing interrupt instructions in operating modes

6,408,386, Method and apparatus for providing event handling functionality in a computer system

6,584,558, Article for providing event handling functionality in a processor supporting different instruction sets

7,010,671, Computer system and method for executing interrupt instructions in two operating modes

7,389,403, Adaptive computing ensemble microprocessor architecture

7,802,073, Virtual core management

APPENDIX 2: MATERIALS CONSIDERED IN THE PREPARATION OF THIS DECLARATION

Exhibit No.	Description
1001	U.S. Patent No. 10,985,943 (“’943 patent”)
1002	Prosecution History of the ’943 patent
1004	U.S. Patent No. 10,776,142 (“Seshadri”)
1005	U.S. Patent Application Publication No. 2018/0191642A1 (“Biederman”)
1006	U.S. Patent Application Publication No. 2019/0095113A1 (“Atsatt-113”)
1007	Tarafdar et al., “Enabling Flexible Network FPGA Clusters in a Heterogeneous Cloud Data Center” (“Tarafdar”)
1008	U.S. Patent No. 8,015,386 (“Kulkarni”)
1009	U.S. Patent Application Publication No. 2019/0041923A1 (“Atsatt-923”)
1010	Kuon et al., “FPGA Architecture: Survey and Challenges” (“Kuon”)
1011	Trimberger, “Three Ages of FPGAs: A Retrospective on the First Thirty Years of FPGA Technology” (“Trimberger”)
1012	Teubner & Woods, “Data Processing on FPGAs” (“Teubner”)
1013	Kean, “Secure Configuration of Field Programmable Gate Arrays” (“Kean”)
1014	Frey, “The IBM Data Center Family” (“Frey”)
1015	Desai et al., “BladeCenter system overview” (“Desai”)
1016	Hughes et al., “BladeCenter processor blades, I/O expansion adapters, and units” (“Hughes”)
1017	Caldeira et al., “IBM Power Systems S814 and S824 Technical Overview and Introduction” (Caldeira)
1018	Steffen, “Nallatech Software Tools on the Maxwell FPGA Parallel Computer” (“Steffen”)
1019	Nallatech, “H100 Series FPGA Application Accelerators” (“Nallatech”)
1020	Bassiri & Shahhoseini, “Configuration Reusing in On-Line Task Scheduling for Reconfigurable Computing Systems” (“Bassiri”)
1021	Exhibit D to Concurrent Ventures’ and XstreamEdge’s Oct. 11, 2024 Preliminary Invalidity Contentions
1022	Declaration of S. Delman re Tarafdar

Exhibit No.	Description
1023	Chen et al., “Enabling FPGAs in the Cloud” (“Chen”)
1024	Asiatici et al., “Virtualized Execution Runtime for FPGA Accelerators in the Cloud” (“Asiatici”)
1025	Vipin & Fahmy, “FPGA Dynamic and Partial Reconfiguration: A Survey of Architectures, Methods, and Applications” (“Vipin”)
1026	Fahmy et al., “Virtualized FPGA Accelerators for Efficient Cloud Computing” (“Fahmy”)
1027	RocketBoards, “PCIe Root Port with MSI”
1028	Aldec, “High Performance Computing – Solutions”
1029	JEDEC Dictionary of Terms for Solid-State Technology, 7th Ed.
1030	IEEE 1532-2002 (“Standard for In-System Configuration of Programmable Devices”)
1031	U.S. Patent Application Publication No. 2016/0049941A1 (“How”)
1032	Cisco, “High Speed Serial Interface (HSSI) Design Specification”
1033	PCI-SIG, “PCI Express Base Specification Revision 1.0”
1034	U.S. Patent Application Publication No. Ex-1034 2011/0090640A1 (“Samson”)

APPENDIX 3: CHALLENGED CLAIMS

1[pre]. A programmable device for data flow processing in a user-configurable server with swappable pods or cards, comprising:

[1a] a programmable logic device (PLD) having a first region and a second region;

[1b] the first region comprising a hardware-based or firmware-based router with a port enabling communication between the router and the second region; and

[1c] the second region comprising one or more sandboxes with user-definable programmable electronic circuits of the PLD,

[1d] wherein the port comprises: a bridge being lockable to prevent user access, and unlockable to enable user access, to each of the sandboxes and the user-definable programmable electronic circuits therein.

2. The programmable device of claim 1, further comprising:

the router communicative to translate between a first protocol and a second protocol, for messaging with system resources.

3. The programmable device of claim 1, wherein each of the sandboxes comprises user-programmable one or more of: analog circuitry, digital circuitry, memory, processors or controllers.

4. The programmable device of claim 1, wherein the port comprises:

a bridge being lockable to prevent reconfiguration of the bridge and prevent reconfiguration of the router while enabling reconfiguration of the one or more sandboxes.

5. The programmable device of claim 1, wherein the port comprises:

a bridge having a security layer to authenticate a user or verify a permission, to reconfigure one or more of the one or more sandboxes.

6. The programmable device of claim 1, further comprising:

the router configured to perform analytics on resources of the programmable device, on a per module basis, including determining communication bandwidth and resource utilization.

7. The programmable device of claim 1, wherein the one or more sandboxes each comprise port interfaces that are configurable as to data width or speed.

8. The programmable device of claim 1, wherein the one or more sandboxes each comprise clock cycle utilization regulation for controlling power consumption or thermal characteristics.

9. The programmable device of claim 1, wherein the one or more sandboxes comprises a plurality of sandboxes that are logically or physically combinable to form a larger sandbox.

10[pre]. A programmable device for data flow processing in a user-configurable server, comprising:

[10a] a programmable logic device (PLD) having a first region and a second region;

[10b] the first region comprising a hardware-based or firmware-based router having a bridge with a port enabling communication between the router and the second region;

[10c] the second region comprising a plurality of sandboxes with user-definable programmable electronic circuits of the PLD; and

[10d] an internal communication network in the PLD, coupling the port of the bridge of the router and the one or more sandboxes for the data flow processing,

[10e] wherein the bridge is lockable to prevent user access, and unlockable to enable user access, to the plurality of sandboxes and user-definable programmable electronic circuits therein.

11. The programmable device of claim 10, further comprising:

the router to translate between a first protocol used external to the second region and a second protocol used in the internal communication network in the PLD.

12. The programmable device of claim 10, wherein each of the plurality of sandboxes comprises user-programmable one or more of: analog circuitry, digital circuitry, memory, processors or controllers.

13. The programmable device of claim 10, wherein the bridge is lockable to enable reconfiguration of the plurality of sandboxes, prevent reconfiguration of the bridge and prevent reconfiguration of the router.

14. The programmable device of claim 10, wherein the bridge has a security layer to authenticate a user or verify a permission, to reconfigure one or more of the plurality of sandboxes.

15. The programmable device of claim 10, further comprising:

the router configured to perform analytics on resources of the programmable device, on a per module basis, including determining communication bandwidth and resource utilization in the plurality of sandboxes.

16. The programmable device of claim 10, wherein the each of the plurality of sandboxes comprises one or more port interfaces configurable as to data width or speed.

17. The programmable device of claim 10, wherein each of the plurality of more sandboxes comprises clock cycle utilization regulation for controlling power consumption or thermal characteristics.

18. The programmable device of claim 10, wherein the plurality of sandboxes is logically or physically combinable to form a single, larger sandbox.

Side-by-Side List of Challenged Claims (differences in bolded green):

1[pre]. A programmable device for data flow processing in a user-configurable server with swappable pods or cards , comprising:	10[pre]. A programmable device for data flow processing in a user-configurable server, comprising:
[1a] a programmable logic device (PLD) having a first region and a second region;	[10a] a programmable logic device (PLD) having a first region and a second region;
[1b] the first region comprising a hardware-based or firmware-based router with a port enabling communication between the router and the second region; and	[10b] the first region comprising a hardware-based or firmware-based router having a bridge with a port enabling communication between the router and the second region;
[1c] the second region comprising one or more sandboxes with user-definable programmable electronic circuits of the PLD,	[10c] the second region comprising a plurality of sandboxes with user-definable programmable electronic circuits of the PLD; and
	[10d] an internal communication network in the PLD, coupling the

	port of the bridge of the router and the one or more sandboxes for the data flow processing,
[1d] wherein the port comprises: a bridge being lockable to prevent user access, and unlockable to enable user access, to each of the sandboxes and the user-definable programmable electronic circuits therein.	[10e] wherein the bridge is lockable to prevent user access, and unlockable to enable user access, to the plurality of sandboxes and user-definable programmable electronic circuits therein.

2. The programmable device of claim 1 , further comprising: the router communicative to translate between a first protocol and a second protocol, for messaging with system resources.	11. The programmable device of claim 10 , further comprising: the router to translate between a first protocol used external to the second region and a second protocol used in the internal communication network in the PLD.
3. The programmable device of claim 1 , wherein each of the sandboxes comprises user-programmable one or more of: analog circuitry, digital circuitry, memory, processors or controllers.	12. The programmable device of claim 10 , wherein each of the plurality of sandboxes comprises user-programmable one or more of: analog circuitry, digital circuitry, memory, processors or controllers.
4. The programmable device of claim 1 , wherein the port comprises: a bridge being lockable to prevent reconfiguration of the bridge and	13. The programmable device of claim 10 , wherein the bridge is lockable to enable reconfiguration of the plurality of sandboxes, prevent reconfiguration

prevent reconfiguration of the router while enabling reconfiguration of the one or more sandboxes.	of the bridge and prevent reconfiguration of the router.
5. The programmable device of claim 1 , wherein the port comprises: a bridge having a security layer to authenticate a user or verify a permission, to reconfigure one or more of the one or more sandboxes.	14. The programmable device of claim 10 , wherein the bridge has a security layer to authenticate a user or verify a permission, to reconfigure one or more of the plurality of sandboxes.
6. The programmable device of claim 1 , further comprising: the router configured to perform analytics on resources of the programmable device, on a per module basis, including determining communication bandwidth and resource utilization.	15. The programmable device of claim 10 , further comprising: the router configured to perform analytics on resources of the programmable device, on a per module basis, including determining communication bandwidth and resource utilization in the plurality of sandboxes .
7. The programmable device of claim 1 , wherein the one or more sandboxes each comprise port interfaces that are configurable as to data width or speed.	The programmable device of claim 10 , wherein the each of the plurality of sandboxes comprises one or more port interfaces configurable as to data width or speed.
8. The programmable device of claim 1 , wherein the one or more sandboxes each comprise clock cycle utilization regulation for controlling power	The programmable device of claim 10 , wherein the each of the plurality of sandboxes comprises clock cycle utilization regulation for controlling

consumption or thermal characteristics.	power consumption or thermal characteristics.
9. The programmable device of claim 1, wherein the one or more sandboxes comprises a plurality of sandboxes that are logically or physically combinable to form a larger sandbox.	18. The programmable device of claim 10, wherein the plurality of sandboxes is logically or physically combinable to form a single , larger sandbox.

APPENDIX 4: UNDERSTANDING OF THE LAW

I have applied the following legal principles provided to me by counsel in arriving at the opinions set forth in this report.

Legal Standard for Prior Art

I am not an attorney. I have been informed by attorneys of the relevant legal principles and have applied them to arrive at the opinions set forth in this declaration.

I understand that the petitioner for inter partes review may request the cancelation of one or more claims of a patent based on grounds available under 35 U.S.C. §102 and 35 U.S.C. §103 using prior art that consists of patents and printed publications.

Anticipation and Prior Art

I understand that § 102 specifies when a challenged claim is invalid for lacking novelty over the prior art, and that this concept is also known as “anticipation.” I understand that a prior art reference anticipates a challenged claim, and thus renders it invalid by anticipation, if all elements of the challenged claim are disclosed in the prior art reference. I understand the disclosure in the prior art reference can be either explicit or inherent, meaning it is necessarily present or implied. I understand that the prior art reference does not have to use the same words as the challenged claim, but all of the requirements of the claim

must be disclosed so that a person of ordinary skill in the art could make and use the claimed subject-matter.

In addition, I understand that § 102 also defines what is available for use as a prior art reference to a challenged claim.

Under § 102(a)(1), a challenged claim is anticipated if it was patented, described in a printed publication, or in public use, on sale, or otherwise available to the public before the effective filing date of the challenged claimed invention.

Under § 102(a)(2), a challenged claim is anticipated if it was described in an issued patent, or in an application for patent that was published by the USPTO, in which the patent or application, as the case may be, names another inventor and was effectively filed before the effective filing date of the challenged claimed invention.

I understand that the filing date of a patent is generally the filing date of the application filed in the United States that issued as the patent. However, I understand that a patent may be granted an earlier effective filing date if the patent owner properly claimed priority to an earlier patent application.

I understand that when a challenged claim covers several structures, either generically or as alternatives, the claim is deemed anticipated if any of the structures within the scope of the claim is found in the prior art reference.

I understand that when a challenged claim requires selection of an element from a list of alternatives, the prior art teaches the element if one of the alternatives is taught by the prior art.

Legal Standard for Obviousness

I understand that even if a challenged claim is not anticipated, it is still invalid if the differences between the claimed subject matter and the prior art are such that the claimed subject matter would have been obvious to a person of ordinary skill in the pertinent art at the time the alleged invention.

I understand that obviousness must be determined with respect to the challenged claim as a whole.

I understand that one cannot rely on hindsight in deciding whether a claim is obvious.

I also understand that an obviousness analysis includes the consideration of factors such as (1) the scope and content of the prior art, (2) the differences between the prior art and the challenged claim, (3) the level of ordinary skill in the pertinent art, and (4) “secondary” or “objective” evidence of non-obviousness.

Secondary or objective evidence of non-obviousness includes evidence of: (1) a long felt but unmet need in the prior art that was satisfied by the claimed invention; (2) commercial success or the lack of commercial success of the claimed invention; (3) unexpected results achieved by the claimed invention; (4)

praise of the claimed invention by others skilled in the art; (5) taking of licenses under the patent by others; (6) deliberate copying of the claimed invention; and (7) contemporaneous and independent invention by others. However, I understand that there must be a relationship between any secondary evidence of non-obviousness and the claimed invention.

I understand that a challenged claim can be invalid for obviousness over a combination of prior art references if a reason existed (at the time of the alleged invention) that would have prompted a person of ordinary skill in the art to combine elements of the prior art in the manner required by the challenged claim. I understand that this requirement is also referred to as a “motivation to combine,” “suggestion to combine,” or “reason to combine,” and that there are several rationales that meet this requirement.

I understand that the prior art references themselves may provide a motivation to combine, but other times simple common sense can link two or more prior art references. I further understand that obviousness analysis recognizes that market demand, rather than scientific literature, often drives innovation, and that a motivation to combine references may come from market forces.

I understand obviousness to include, for instance, scenarios where known techniques are simply applied to other devices, systems, or processes to improve them in an expected or known way. I also understand that practical and common-

sense considerations should be applied in a proper obviousness analysis. For instance, familiar items may have obvious uses beyond their primary purposes.

I understand that the combination of familiar elements according to known methods is obvious when it yields predictable results. For instance, obviousness bars patentability of a predictable variation of a technique even if the technique originated in another field of endeavor. This is because design incentives and other market forces can prompt variations of it, and predictable variations are not the product of innovation, but rather ordinary skill and common sense.

I understand that a particular combination may be obvious if it was obvious to try the combination. For example, when there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. This would result in something obvious because the result is the product not of innovation but of ordinary skill and common sense. However, I understand that it may not be obvious to try a combination when it involves unpredictable technologies.

It is further my understanding that a proper obviousness analysis focuses on what was known or obvious to a person of ordinary skill in the art, not just the patentee. Accordingly, I understand that any need or problem known in the field

of endeavor at the time of invention and addressed by the patent can provide a reason for combining the elements in the manner claimed.

It is my understanding that the Manual of Patent Examining Procedure §2143 sets forth the following as exemplary rationales that support a conclusion of obviousness:

- Combining prior art elements according to known methods to yield predictable results;
- Simple substitution of one known element for another to obtain predictable results;
- Use of known technique to improve similar devices (methods, or products) in the same way;
- Applying a known technique to a known device (method, or product) ready for improvement to yield predictable results;
- Choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success;
- Known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces if the variations are predictable to one of ordinary skill in the art;

- Some teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention.

A person of ordinary skill in the art looking to overcome a problem will often use the teachings of multiple publications together like pieces of a puzzle, even though the prior art does not necessarily fit perfectly together. Therefore, I understand that references for obviousness need not fit perfectly together like puzzle pieces. Instead, I understand that obviousness analysis takes into account inferences, creative steps, common sense, and practical logic and applications that a person of ordinary skill in the art would employ under the circumstances.

I understand that a claim can be obvious in light of a single reference, if the elements of the challenged claim that are not explicitly or inherently disclosed in the reference can be supplied by the common sense of one of skill in the art.

I understand that obviousness also bars the patentability of applying known or obvious design choices to the prior art. One cannot patent merely substituting one prior art element for another if the substitution can be made with predictable results. Likewise, combining prior art techniques that are interoperable with respect to one another is generally obvious and not patentable.

In order for a claim to be found invalid based upon a modification or combination of the prior art, there must be reasonable expectation that a person of ordinary skill would have successfully modified or combined the prior art to arrive at the claimed arrangement. This does not mean that it must be certain that a person of ordinary skill would have been successful – the law only requires that the person of ordinary skill in the art would have perceived a reasonable expectation of success in modifying or combining the prior art to arrive at the claimed invention.

In sum, my understanding is that obviousness invalidates claims that merely recite combinations of, or obvious variations of, prior art teachings using understanding and knowledge of one of skill in the art at the time and motivated by the general problem facing the inventor at the time. Under this analysis, the prior art references themselves, or any need or problem known in the field of endeavor at the time of the invention, can provide a reason for combining the elements of or attempting obvious variations on prior art references in the claimed manner.

Legal Standard for Claim Construction

I understand that before any invalidity analysis can be properly performed, the scope and meaning of the challenged claims must be determined by claim construction.

I understand that a patent may include two types of claims, independent claims and dependent claims. I understand that an independent claim stands alone

and includes only the limitations it recites. I understand that a dependent claim depends from an independent claim or another dependent claim. I understand that a dependent claim includes all the limitations that it recites in addition to the limitations recited in the claim (or claims) from which it depends.

In comparing the challenged claims to the prior art, I have carefully considered the patent and its file history in light of the understanding of a person of skill at the time of the alleged invention.

I understand that to determine how a person of ordinary skill would have understood a claim term, one should look to sources available at the time of the alleged invention that show what a person of skill in the art would have understood disputed claim language to mean. It is my understanding that this may include what is called “intrinsic” evidence as well as “extrinsic” evidence.

I understand that, in construing a claim term, one should primarily rely on intrinsic patent evidence, which includes the words of the claims themselves, the remainder of the patent specification, and the prosecution history. I understand that extrinsic evidence, which is evidence external to the patent and the prosecution history, may also be useful in interpreting patent claims when the intrinsic evidence itself is insufficient. I understand that extrinsic evidence may include principles, concepts, terms, and other resources available to those of skill in the art at the time of the invention.

I understand that words or terms should be given their ordinary and accepted meaning unless it appears that the inventors were using them to mean something else or something more specific. I understand that to determine whether a term has special meaning, the claims, the patent specification, and the prosecution history are particularly important, and may show that the inventor gave a term a particular definition or intentionally disclaimed, disavowed, or surrendered claim scope.

I understand that the claims of a patent define the scope of the rights conferred by the patent. I understand that because the claims point out and distinctly claim the subject matter which the inventors regard as their invention, claim construction analysis must begin with and is focused on the claim language itself. I understand that the context of the term within the claim as well as other claims of the patent can inform the meaning of a claim term. For example, because claim terms are normally used consistently throughout the patent, how a term is used in one claim can often inform the meaning of the same term in other claims. Differences among claims or claim terms can also be a useful guide in understanding the meaning of particular claim terms.

I understand that a claim term should be construed not only in the context of the particular claim in which the disputed term appears, but in the context of the entire patent, including the entire specification. I understand that because the

specification is a primary basis for construing the claims, a correct construction must align with the specification.

I understand that the prosecution history of the patent as well as art incorporated by reference or otherwise cited during the prosecution history are also highly relevant in construing claim terms. For instance, art cited by or incorporated by reference may indicate how the inventor and others of skill in the art at the time of the invention understood certain terms and concepts. Additionally, the prosecution history may show that the inventors disclaimed or disavowed claim scope, or further explained the meaning of a claim term.

With regard to extrinsic evidence, I understand that all evidence external to the patent and prosecution history, including expert and inventor testimony, dictionaries, and learned treatises, can also be considered. For example, technical dictionaries may indicate how one of skill in the art used or understood the claim terms. However, I understand that extrinsic evidence is considered to be less reliable than intrinsic evidence, and for that reason is generally given less weight than intrinsic evidence.

I understand that in general, a term or phrase found in the introductory words or preamble of the claim, should be construed as a limitation if it recites essential structure or steps, or is necessary to give meaning to the claim. For instance, I understand preamble language may limit claim scope: (i) if dependence

on a preamble phrase for antecedent basis indicates a reliance on both the preamble and claim body to define the claimed invention; (ii) if reference to the preamble is necessary to understand limitations or terms in the claim body; or (iii) if the preamble recites additional structure or steps that the specification identifies as important.

On the other hand, I understand that a preamble term or phrase is not limiting where a challenged claim defines a structurally complete invention in the claim body and uses the preamble only to state a purpose or intended use for the invention. I understand that to make this determination, one should review the entire patent to gain an understanding of what the inventors claim they invented and intended to encompass in the claims.

I understand that 35 U.S.C. §112 ¶6 created an exception to the general rule of claim construction called a “means plus function” limitation. These types of terms and limitations should be interpreted to cover only the corresponding structure described in the specification, and equivalents thereof. I also understand that a limitation is presumed to be a means plus function limitation if (a) the claim limitation uses the phrase “means for”; (b) the “means for” is modified by functional language; and (c) the phrase “means for” is not modified by sufficient structure for achieving the specified function.

I understand that a structure is considered structurally equivalent to the corresponding structure identified in the specification only if the differences between them are insubstantial. For instance, if the structure performs the same function in substantially the same way to achieve substantially the same result. I further understand that a structural equivalent must have been available at the time of the issuance of the claim.