

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 Owner.

Case No. IPR2025-00486

U.S. Patent 10,873,753

PETITION FOR *INTER PARTES* REVIEW

OF U.S. PATENT NO. 10,873,753

TABLE OF CONTENTS

Exhibit List.....	4
I. Background.....	6
A. Blade Servers.....	6
B. Packets.....	7
C. Processors and Memory	8
II. The '753 patent.....	10
A. Prosecution History	11
B. PO's Allegations.....	12
C. Priority.....	13
III. Field and Level of Skill	13
IV. Claim Construction.....	14
A. "memory superposition"	15
B. "interleaved memory"	15
C. "swapped memory over a specified address range" (claims 4, 7, and 15).....	15
D. "memory allocated to specified address ranges according to parameterized task needs" (claims 4, 7, and 15).....	16
V. Invalidity Grounds.....	16
A. Ground 1	17
1. U.S. Patent App. Pub. 2013/0346667A1 ("Stroud")	17
2. U.S. Patent 9,148,389 ("DaSilva")	19
3. Claim 1	21

4.	Claim 2	53
5.	Claim 3	57
6.	Claim 13	60
7.	Claim 14	68
B.	Ground 2	68
1.	“Memory Systems and Pipelined Processors” (“Cragon”).....	68
2.	Claim 7	69
3.	Claim 9	81
4.	Claim 10	82
C.	Ground 3	82
1.	U.S. Patent App. Pub. 2002/0087821 (“Saulsbury”).....	82
2.	Claim 7	83
3.	Claims 9-10	93
D.	Ground 4	93
1.	Claim 4	94
2.	Claim 8	94
3.	Claim 15	94
VI.	Standing	95
VII.	Discretionary Analysis for Review.....	95
VIII.	Mandatory Notices and Fees	96

EXHIBIT LIST

Exhibit No.	Description
1001	U.S. Patent No. 10,873,753 (the '753 patent)
1002	Prosecution History of the '753 patent
1004	U.S. Patent App. Pub. No. 2013/0346667A1 ("Stroud")
1005	U.S. Patent No. 9,148,389 ("DaSilva")
1006	Harvey G. Cragon, Memory Systems and Pipelined Processors ("Cragon")
1007	U.S. Patent App. Pub. No. 2002/0087821 ("Saulsbury")
1008	U.S. Patent No. 8,015,386 ("Kulkarni")
1009	<i>Reserved</i>
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	<i>Reserved</i>
1013	<i>Reserved</i>
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	<i>Reserved</i>
1021	Exhibit C to Concurrent Ventures' and XtreamEdge's Oct. 11, 2024 Preliminary Infringement Contentions
1022	Declaration of J. Munford re Cragon
1023	IETF, RFC 6146 "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers"
1024	U.S. Patent No. RE45708 ("Frantz")
1025	U.S. Patent No. 9,270,592 ("Sites")
1026	U.S. Patent No. 5,155,833 ("Cullison")
1027	U.S. Patent No. 5,586,293 ("Baron")
1028	U.S. Patent App. Pub. No. 2018/0088853A1 ("Kotra")

Exhibit No.	Description
1029	U.S. Patent No. 9,959,212 (“Takeda”)
1030	U.S. Patent App. Pub. No. 2013/0318196A1 (“Yamamoto”)
1031	U.S. Patent App. Pub. No. 2006/0277365A1 (“Pong”)
1032	U.S. Patent App. Pub. No. 2006/ 0294308A1 (“Bailey”)
1033	U.S. Patent No. 10,847,196 (“Ware”)
1034	U.S. Patent No. 6,029,224 (“Asthana”)
1035	Concurrent Ventures’ and XstreamEdge’s March 29, 2024 Complaint in WDTX 1:24-cv-335
1036	Concurrent Ventures’ and XstreamEdge’s Jan. 23, 2025 Preliminary Claim Construction Disclosures
1037	Declaration of R. Jones re Cragon
1038	Microsoft, Microsoft Computer Dictionary (5th Ed., 2002) (excerpts)
1039	Chambers Dictionary (13th Ed., 2014) (excerpts)
1040	U.S. Patent App. Pub. No. 2014/0115137A1 (“Keisam”)
1041	Exhibit C to Concurrent Ventures’ and XstreamEdge’s Dec. 20, 2024 Supplemented Infringement Contentions
1042	Concurrent Ventures’ and XstreamEdge’s July 22, 2024 Response to Defendant’s Motion to Dismiss
1043	U.S. Patent No. 7,356,047 (“Mahalawat”)
1044	Chu, ENG-46158 rev 1.8 “Serial-GMII Specification”

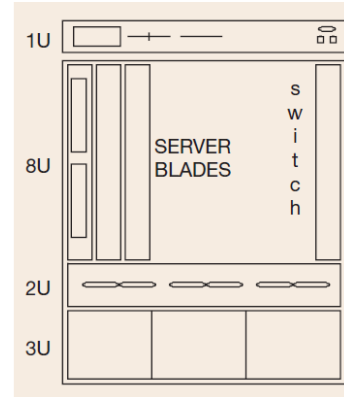
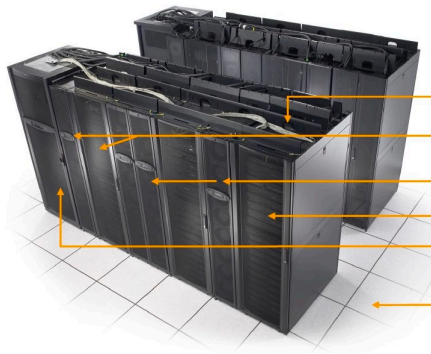
The '753 patent was directed to a common form factor in data centers: a chassis with swappable cards. However, since this was widely used by 2018, the claims were amended to recite particular identifiers in a packet header (e.g., “chassis identifier,” “board identifier,” etc.) and common memory arrangements (e.g., “multilevel cache,” “interleaved memory,” etc.).

Even with these additions, the claims are plainly invalid because the Stroud and DaSilva teach the recited packet header identifiers (Ground 1). A textbook Cragon teaches the recited memory arrangements (Ground 2). Moreover, Patent Owner (“PO”) alleged that the packet limitations are satisfied by conventional Ethernet and TCP/IP packets, while the memory limitations are satisfied by a network interface card with more than one type of memory. Ex-1003, ¶3.

I. BACKGROUND

A. Blade Servers

Blade servers and chassis platforms were common in data centers by the 2000s. To maximize space, modular platforms concentrated modules (cards/blades/boards/pods) in chassis. Ex-1015, 000014-15. Data centers networked multiple such chassis together. Ex-1040, ¶2. For example, IBM’s BladeCenter chassis included rows of racks, loaded with multiple blades, each blade being a server with processors, memory, disk drives, and Ethernet. Ex-1016, 000001-4.



Ex-1003, ¶¶4-6.

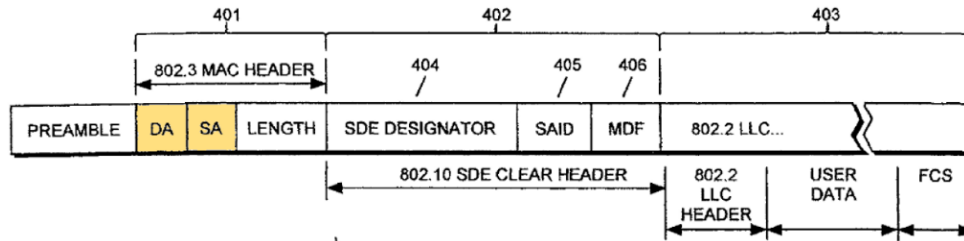
B. Packets

Packet-based networks have been the dominant form of computer networking for decades. Packets include a payload (with data) and a header with information for routing the packet, including source and destination addresses. Ex-1003, ¶7.

By the 2000s, Ethernet was the predominant layer 2 protocol for nearly any wired network and was widely used in data centers. Ethernet used packets, called frames, with destination and source MAC addresses (“DA” and “SA” respectively) in the packet header. Ex-1024, Fig. 4:¹

¹ Annotations/emphasis added unless otherwise noted.

(PRIOR ART)



Ex-1003, ¶¶8-10.

Most Internet communications used TCP/IP. In data centers, TCP/IP packets were typically sent over Ethernet. When two devices communicated using TCP/IP, they formed a TCP connection defined by a “five-tuple”: (1) source IP address, (2) source port, (3) destination IP address, (4) destination port, and (5) transport protocol. *See* Ex-1023, 000018; Ex-1025, Figs. 3A-3B; Ex-1003, ¶¶11-13.

C. Processors and Memory

Like all computers, blade servers included processing elements and memory. Common processing elements included CPUs, application-specific integrated circuits (“ASICs”), and programmable logic devices (“PLDs”).

PLDs are a special class of integrated circuit (“IC”) with logic circuits that users can configure after manufacturing, in contrast to general-purpose CPUs and ASICs with logic circuits that are fixed at manufacturing. PLDs allowed users to define the function of hardware circuits in the IC by specifying the logical

interconnections of those circuits, thereby creating custom hardware to perform custom functions. Ex-1010, 000004, 000009-11; Ex-1008, 1:30-37; Ex-1011, Fig. 4. Therefore, PLDs were considered a type of configurable logic device (“CLD”). Ex-1004, ¶107. Field Programmable Gate Arrays (“FPGAs”) were a popular type of PLD. Ex-1010; Ex-1017, 000087; Ex-1018; Ex-1003, ¶¶14-15.

Computers used different types of memories for different purposes, and blade servers were no different. For example, SRAM was often used as a small, fast cache, DRAM as larger main memory, and Flash solid-state disk (“SSD”) for non-volatile storage. Common memory arrangements included multilevel cache, interleaved memory, swap files, and memory allocated to specified address ranges. These were basic concepts taught in textbooks. E.g., Ex-1007; Ex-1003, ¶¶16-20.

Memory was often used in selectable arrangements. By the 1980s, it was known for computers to use “an arrangement...for selectively causing the memory...to selectively serve as either a boot memory or a cache memory[.]” Ex-1026, 1:48-58. By the 1990s, it was known for processors to have “on-chip memory having standard and cache operating modes.” Ex-1027, Title, Abs., Fig. 4. Selectable arrangements were applied to various memory types because memory was a limited resource, and each mode of operation had known benefits. E.g., Ex-1028; Ex-1029, Abs.; Ex-1030, Abs.; Ex-1003, ¶¶18-19.

II. THE '753 PATENT

The '753 patent is directed to “one or more swappable pods or cards in one or more chassis, coupled through a module messaging interface network.” Ex-1001, Abs. The pods/cards contain software modules and hardware modules (such as FPGAs or PLDs). The messaging interface network includes “Ethernet...or any other suitable connection.” Ex-1001, 3:49-46.

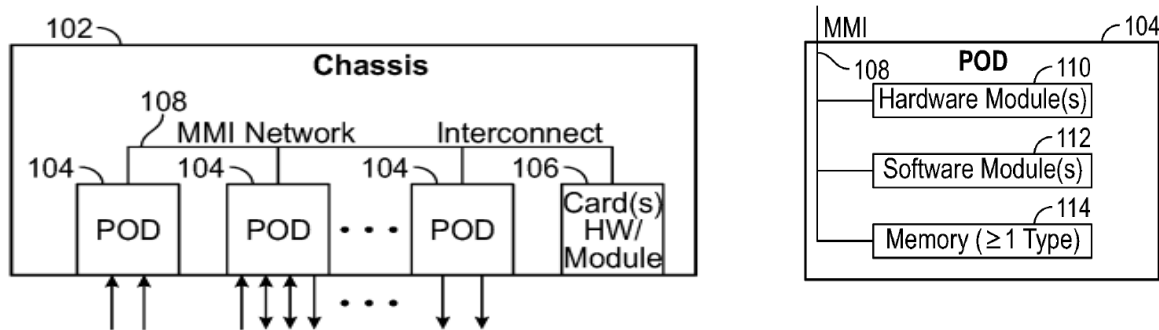


FIG. 1A

Ex-1003, ¶¶21-22.

Based on the prosecution history, the purportedly novel feature includes several identifiers in the packet header. Ex-1001, 3:4-7.

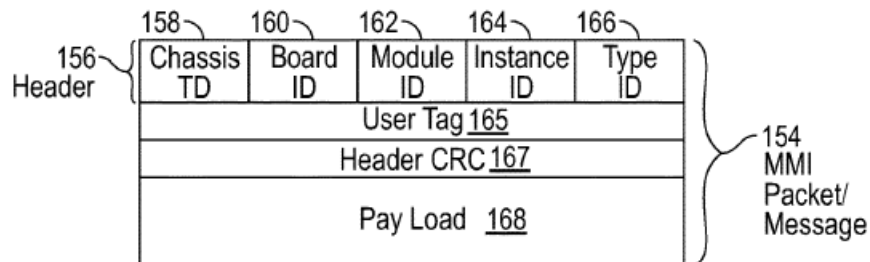


FIG. 6A

Ex-1003, ¶22.

The '753 patent also refers to something it calls “memory superposition” where “two or more types of memory, such as SRAM, DRAM, FRAM, RLDRAM, flash...etc., are made available[.]” Ex-1001, 7:1-15 (parenthesis/citations omitted); Ex-1003, ¶23.

However, as explained above, it was common for blade server chassis to include swappable cards connected through a backplane using Ethernet (*supra* §I.A-B), and the recited packet header identifiers were taught by Stroud and DaSilva. Moreover, it was common for blades to include multiple types of memory for data processing operations, given the network context in which they were used. *Supra* §I.C; Ex-1003, ¶24.

A. Prosecution History

The originally-filed independent claims were directed to a chassis with swappable cards having hardware or software modules. After an Office Action rejection (Ex-1002, 000083-92), the Applicant amended the independent claims to include limitations originally recited in dependent claims (Ex-1002, 000107-114, 000090-91), leading to allowance. Claim 1 was amended to recite packet header identifiers: “a chassis identifier, a board identifier, a module identifier, an instance identifier, and a type identifier.” Claim 7 was amended to recite “a plurality of types of memory and memory superposition with selectable arrangement of the plurality of types of memory as multilevel cache, interleaved memory, swapped

memory over a specified address range, and memory allocated to specified address ranges according to parameterized task needs.” Ex-1002, 000119; Ex-1003, ¶¶25-27.

B. PO’s Allegations

In district court, PO alleged infringement of the ’753 patent based on a network card (called a “DSC”) supporting conventional TCP/IP packets and Ethernet. Ex-1035, ¶51; Ex-1021, 000004. According to PO, the claimed packet header identifiers are satisfied by the “five-tuple” in traditional TCP/IP packet headers, when sent over Ethernet (and thus having Ethernet “MAC addresses”). Ex-1042, 000021; Ex-1021, 000005:

Table: Packet 1 - Redirect Action - Test Packet Header Information

Receiving Port	Destination MAC	Destination IP	Destination Port	Source IP	Source Port	Protocol
Eth 1/2	00:00:00:00:10:02	1.2.3.4	456	2.3.4.5	345	UDP

Ex-1003, ¶¶28-32.

In supplemental contentions, PO reiterated the allegation that a “MAC address” and “IP” address are “examples of ‘chassis identifier[s]’ and ‘board identifiers’” while expanding the allegations to encompass a wide range of other fields as well. Ex-1041, 000005. PO alleged that the claimed packet header identifiers can be satisfied by any data in a packet, even if not part of the packet header. *Id.*, 000005-6; Ex-1003, ¶33.

PO alleged the entire memory superposition claim limitation is satisfied by the mere presence of multiple types of memory, such as SRAM and DRAM: “[t]he Accused Products support/handle various types of memory to optimize performance[.]” Ex-1021, 000006; Ex-1041, 000009-11; Ex-1003, ¶34.

C. Priority

Petitioner demonstrates invalidity as of July 15, 2018. Ex-1003, ¶28.

III. FIELD AND LEVEL OF SKILL

The patent’s field is data processing and streaming architectures including user-defined hardware and software modules. That is the focus of the claims and specification and how the patent describes its field and related art. Ex-1001, 1:6-21, 2:47-3:20, Abs., Title; Ex-1003, ¶35.

A POSITA would have been a person with a bachelor’s degree in electrical engineering, computer engineering, or similar major with at least two years of experience in the field (described above) or a master’s degree in electrical engineering, computer engineering, or similar major with specialization in the field. A person with less education but more relevant practical experience may also meet this standard. Ex-1003, ¶¶36-37.

A POSITA of this background in 2017 would have had the knowledge and skills to:

- Design, analyze, evaluate, and modify data-processing/network systems;
- Design computer systems and networks supporting common networking protocols;
- Utilize Ethernet and IP protocols;
- Modify Ethernet and IP packets;
- Implement known network traffic management principles, including routing based on MAC addresses;
- Program CPUs and network processors for data-flow and data-processing applications;
- Utilize and implement common memory arrangements;
- Modify existing CPU designs to implement known memory architectures;
- Design printed circuit board cards; and
- Implement CLD, PLD, and FPGA designs.

Ex-1003, ¶38.

IV. CLAIM CONSTRUCTION

The following constructions were proposed in district court. The prior art teaches these terms irrespective of whether these constructions are adopted.

A. “memory superposition”

In district court, PO proposed the construction for this term as “two or more types of memory that can be made available for a data flow or other data processing operation.” Ex-1036, 000003; Ex-1001, 7:1-9; Ex-1003, ¶41.

Petitioners adopt this construction for the purposes of this Petition.

B. “interleaved memory”

In the only discussion of “interleaved memory” in the patent, the claims and specification state that it is a type of memory superposition for a “plurality of types of memory” (Ex-1001, claims 4/5/15, 7:1-9) “to achieve an average access bandwidth” where “[i]nput bandwidth, output bandwidth, or power could be *blended among different types* of RAM or drives” (*id.*, 8:35-40). Therefore, interleaved memory means an arrangement of different types of memory that are alternately accessed to achieve an average access bandwidth. Ex-1003, ¶¶43-45.

C. “swapped memory over a specified address range” (claims 4, 7, and 15)

The patent does not explain swapped memory, other than stating it is “another arrangement of memory superposition.” Ex-1004, 8:41-42. Because it is an arrangement of memory superposition, the term at least involves the use of “two or more types of memory.” Ex-1001, 7:1-9; Ex-1003, ¶¶47-48.

Swapped memory is a term of art. As technical dictionaries explain, swapping “involves moving blocks of information in units known as pages

between memory and disk as they are needed during the execution of the application” (Ex-1038, 000003) using a “swap file” on non-volatile storage. Ex-1039, 000003. Therefore, this term means memory with a specified address range where information is moved between the memory and non-volatile storage. Ex-1003, ¶¶49-50.

D. “memory allocated to specified address ranges according to parameterized task needs” (claims 4, 7, and 15)

This term means different types of memory allocated to specified address ranges according to parameters for task needs. That is how the claims (4, 7, 15) and specification describe the term (Ex-1004, 8:22-31, 7:1-9) and is supported by examples in the specification. Ex-1001, 7:21-8:21; Ex-1003, ¶¶52-56.

V. INVALIDITY GROUNDS

Petitioner requests *inter partes* review and cancelation of the challenged claims on the following grounds:

Grounds	Claims	Statutory Basis	Prior Art
1	1-3, 13, 14	§103	Stroud and DaSilva
2	7, 9-10	§103	Stroud and Cragon
3	7, 9-10	§103	Stroud, Cragon, and Saulsbury
4	4, 8, 15	§103	Stroud, DaSilva, Cragon, and Saulsbury

The Grounds render the challenged claims obvious because any differences between the claimed subject matter and the prior art are such that the subject

matter, as a whole, would have been obvious at the time the invention was made to a POSITA to which the subject matter pertains. *KSR Int’l Co. v. Teleflex Inc.*, 550 U.S. 398, 406 (2007).

A. Ground 1

Stroud teaches a chassis with multiple cards that communicate using Ethernet with packet header identifiers recited in claim 1. Ex-1004, ¶¶367-372. DaSilva teaches an improvement for networks with multiple chassis, where packet headers include a “chassis ID” to indicate a particular chassis. Ex-1005, 4:23-5:24. Since it was common to use multiple chassis in data centers (Ex-1040, ¶2), Ground 1 combines DaSilva’s teachings with Stroud. The motivation is explained below. *Infra* §V.A.3[1e] (motivation subsection); Ex-1003, ¶¶58-59.

1. U.S. Patent App. Pub. 2013/0346667A1 (“Stroud”)

Stroud is §102(a)(1) prior art published in 2013. Ex-1003, ¶¶60-61.

Stroud teaches a chassis platform with “modular” cards, each having configurable logic devices (“CLDs”) and network processors. Ex-1004, ¶¶98-107, Fig. 3:

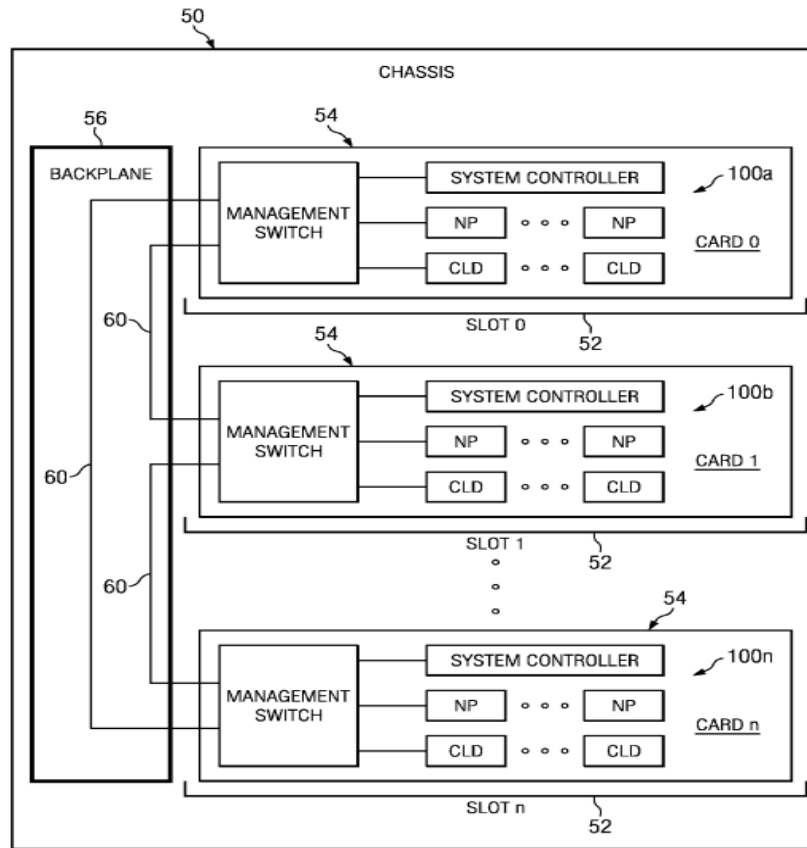


FIG. 3

Ex-1003, ¶62.

The cards communicate over an Ethernet backplane. Ex-1004, ¶99, ¶101.

To route packets to the correct destination, Stroud teaches subfields within the MAC address with several identifiers. Ex-1004, ¶¶366-372; Ex-1003, ¶63. While Stroud uses several exemplary embodiments, Stroud's teachings are meant to be used together because Stroud presents these examples as part of the same modular chassis embodiment/platform. Ex-1004, ¶¶85-86, ¶79. Stroud's interrelated figures describe common components in the chassis, how cards operate within a chassis or network. *Id.*, ¶66, ¶¶87-98, ¶176, ¶376. Several exemplary card

architectures are presented, which work together in a chassis. Ex-1004, ¶176, ¶343, ¶365, ¶¶100-101, Figs. 6A-9A. Therefore, a POSITA would have been motivated to combine Stroud's teachings for the benefits Stroud describes. Ex-1004, ¶5; Ex-1003, ¶¶64-65.

Stroud is analogous art in the same field (*supra* §III) and reasonably pertinent to the '753 patent because Stroud presents an architecture for processing data flows (e.g., generating/simulating/processing network traffic) that combines FPGA hardware modules and software modules. Ex-1004, ¶86, ¶298, ¶101, ¶¶106-113, ¶144, ¶373; Ex-1003, ¶66.

2. U.S. Patent 9,148,389 ("DaSilva")

DaSilva is §§102(a)(1)-(2) prior art filed in 2012 and issued in 2015. Ex-1003, ¶¶67-68. DaSilva discloses chassis with multiple cards. To distinguish between chassis, DaSilva uses a "chassis ID." Ex-1005, Abs.:

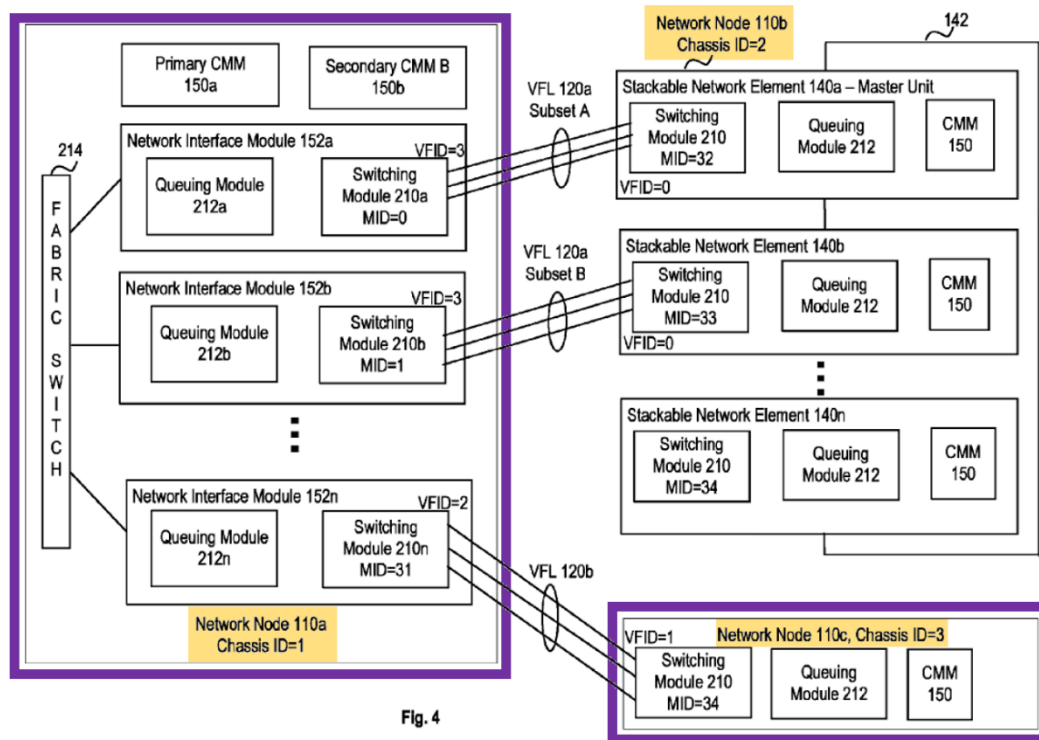


Fig. 4

Ex-1003, ¶69.

DaSilva is analogous art in the same field (*supra* §III) and reasonably pertinent because it is directed to architectures that allow the flexible configuration of packet processing and routing nodes in a network, including data streaming flows e.g., “Voice over-IP (VoIP), data and video applications.” Ex-1005, 1:45-2:26, 1:40-43. DaSilva recognizes “it is sometimes difficult to upgrade or transition” processing elements used in network data processing architectures. Ex-1005, 2:9-22; Ex-1003, ¶70.

3. Claim 1

[1pre]. A platform for data flow processing, comprising:

Stroud teaches a “testing and simulation platform” (Ex-1004, ¶¶85-87) that processes “data flows” in a network (Ex-1004, ¶298). Ex-1004, ¶¶80-81, Fig. 1:

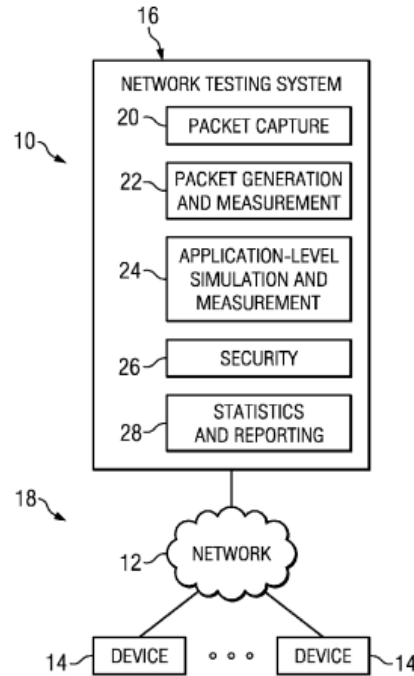


FIG. 1

The platform processes data flows by, e.g., simulating network traffic to test the “traffic-handling performance” and “security” of network devices. Ex-1004, ¶¶80-81, ¶84. The platform also processes data flows by generating “synthetic data flows” to run data-processing applications (“e.g., database server, Web server”). Ex-1004, ¶84, ¶¶89-93, ¶298. The platform processes the generated packet data flowing through the network. Ex-1004, ¶86, ¶219 (deep packet inspection for

“analyz[ing]...application flows”), ¶¶220-250 (discussing exemplary data flow processing); Ex-1003, ¶72.

The platform further comprises elements discussed for subsequent limitations. Ex-1003, ¶73.

[1a] **one or more swappable pods or cards in one or more chassis, coupled through a messaging interface network;**

Stroud teaches **one or more swappable pods or cards in one or more chassis**. Ex-1004, ¶98 (“a chassis 50...configured to receive a modular card, or blade”), ¶376 (“chassis 50 that interconnects multiple cards 54”), ¶365, ¶¶376-377, ¶66, ¶714.

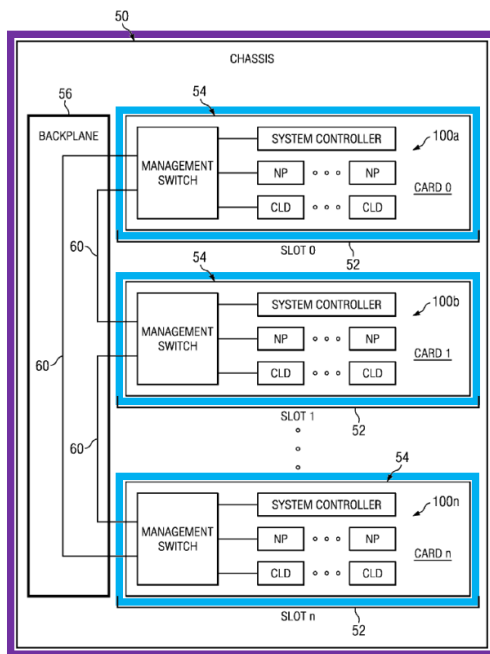


FIG. 3

Ex-1003, ¶¶75-76.

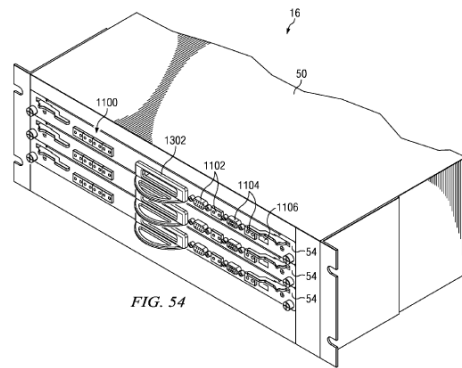
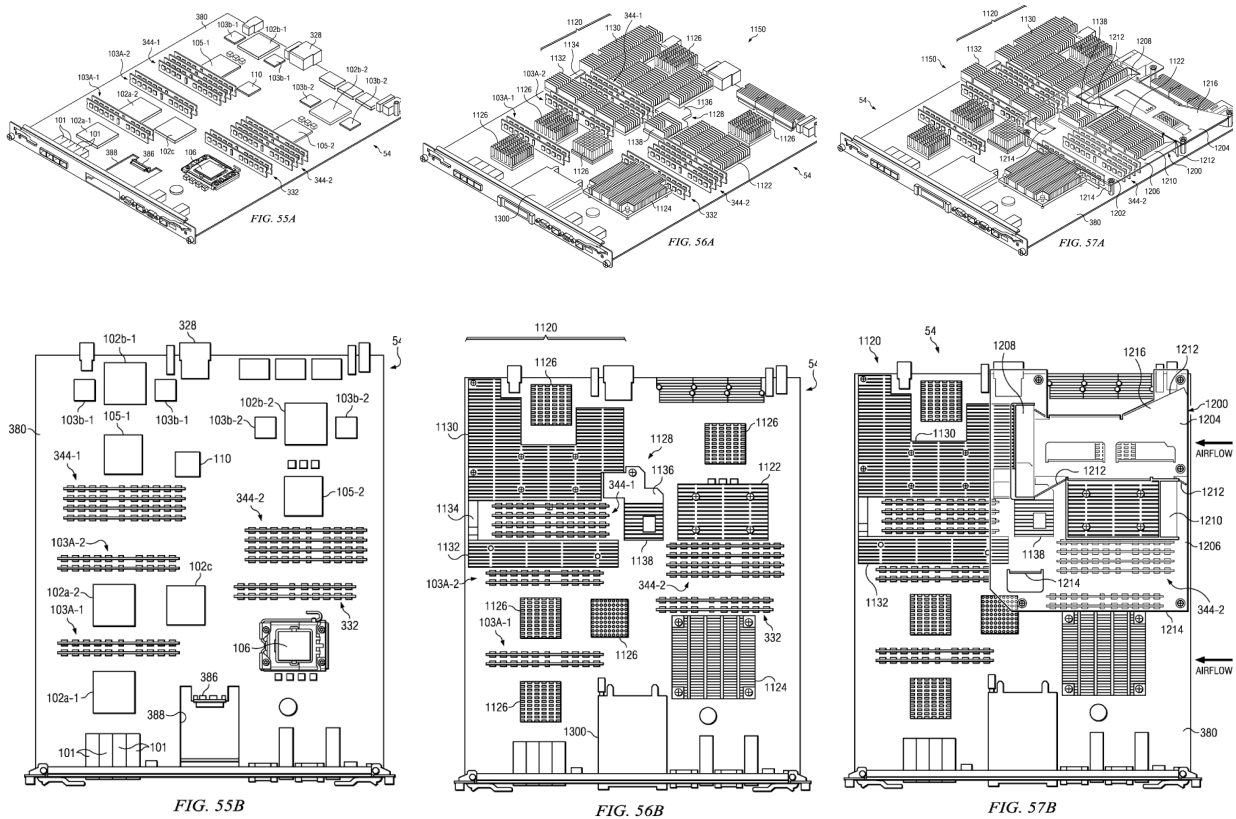


FIG. 54

Stroud’s cards are blades² that satisfy the claimed “swappable cards” and “swappable pods,” but for simplicity, this Petition shall refer to cards. The cards/blades comprise one or more circuit boards. *Id.*, ¶¶98. Each blade is a module that comprises a large card. *Id.*, ¶¶115, ¶¶176, ¶¶382, Figs. 54-57B.

Card/Blade



Ex-1003, ¶¶77-78.

² Stroud uses the terms “card” and “blade” interchangeably. *E.g.*, Ex-1004, ¶¶98, ¶¶13, 33, ¶¶176.

Stroud explains the modular cards/blades are swappable. Ex-1004, ¶382 (“hot-swapped”), ¶378, ¶98 (slots receive cards). The cards are modular and meant to be swapped into different combinations or numbers. Ex-1004, ¶176 (“configured to support any suitable number of cards”). The cards have features to support their insertion and removal from the slots for swapping. Ex-1004, ¶714, ¶382; Ex-1003, ¶78.

Stroud teaches the swappable cards are **coupled through a messaging interface network**. For example, the cards are connected to each other through an Ethernet network that goes through physical backplane interfaces. Ex-1004, ¶101 (“Cards 54 may be communicatively coupled to each other via the backplane 56”), ¶99, ¶¶113-115, ¶176, ¶¶353-354, ¶¶376-378, ¶382, ¶393.

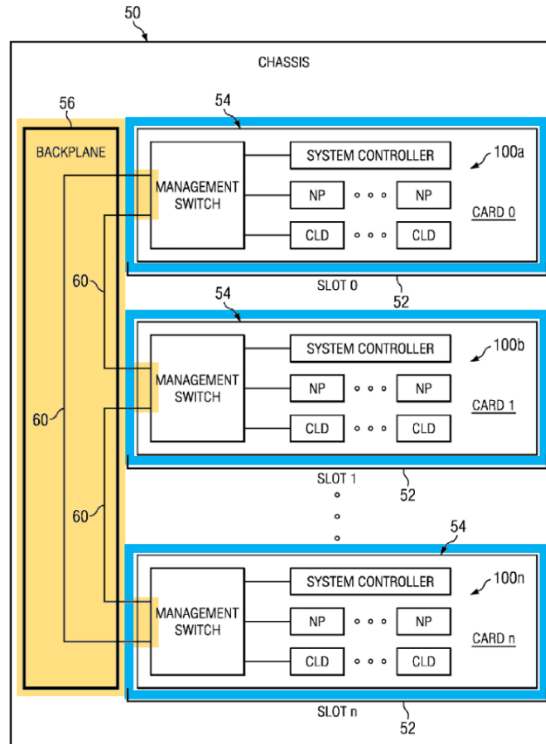


FIG. 3

Cards are “connected to each other” through a packet-based message network: “Ethernet over the backplane[.]” Ex-1004, ¶377, ¶176, ¶393, ¶389; Ex-1003, ¶¶79-80.

Stroud further describes how the cards in the chassis are “communicatively coupled to each other via the backplane 56.” Ex-1004, ¶101. Each card contains a “management switch” that connects to the interface on the backplane. Ex-1004, ¶113, ¶114, Figs. 4-5, ¶¶314-319 (exemplary backplane connection using SGMII). SGMII provides a serial LVDS interface for Ethernet packets in high-speed networking applications. Ex-1043, 2:10-35; Ex-1044, 000001-3, 000008; Ex-1003, ¶81.

The cards send messages (e.g., Ethernet packets) via the management switch through the backplane interface network. Ex-1004, ¶174 (“[M]anagement switch 110 may essentially act as a router, allowing control CPUs 106 to communication with network processor 105 and CLD 102 on the same card 54 and other cards 54 in the system 16.”), ¶¶113-115, ¶365, Fig. 3. The cards generate messages (e.g., Ex-1004, ¶349 (“control messages”), ¶361 (“error message”)) communicated through Ethernet. The messages include card-internal communications and external messages for other cards or devices. Ex-1004, ¶399, ¶99; Ex-1003, ¶82.

[1b] each of the one or more swappable pods or cards having one or more hardware modules or one or more software modules;

Stroud teaches that each **card** “include[s] any architecture 100 of *hardware, software, and/or firmware components*.” Ex-1004, ¶100, ¶85, Fig. 3. As shown in Stroud’s exemplary architectures, the cards comprise **software** and **hardware** modules of “a **controller** 106, two **network processors** 105 and **multiple CLDs** 102 coupled to a management switch 110.” Ex-1004, ¶¶103-113, ¶114, ¶¶456-459, Figs. 3-5, 14A-14B:

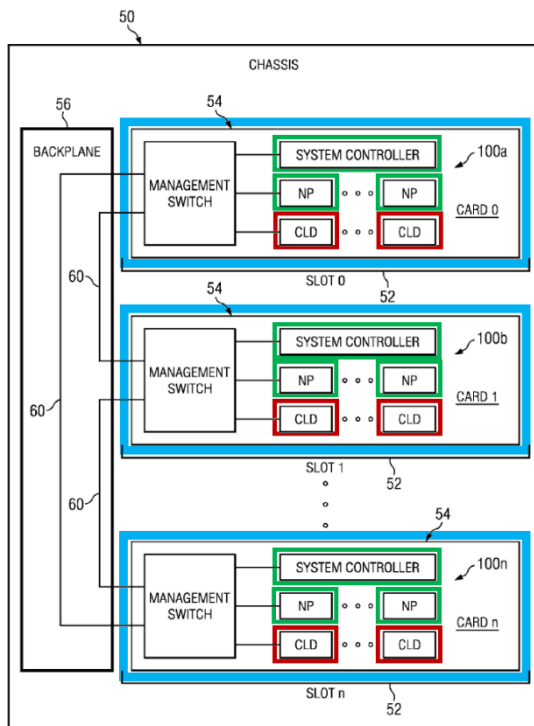


FIG. 3

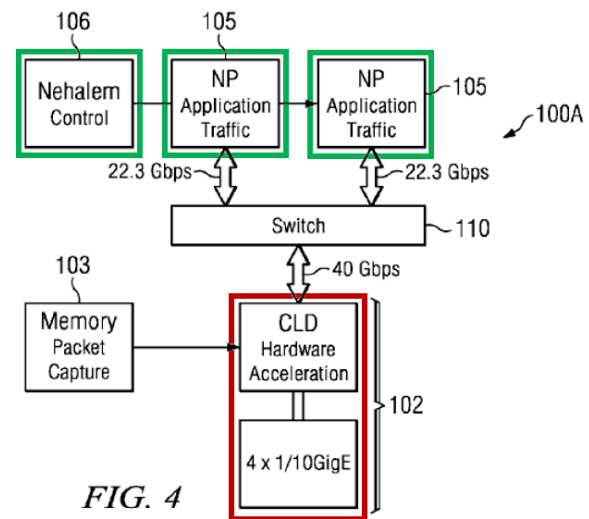


FIG. 4

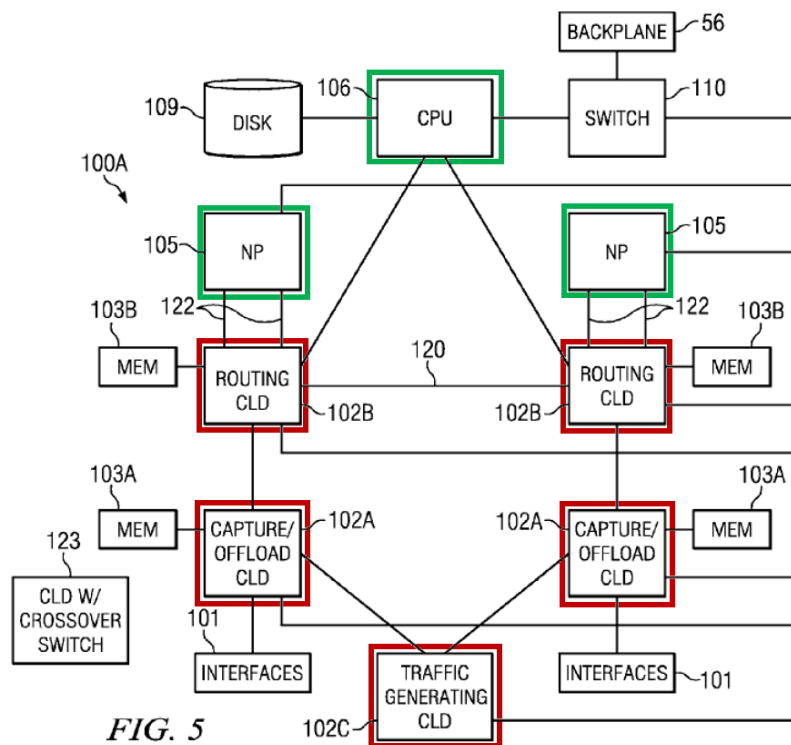
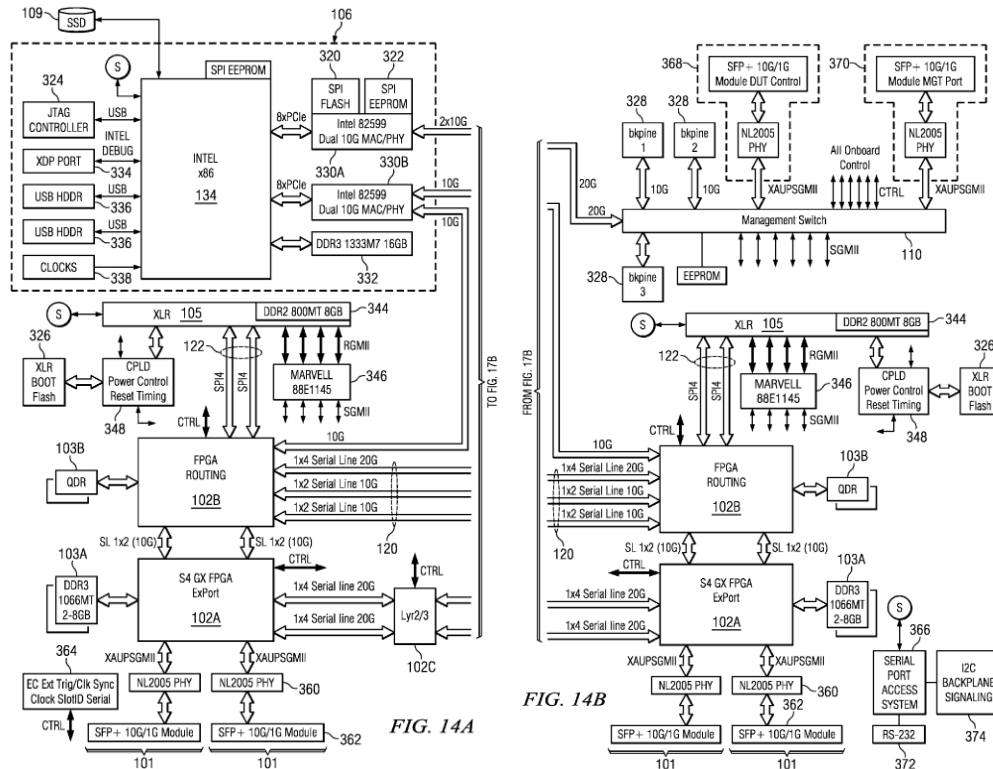


FIG. 5



Ex-1003, ¶¶84-85.

Regarding **hardware modules**, Stroud teaches that the configurable logic devices (“CLDs”) are hardware modules with “programmable logic units,” for example “programmable logic devices (PLDs)” and “FPGAs.” Ex-1004, ¶107, ¶¶110-112, ¶¶311-313. Stroud teaches the CLDs are hardware modules, because they include user-programmable “configurable logic,” where the user programs and alters the connection pattern of the circuits, creating custom circuits for

performing custom functions in hardware.³ Ex-1004, ¶¶107-112, ¶117, ¶¶311-313; Ex-1003, ¶¶86-87.

Regarding **software modules**, Stroud teaches “application software...executed on network processor [(“NP”)]”. Ex-1004, ¶309. Stroud teaches several exemplary applications, including application-level simulation and measurement with deep packet inspection for “analyz[ing] traffic beyond individual packets” and “instead look[ing] at application flows.” *Id.*, ¶219. These applications are provided, in part, by software running on NPs. *Id.*, ¶224, ¶229, ¶464. In general, NPs run software programs to generate, receive, and process network data packets. Ex-1004, ¶105. NPs also execute software for the TCP/IP stack and user- and application-layer simulations. *Id.*, ¶¶149-153; Ex-1003, ¶¶88-89.

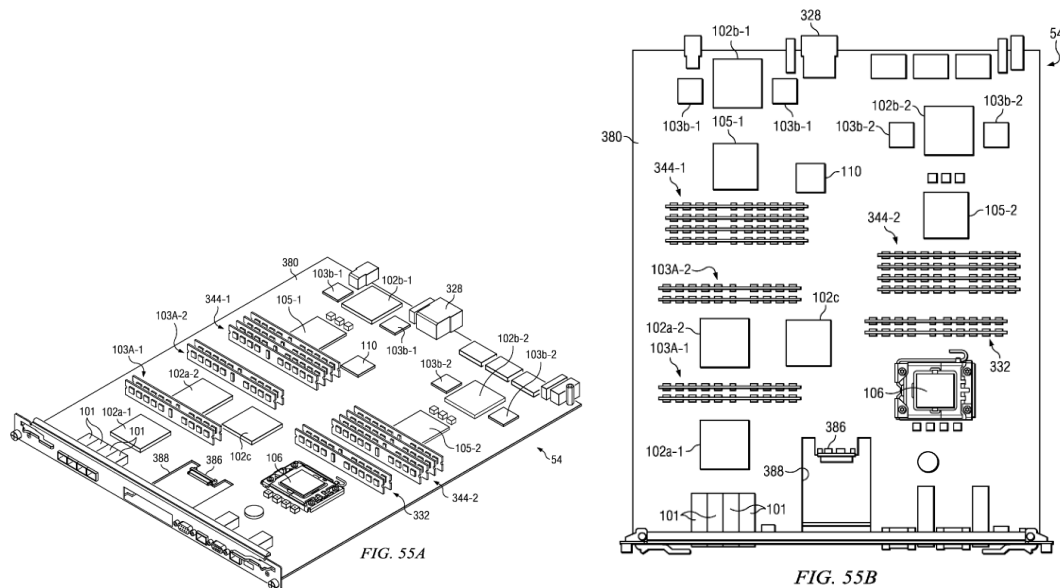
Each card also includes software modules running on controller 106. Ex-1004, ¶188 (“management processor 134 of controller 106 configured to execute controller software 132”), ¶208, ¶227, ¶262 (“management processor 134 on controller 106 configured to execute the controller software 132, collect and store statistics, and/or generate malicious application traffic”), ¶263 (“the management

³ The ’753 patent’s examples confirm that Stroud’s FPGAs fall within the scope of the claimed “hardware modules.” Ex-1001, 5:19-29.

processor 134 of controller 106 can generate malicious traffic packet-by-packet and forward these to the network processor 105 as if they were generated locally”), ¶266 (“management processor 134 of controller 106, in addition to providing a place for much of the control software for various subsystems to execute, may also host a statistics database 160 and reporting engine 162”). The controller runs software applications and further provides software for execution on NPs. Ex-1004, ¶234; Ex-1003, ¶90.

The software module for each NP and controller includes a processor for executing software (Ex-1004, ¶144 (“Each network processor (NP) 105 may be a general purpose CPU with multiple cores, security, and network acceleration engines.”), ¶105, ¶188) and the software itself. Each card runs a “software image.” Ex-1004, ¶376. Stroud teaches storing the software on, e.g., a hard drive. *Id.*, ¶158 (“In some embodiments, all firmware or software...resides on the disk drive 109.”), Fig. 5-7A; Ex-1003, ¶91.

Stroud provides further details of the physical layout of controllers, NPs, CLDs, and other components on a card. Ex-1004, ¶717, Figs. 55A-55B:



These correspond to the controller, NPs, and CLDs “shown in FIGS. 14A and 14B” (*id.*, ¶717) that comprise hardware and software modules in the swappable cards. Ex-1003, ¶92.

[1c] one or more of the plurality of swappable pods or cards having a portion for user-definable hardware modules or user-definable software modules;

Stroud teaches **one or more of the plurality of swappable pods or cards having a portion for user-definable hardware modules** (e.g., CLD modules) or **user-definable software modules** (e.g., controller or network processor modules). For example, Stroud teaches that each **card** has one portion (highlighted red below) with multiple **CLD** modules and another portion (highlighted green below) having multiple **NP** modules and/or a **controller** module. Ex-1004, ¶83, ¶¶103-109, ¶188.

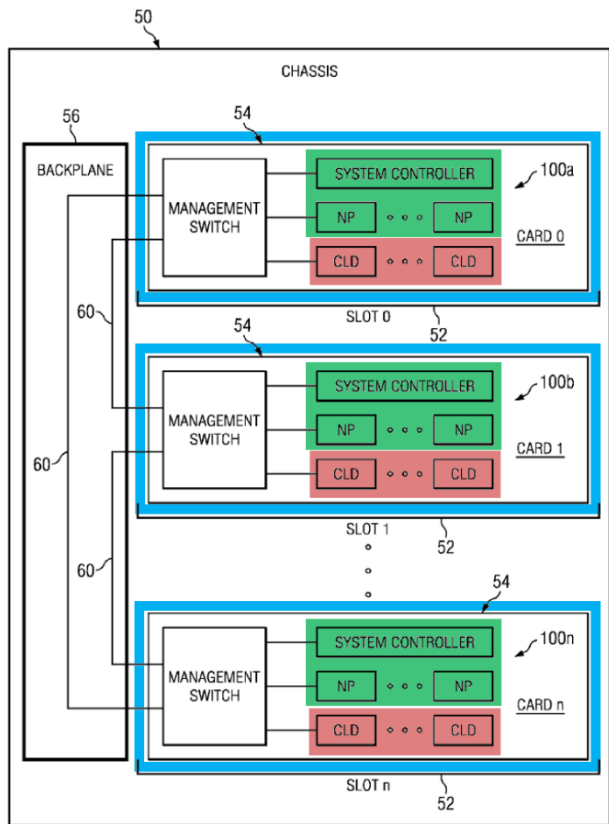


FIG. 3

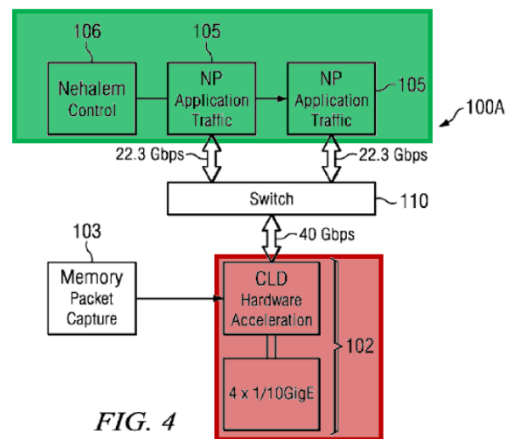
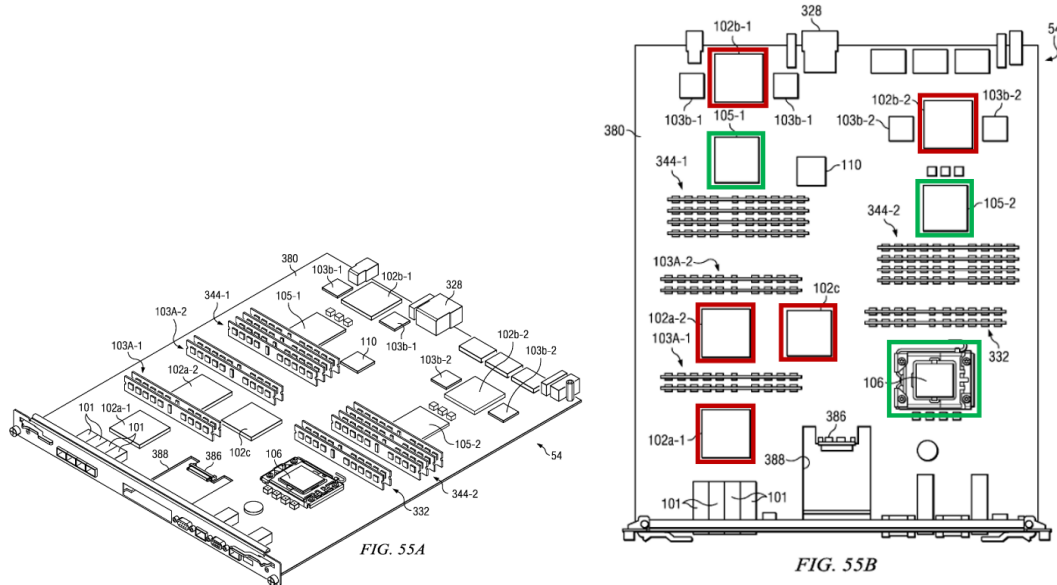


FIG. 4

Ex-1003, ¶94.

Stroud also teaches exemplary physical layouts showing the portions of the card with hardware or software modules. The claimed “portion” is satisfied by a portion of Stroud’s card that contains CLDs, NPs, and/or a controller. Ex-1004, ¶717, Ex-1003, ¶95:



For example, the card includes a **portion for user-definable hardware modules**. As discussed above, Stroud’s CLD modules are hardware modules with user-programmable logic, including, for example, “FPGAs” and “PLDs”. *Supra* §V.A.3[1b]; Ex-1004 ¶¶85, ¶100, ¶107, ¶¶109-112, ¶114, ¶117, ¶¶457-459, Figs. 3-5, 14A-14B. With Stroud’s CLD modules, the user defines the function of the hardware by altering the logical interconnections of the circuits in the chip. *Supra* §V.A.3[1b]. Stroud’s CLDs include “blocks of configurable logic” where those logical circuit connections can be programmed and reprogrammed by the user. Ex-1004, ¶¶106-110, , ¶117. Stroud teaches detailed steps for loading custom logic onto the CLD. Ex-1004, ¶¶63-64, ¶¶689-700; Ex-1003, ¶¶96-97.

Each card includes a **portion for user-definable software modules**. As previously discussed, each **card** has a portion for multiple **NP modules** and a **controller** that run user-defined application software. *Supra* §V.A.3[1b]; Ex-1004,

¶103 (“[E]xample architecture 100A...may include a controller 106, two network processors 105....”), ¶105, ¶144, ¶149, ¶¶150-153, ¶158, ¶188, ¶208, ¶219, ¶223, ¶227, ¶229, ¶234, ¶¶262-263, ¶266, ¶309, ¶376, ¶464, Figs. 3-5, 14A-14B.

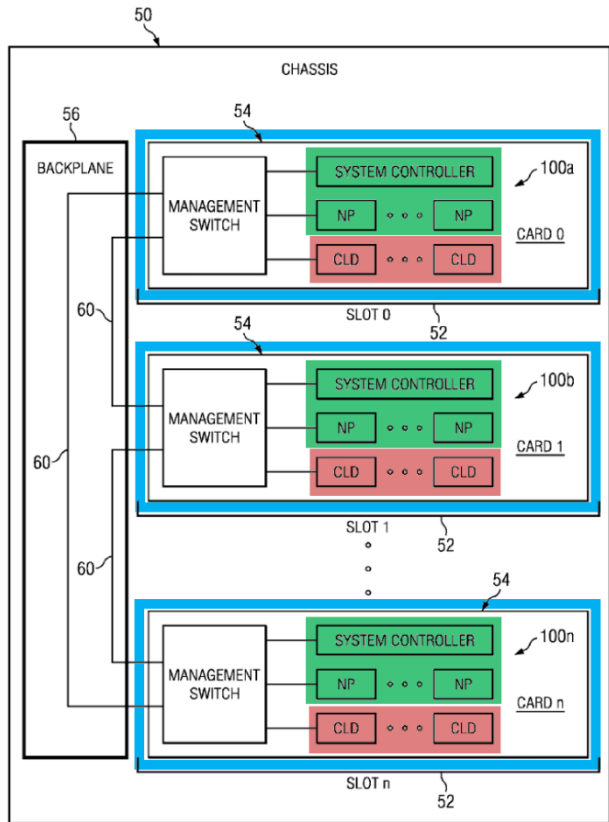


FIG. 3

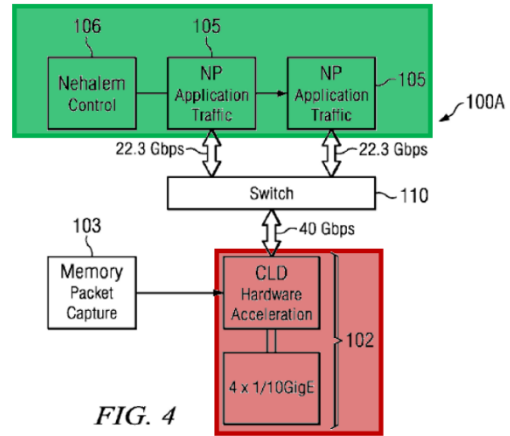


FIG. 4

Ex-1003, ¶98.

The NPs and controller are user-definable software modules because the user defines Stroud’s software modules with software programs that run on NPs and the controller for the user’s application, including for example network packet generation and processing, TCP/IP stack, user and application-layer simulation, deep-packet analysis, and/or collecting network statistics. Ex-1004, ¶158, ¶105, ¶¶219-220, ¶223, ¶144, ¶¶149-153, ¶188, ¶¶227-229, ¶208, ¶234, ¶¶262-263, ¶266,

¶309, ¶464, Figs. 3-5, 14A-14B. The software programs are stored on non-volatile storage, e.g., “removable disk storage109.” Ex-1004, ¶158, ¶¶170-173; Ex-1003, ¶99.

To the extent PO argues the claim refers to a single portion, it is satisfied by a portion on Stroud’s card that has hardware and software modules. Ex-1004, Fig. 3 (green and red):

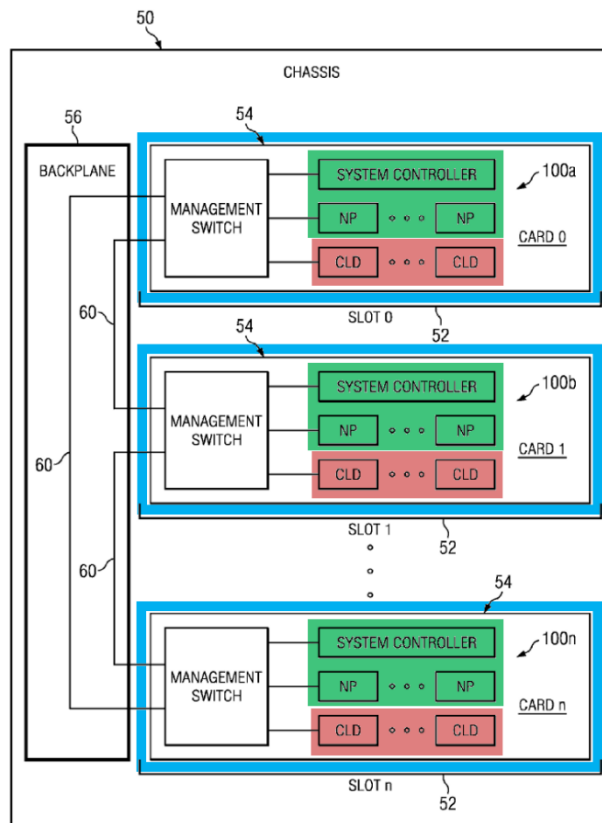


FIG. 3

Additionally, Stroud teaches configuring the CLDs to implement general purpose processors. Ex-1004, ¶112, ¶321; Ex-1003, ¶100.

[1d] the plurality of swappable pods or cards being user-configurable to implement data flow processing architectures; and
--

Stroud teaches the **plurality of swappable pods or cards being user-configurable**. As discussed above, Stroud teaches a plurality of swappable pods or cards; each card has user-definable hardware and software modules. *Supra* §V.A.3[1b]-[1c]; Ex-1003, ¶102.

Each hardware module comprises a “*configurable logic device (CLD)*” that “includes a set of programmable logic units” where the user alters the connection pattern of the circuits to define the function of the hardware. Ex-1004, ¶107, ¶117; *supra* §V.A.3[1b]. Stroud teaches FPGAs and “programmable logic devices (PLDs)” as examples—these were common CLDs where the user programmed and configured the hardware circuits, rather than the manufacturer. Ex-1004, ¶109, ¶117. Stroud teaches that configurable logic devices are user-configurable because the user configures the programmable logic units by altering the logical connections of circuits, creating custom hardware (with custom circuits) for performing specific functions. Ex-1004, ¶¶107-112. Therefore, Stroud teaches that the CLDs are user-configurable for performing the user’s applications. *See id*; Ex-1003, ¶103.

The user configures software modules using software programs that run on NPs and the controller, including, for example, software configured to perform

network packet generation and processing, TCP/IP stack, user and application-layer simulation, deep-packet analysis, and/or collecting network statistics. Ex-1004, ¶158, ¶105, ¶¶219-220, ¶223, ¶144, ¶¶149-153, ¶188, ¶¶227-229, ¶208, ¶234, ¶¶262-263, ¶266, ¶309, ¶464, Figs. 3-5, 14A-14B; *supra* §V.A.3[1c]. The user's software programs are stored on non-volatile storage, e.g., a hard disk, attached to the card. Ex-1004, ¶158, ¶¶170-173; Ex-1003, ¶104.

Therefore, Stroud's cards are user-configurable because the user configures the hardware modules and software modules on each card. Additionally, the user configures the overall function of the card and the way in which the hardware and software modules work together to perform the user's application. *E.g.*, Ex-1004, ¶84 ("e.g., database server, Web server," etc.), ¶¶89-93; Ex-1003, ¶105.

Stroud teaches the plurality of swappable pods or cards being user-configurable to **implement data flow processing architectures**. As discussed above, Stroud's platform is used for data flow processing. *Supra* §V.A.3[1pre]. For example, Stroud teaches using "[c]onfigurable logic devices" on the swappable cards to "provide high-performance, specialized computation, data transfer, and data analysis capabilities to process certain data or computation intensive tasks at or near the network line rates." Ex-1004, ¶106. By processing data at network-line rates, Stroud's user-configurable cards implement data-flow processing architectures that perform computation and analysis on packetized data flows at the

speed in which data is transmitted over the network. *Id.* Stroud uses the swappable cards in “the operation of a test system” architecture (Ex-1004, ¶84) that generates and processes “synthetic data flows” (Ex-1004, ¶298); Ex-1003, ¶106.

In particular, the cards in Stroud’s architecture are configured to implement data flow processing architectures such as “(1) high-speed, high-resolution network packet capture, (2) high-speed packet generation and measurement, (3) application-level simulation and measurement, (4) security and exploit simulation and analysis, and (5) statistics collection and reporting..” *Id.*, ¶177. Therefore, Stroud configures its cards to implement several data flow processing architectures that process packetized data flows by, for example, capturing packets, generating and measuring packets, simulating and measuring network traffic at the application level, analyzing security in network data flows, simulating security exploits in packet traffic, and processing data flows to collect and report statistics of the data flow. *Id.*; Ex-1003, ¶107.

[1e] **a network coupled to the one or more swappable pods or cards and supporting messaging-based communication using packets each having a header with a chassis identifier, a board identifier, a module identifier, an instance identifier, and a type identifier, so that each type of module, each instance of a type of module, and each module on each board in each chassis can be addressed through the header.**

Stroud teaches **a network coupled to the one or more swappable pods or cards**, with cards in the chassis “communicatively coupled to each other via the backplane 56” using an Ethernet network. Ex-1004, ¶¶99-101, ¶115 (“[M]ultiple blades 54 may be connected together via backplane 54 to create larger systems.”), ¶¶174-176. Each card is coupled to the Ethernet backplane interface network via its management switch; therefore the network is coupled to the cards. *Supra* §V.A.3[1a]; ¶101, ¶176; Ex-1003, ¶109.

Stroud teaches the **network supports messaging based communication using packets**, with the backplane network supporting Ethernet.⁴ Ex-1004, ¶174. Stroud’s Ethernet communications use “Ethernet packets.” Ex-1004, ¶¶345-357, ¶181, ¶236, ¶242, ¶121, ¶119. “[C]ommunication” through Stroud’s Ethernet switches (Ex-1004, ¶174) comprises messaging-based communication because it

⁴ A POSITA looking to implement a scalable network simulation/testing platform would have been motivated to use Ethernet given its prevalence and performance. Ex-1004, ¶322, ¶¶495-503, ¶279; Ex-1003, ¶110.

includes Ethernet messages (e.g., packets/frames) exchanged between components. Ethernet used packetized messages, called frames. A POSITA would have understood this because Ethernet had been the most common wired networking protocol since the 1990s and was taught and used in universities worldwide. The management switch facilitates the communication of these packets. Ex-1004, ¶¶174-176 (“Thus, management switch 110 may essentially act as a router...”), ¶314. Stroud’s cards used Ethernet to communicate with destinations internal and external to the chassis. Ex-1004, ¶174, ¶¶113-115, ¶365, ¶¶413-426, Figs. 3, 12, 31; Ex-1003, ¶¶110-111.

Additionally, Stroud teaches IP packets as messages. Ethernet was a layer 2 protocol that supported IP packets for layer 3, which Stroud teaches when referencing the “packet’s layer 2/3/4 headers.” Ex-1004, ¶¶124-125, ¶164 (CLD. For example, Stroud teaches that the CLDs generating “layer2/layer 3 traffic”). A POSITA would have understood this because Ethernet packets were ubiquitous by the 1990s. *Supra* §I.B; Ex-1003, ¶110; Ex-1003, ¶112.

Stroud teaches the **packets each having a header**, including a “header for Ethernet packets.” Ex-1004, ¶349 ¶124.

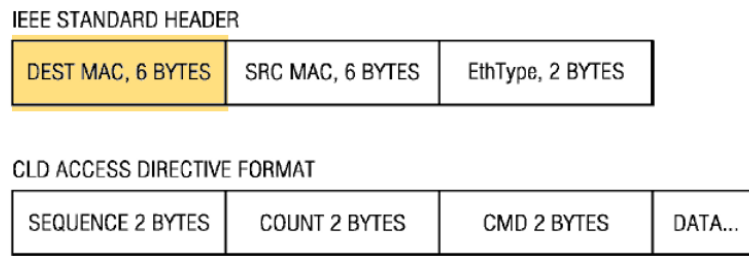


FIG. 17

Stroud further teaches distinct sub-fields within the destination MAC address of the Ethernet header. Ex-1004, ¶¶366-372. These sub-fields hold various identifiers, including those recited in [1e], explained sequentially below with chassis ID at the end. Ex-1003, ¶¶113-115.

Board identifier. Stroud teaches a “**board slot ID**” as a “[s]ub-field[] within the destination MAC address” of the packet header. Ex-1004, ¶370, ¶347. This distinct sub-field identifies a particular board, which resides in a particular slot in a chassis. Ex-1004, ¶98, ¶344.

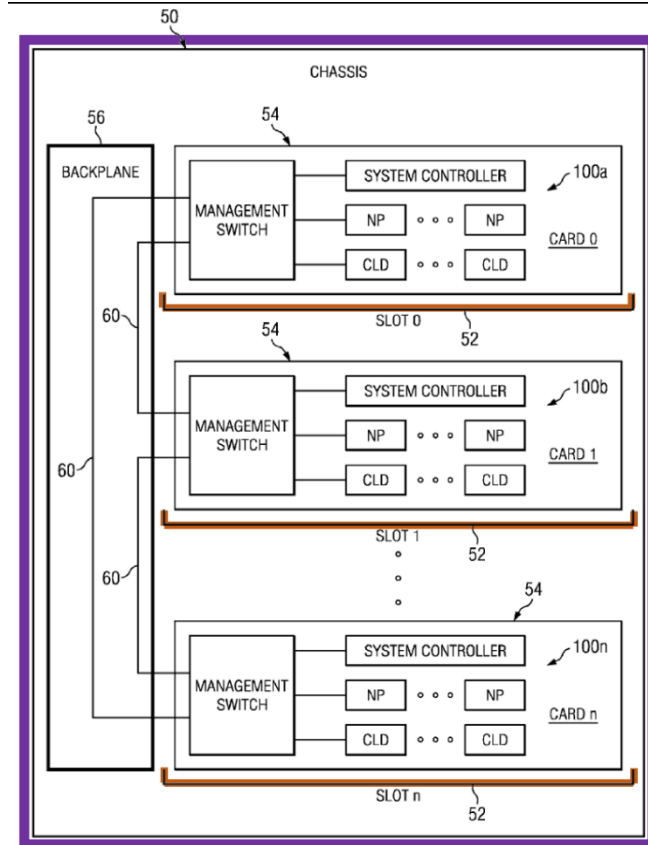


FIG. 3

To ensure that a packet reaches one of the slots within the chassis and to identify the correct board, Stroud teaches using the “board slot ID” to route packets to the correct slot and the board in that slot, identified by the board slot ID. Ex-1004, ¶370, ¶381.⁵ Ex-1003, ¶116.

Module identifier. Stroud teaches a sub-field for an “interface” identifier that identifies modules by their interface to the management switch. Ex-1004,

⁵ The ’753 patent describes a similar example for routing packets to the correct “pod or card in the chassis...” Ex-1001, 3:4-8.

¶372. Stroud’s card includes a number of controller, NP, and CLD modules; each is connected to the management switch. *Id.*, ¶101, ¶314, ¶103, ¶113, ¶174, ¶176, ¶189, Figs. 3-5. As Stroud explains, “each interface to the management switch has its own MAC address” because the value for the field in the 6th byte of the MAC address is unique to the module within each card. Ex-1004, ¶372. Therefore, the interface identifier in the 6th byte identifies each module on a card (e.g., each controller, NP, or CLD⁶) by its interface to the management switch. *Id.*, ¶100, ¶372.

⁶ Stroud’s interface identifier is assigned to the controller, NP, and CLD modules. Each interface has its own unique identifier. Ex-1004, ¶372. While Stroud refers to the “processor” interface (*id.*), Stroud’s teachings are not limited to just the network processors because controllers, NPs, and CLDs all have an interface to the management switch (*id.*, Figs. 4-5, 7A-8A), which uses the unique interface identifier—and therefore unique MAC address—to route packets to them. Ex-1004, ¶366, ¶354.

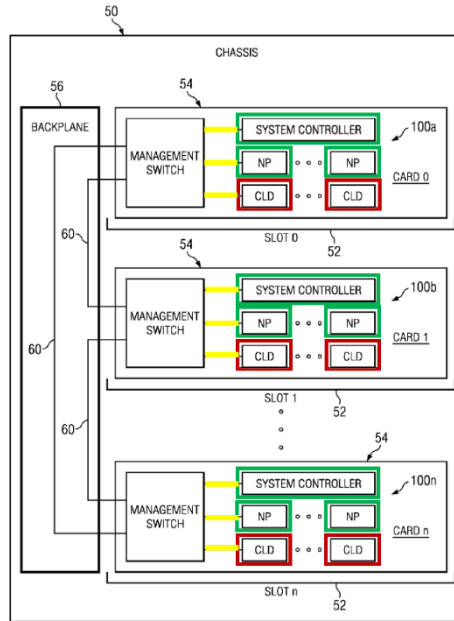
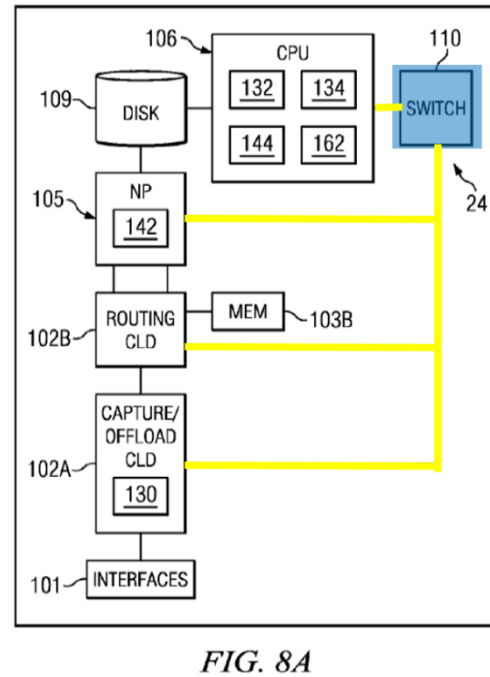
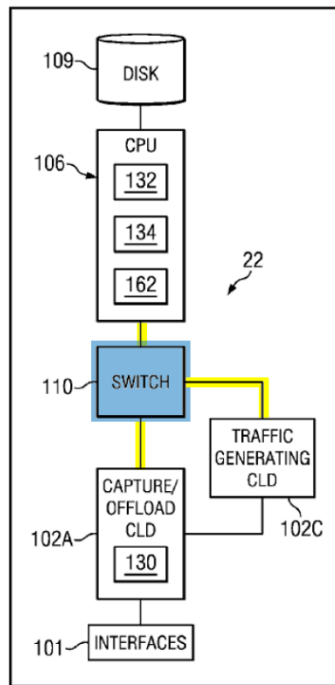
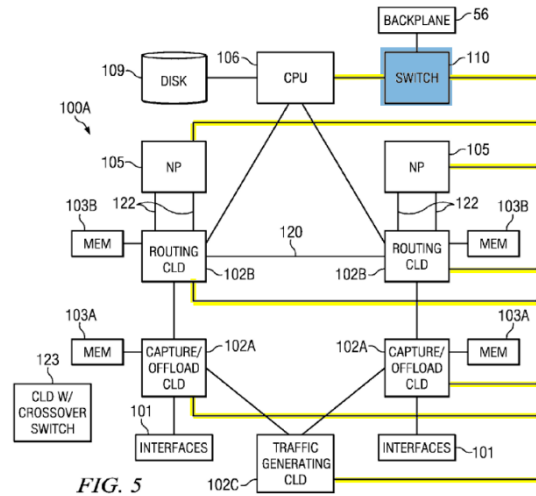
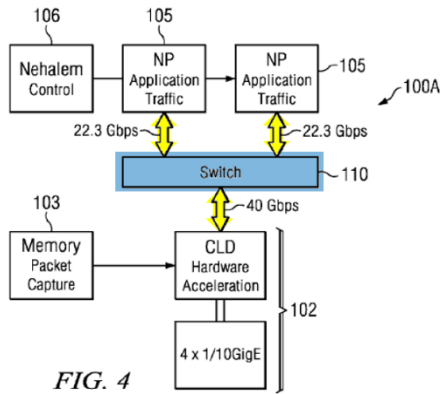


FIG. 3

Ex-1003, ¶¶117-118.

A module's interface to the management switch sits at the boundary between the module and the connection to the switch. Additionally, Stroud teaches each NP/CLD having a dedicated connection to the management switch. Ex-1004, ¶¶174-176, ¶134, ¶¶146-147, ¶¶156-157, ¶167, ¶189, ¶310, ¶¶314-319, Figs. 4-5, 6A-9A.



Ex-1003, ¶119.

Instance identifier. Stroud teaches a sub-field for a “processor or CLD number” that is used “to *distinguish between multiple instances* of the same type of processor or CLD on the same card.” Ex-1004, ¶371. When there are multiple instances of the same type of NP/CLD on a card, the processor or CLD number

identifies the unique instance of the CLD or processor on that card.⁷ Ex-1003, ¶120.

Type identifier. Stroud teaches a sub-field for a “chip type: e.g., processor, CLD, or other type of device.” Ex-1004, ¶368; Ex-1003, ¶121.

Stroud teaches the packet-header identifiers functioning **so that each type of module, each instance of a type of module, and each module on each board in each chassis can be addressed through the header.** Ground 1 teaches a packet header with chassis, board, module, instance, and type identifiers stored as distinct subfields within the destination MAC address. Ex-1004, ¶¶366-372. Each card’s management switch routes Ethernet packets to each module on the board using the destination MAC address. Ex-1004, ¶174, ¶354, ¶113. Therefore, each type of module (e.g., processor or CLD), each instance, and each module on each board can be addressed through the destination MAC address of the Ethernet header, using the respective sub-fields. Ex-1004, ¶¶365-373, ¶¶347-349; Ex-1005, 5:13-18, 11:55-12:14, Fig. 7. Ethernet switches route packets to their destination using the destination MAC address. Ex-1004, ¶354. A POSITA would have understood this basic networking concept. Ex-1003, ¶122.

⁷ The ’753 patent describes a similar example. Ex-1001, 8:61-65.

Chassis identifier. The combination with DaSilva teaches a “chassis ID.”

Ex-1005, 9:58-10:14 (“The hardware device information includes, for example, *chassis ID*....”), 5:13-18, Fig. 6 (chassis ID in Destination HDI):

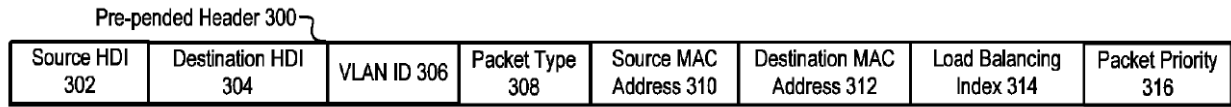


Fig. 6

The chassis identifier is a distinct sub-field that identifies a particular chassis.

DaSilva presents an example with three chassis, assigned “Chassis ID” 1, 2, and 3.

Ex-1005, 7:58-64.

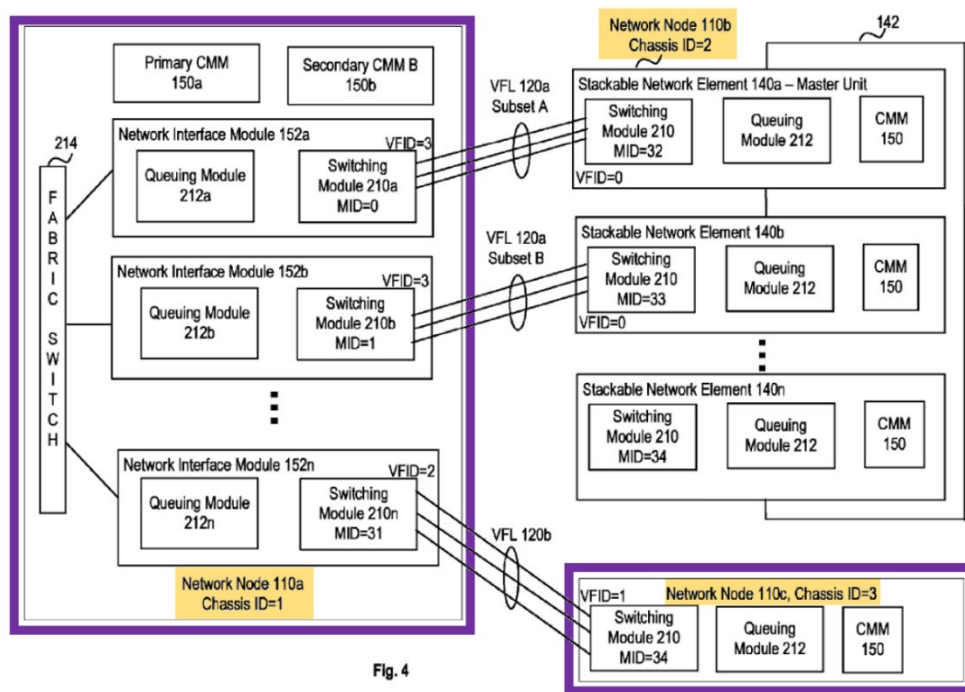


Fig. 4

Ex-1003, ¶123.

DaSilva teaches 2 ways of adding the chassis ID to a packet: “add[ing] a pre-pended header (or otherwise modif[y]ing the packet header) to include

hardware device information (HDI)” that includes “chassis ID.” Ex-1005, 9:58-10:14. With the first option, the chassis ID is pre-pended to the beginning of the packet header. *Id.*, 5:13-18 (“network nodes 110 exchange packets with *pre-pended headers*...that include...source *chassis ID*...”). For the second option, Stroud teaches a mechanism to modify an existing packet header to include identifiers in sub-fields of the destination MAC address. Ex-1004, ¶¶366-373. The combination simply adds another sub-field for chassis ID. For example, the second byte of the MAC address includes the type sub-field. Ex-1004, ¶368. Since Stroud only has three identified chip types (“processor, CLD or other”), two bits would be sufficient, leaving six bits for a chassis ID sub-field. *See id.* Having 64 chassis (represented by six bits) would allow for over 4 million addressable processing elements. Ex-1003, ¶124.

DaSilva further explains that the “chassis ID for each network node 110 is unique.” Ex-1005, 9:1-5, 4:53-54. DaSilva teaches packets being routed to a particular chassis based on its unique chassis ID. Ex-1005, 11:55-12:14 (explaining steps for parsing and using HDI with chassis ID); Ex-1003, ¶¶125-126.

Motivation to Combine. Stroud expands capacity by adding “modular card[s], or blade[s]” to a chassis. Ex-1004, ¶98, ¶176. When all slots are full, it would have been obvious to add more chassis as the most straightforward option of a finite number of ways to expand capacity further. Stroud teaches this type of

expansion, suggesting that various implementations can be “distributed across multiple [chassis] devices of 16.” Ex-1004, ¶343, Fig. 2E. It was common to use multiple chassis in data centers. Ex-1040, ¶2; Ex-1014, 000014-18, 000023. A POSITA looking to scale Stroud’s architecture to serve a larger network load would increase the number of chassis. Ex-1003, ¶¶127-129.

DaSilva provides teachings for multi-chassis architectures. When multiple chassis exist on a network, DaSilva uses a “chassis ID” to distinguish them and facilitate communication between them. Ex-1005, 7:58-64, 9:1-6. Therefore, DaSilva provides teachings for scaling a multi-card, chassis-based architecture such as the one taught by Stroud. Ex-1005, 7:51-8:22, 9:1-6. A POSITA would have been motivated to apply DaSilva’s teachings to Stroud. Ex-1003, ¶130.

A POSITA looking to add capacity to Stroud would naturally have looked to DaSilva. The references teach similar architectures for similar purposes. Stroud features a multi-card chassis for network testing (Ex-1004, ¶81), where each card has a management switch connected to a backplane (*id.*, ¶101, ¶99). Similarly, DaSilva features a multi-card chassis for routing traffic over network topologies, where each card has a switching module connected to a backplane. Ex-1005, 7:58-67. The references share similar goals—providing reliable and flexible networking and data processing infrastructure. Ex-1004, ¶2; Ex-1005, 1:55-59; Ex-1003, ¶133.

The references provide motivation. Stroud contemplates “creat[ing] larger systems” (Ex-1004, ¶115) and suggests modify its embodiments. Ex-1004, ¶746. Likewise, DaSilva teaches that “[w]hile particular combinations of various functions and features of the present invention are expressly described herein, other combinations of these features and functions are likewise possible.” Ex-1005, 25:37-42; Ex-1003, ¶132.

DaSilva provides express motivation to add HDI including chassis ID to existing packet formats by prepending or otherwise modifying the packet header. Ex-1005, 9:58-10:14. For the first (prepended) option, variations on Ethernet frames were known; it would have been trivial to add a layer of processing for DaSilva’s prepended header. Indeed, Stroud already teaches adding a “prepend header” for routing and segmentation. Ex-1004, ¶¶530-535, ¶¶405-426. For the second option, Stroud teaches a mechanism to modify packet headers with sub-fields within the destination MAC address. Ex-1004, ¶¶365-372. The combination would therefore simply have added one more sub-field to hold the chassis ID, explained above. *Id.*, ¶347, ¶¶364-372; Ex-1003, ¶124, ¶131.

Moreover, since Stroud’s cards were already connected using Ethernet (Ex-1004, ¶113, ¶176, ¶377), adding additional chassis would have simply used Ethernet connections on the same network. Ex-1004, ¶321 (citing “scalability” of

“utilizing Ethernet as a common internal communication channel”), ¶175; Ex-1003, ¶129.

The combination would have entailed minor additions to Stroud that would not have changed the principle of operation for either reference. Stroud taught a networked environment with multiple devices, with a chassis that used Ethernet. Ex-1004, ¶116, ¶129, ¶422, ¶474, ¶¶174-176. The combination simply adds additional chassis to the network using Ethernet. Additionally, Stroud already included several sub-fields for identifiers; adding one more would have been consistent with Stroud’s usage. Ex-1004, ¶¶366-372. DaSilva taught a chassis ID for identifying multi-card chassis—precisely how Ground 1 uses DaSilva’s teachings. *See* Ex-1005, 9:58-10:14, 11:55-12:14, 4:53-5:18, 7:58-64, Figs. 4, 7; Ex-1003, ¶134.

Moreover, the combination would not have adversely affected the compatibility of Stroud or DaSilva with other Ethernet equipment. Stroud teaches local MAC addresses to facilitate the use of sub-fields, and Stroud teaches a technique for encapsulating a packet, e.g., those with a prepend header or sub-fields, inside another Ethernet frame (Ex-1004, ¶426), allowing modified packets/frames to be processed by standard routers/switches. Ex-1003, ¶136.

Additionally, while the combination does not necessarily rely on virtualization, both references teach it; therefore the inclusion of virtualization—

though not needed for Ground 1 or a chassis ID—would not have precluded the combination either. Stroud teaches implementing virtual infrastructure (Ex-1004, ¶84) and using virtual machines (Ex-1004, ¶89). DaSilva teaches a “virtual chassis system” that allowed multiple physical chassis to be grouped together. Ex-1005, 3:64-4:7, 4:23-38. This was compatible because Stroud already contemplated implementing the chassis architecture in a virtual environment. Ex-1003, ¶135.

A POSITA would have had a reasonable expectation of success in combining Stroud and DaSilva’s teachings. The combination was simple, as explained above, merely adding chassis and a chassis ID. This required minimal modification—a prepended header or additional sub-field—with minor changes following the direct teachings of DaSilva and Stroud. Ex-1005, 9:58-10:14; Ex-1004, ¶¶366-372. The combination would not have changed the principle of operation and thus would have yielded the predictable result of a packet header including a chassis ID, and the trivial addition of network devices on a network. Adding more cards, either in the same chassis or over Ethernet to another chassis, would have been entirely consistent with Stroud’s modular, parallel architecture. *See* Ex-1004, ¶¶98-99, ¶102, ¶86; Ex-1003, ¶124, ¶¶137-138.

At bottom, the combination of Ground 1 deals with basic networking concepts for Ethernet/packets, taught in undergraduate courses and labwork. A POSITA with experience in data streaming and processing and the requisite

education would have had more than enough skill to combine Stroud and DaSilva.

Supra §III; Ex-1003, ¶139.

4. Claim 2

2. The platform of claim 1, further comprising: at least a subset of the one or more swappable pods or cards having an option for direct connection to one or more pods, cards or components, bypassing the module messaging interface network.

Stroud teaches two options for direct connection that bypass the Ethernet backplane network: I2C or serial bus (MLVDS) connection. Stroud's cards have "an "inter-card serial bus, and an...inter-card serial port access system." Ex-1004, Abs., ¶¶6-8, ¶320. The cards establish direct serial connections over these two interfaces to communicate with each other. *Id.*, ¶671 ("direct serial access [to] each other processor in system 16"), ¶665 ("direct serial access to any processor on any card 54 in system 16"):

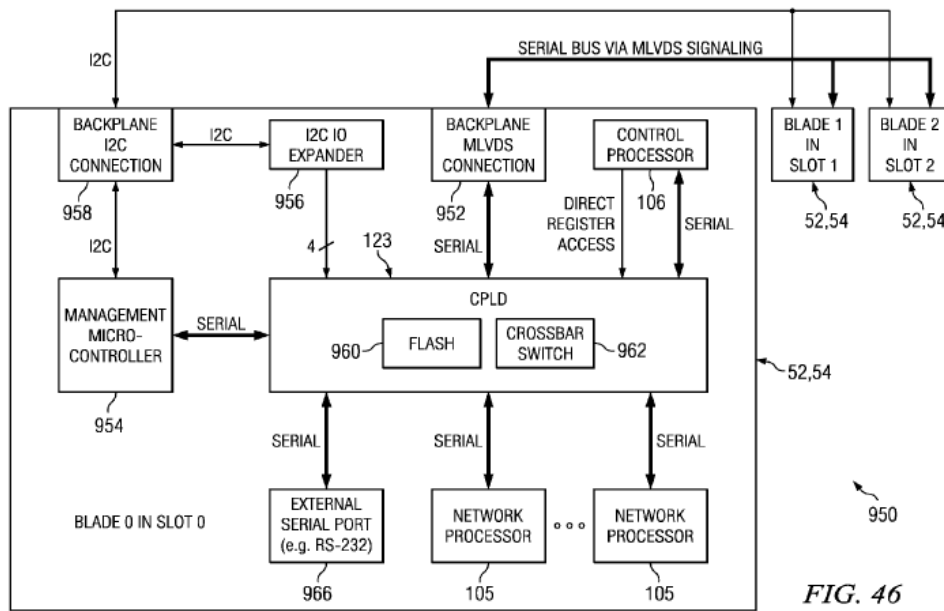
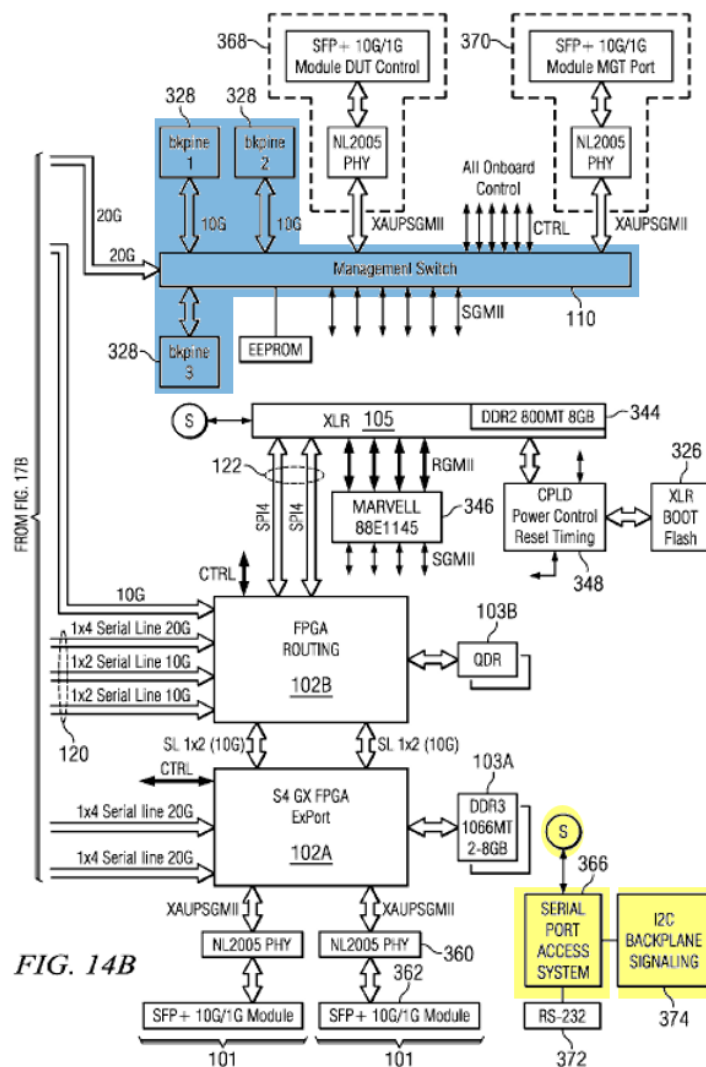


FIG. 46

Ex-1003, ¶¶141-142.

Stroud's I2C and serial bus connections pass through separate connections in the backplane, distinct from the Ethernet network, which includes "physical connections for connecting each card 54 directly to each other card 54." Ex-1004, ¶101. The cards choose which other card to connect to using an "I2C backplane signaling system 374" that "manag[es] inter-card serial connections." Ex-1004, ¶320, ¶¶669-671. The I2C connections and the backplane serial bus connections (yellow below) are separate from the Ethernet-based messaging interface network that goes through management switch (blue below). Ex-1004, ¶320.



Ex-1003, ¶¶143-144.

Additionally, Stroud teaches this limitation under PO's apparent interpretation of the claims in district court. PO alleged front-panel Ethernet connections satisfy direct connection of claim 2. Ex-1041, 000007-8; Ex-1003, ¶145. Stroud teaches front-panel Ethernet interfaces (e.g., Ethernet "test interfaces 101"). Ex-1004, ¶116, ¶¶714-717, Figs. 5-9A, 14A-14B, 31, 54-59B:

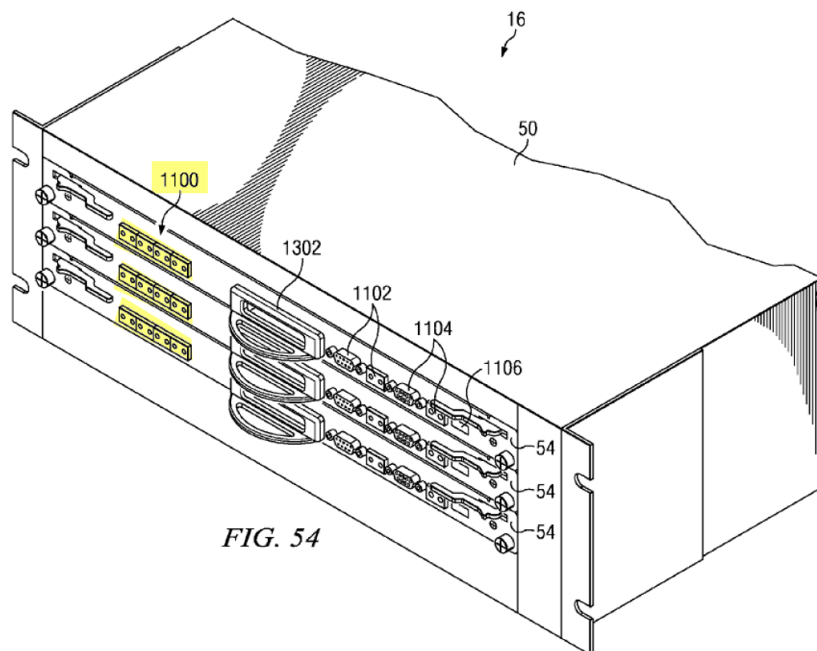


FIG. 54

Ex-1003, ¶¶145-146.

The test interfaces provide an option for the cards to directly connect with other cards or other components, including “a system 18 to be tested.” Ex-1004, ¶¶114-116. The interfaces deliver packets to/from the test system. Ex-1004, ¶¶121-125, ¶¶128-132, ¶145; *infra* §V.B.2[7c] (discussion of data flow between modules); Ex-1003, ¶147.

5. Claim 3

3. **The platform of claim 1, wherein the one or more swappable pods or cards comprises at least one processor card having a at least one processor allocatable for offloading hardware tasks to software, orchestrating data flows, orchestrating configurations of hardware modules, and executing in cooperation with a sandbox of one of the plurality of swappable pods or cards.**

Stroud teaches each card is a **processor card having at least one processor**, for example, controller 106 which comprises a “management processor.” Ex-1004, ¶154, ¶234, Fig. 6A. The controller runs software to control and configure the CLDs and NPs. Ex-1004, ¶187. The controller executes a number of software modules, including “a CLD server” software module that handles CLD control packets. Ex-1004, ¶356, ¶¶155-163; Ex-1003, ¶¶149-150.

The controller is allocatable for: **offloading hardware tasks to software** because the platform practices “[i]ncoming packet routing and load balancing” to distribute packets among NPs and CLDs (FPGAs). Ex-1004, ¶¶245-248, ¶321, Fig. 15. Therefore, when were CLDs near capacity, it was obvious to offload hardware tasks to a NP, including the controller which acts “as an auxiliary network processor.” Ex-1004, ¶154. It was common to offload complicated tasks from hardware to avoid clogging the pipeline. Ex-1003, ¶151. Stroud suggests workloads are portable between NPs/CLDs with exemplary details of dynamic routing. Ex-1004, ¶296, ¶¶288-294 ¶¶406-412, Fig. 24; Ex-1003, ¶151.

Orchestrating data flows. The “controller...initiate[s] and coordinate[s]...the functions of network testing system.” Ex-1004, ¶104. For example, the controller directs the flow of control packets to their targets (Ex-1004, ¶357, Fig. 19); receives traffic-generating requests from NPs, determines a CLD to execute the command, and modifies the command packet for the target CLD. Ex-1004, ¶¶357-359. The controller monitors such directive packets and follows up appropriately, thus orchestrating the command and control packets that control data flows. Ex-1004, ¶360-362; Ex-1003, ¶152.

Moreover, the controller orchestrates data flows by directing the activities of the NPs and CLDs, including distributing work to them. Ex-1004, ¶321. The controller controls when to “start, stop, and post-process packet captures.” Ex-1004, ¶187. The controller configures/instructs the NPs and CLDs to implement the simulation as discussed above (Ex-1004, ¶234; *supra* §V.A.3[1b]) and immediately below. Ex-1003, ¶153.

Orchestrating configurations of hardware modules. The “controller...initiate[s] and coordinate[s]...[the] network testing system” (Ex-1004, ¶104), including “connect[ing] to the management switch 110 to configure, boot, and manage all...CLDs” including for traffic generation. Ex-1004, ¶¶156-158, ¶167. The CLD modules are hardware modules. *Supra* §V.A.3[1b]. Stroud

illustrates the process of the controller configuring CLDs during a boot up sequence. Ex-1004, ¶64, ¶¶691-696, Fig. 52; Ex-1003, ¶¶154-156.

Also, the controller also has “high-level responsibilities” that orchestrate what the CLDs are configured to do. Ex-1004, ¶159. For example, the controller applies user settings for the test platform and coordinates its operation. Ex-1004, ¶¶160-163. Ex-1003, ¶157.

Executing in cooperation with a sandbox (e.g., in a CLD) of one of the plurality of swappable pods or cards. The controller “coordinates the rest of the system operation,” cooperating with the NPs and CLDs. Ex-1004, ¶161.

Connections to the CLDs are “used both for inter-processor communication and for sending traffic to and from the test interfaces.” Ex-1004, ¶132. This allows the controller to execute in cooperation with the router CLD. Ex-1003, ¶152, ¶158.

Additionally, the controller executes “as an auxiliary network processor.” Ex-1004, ¶154. Stroud teaches several examples of NPs executing in cooperation with CLDs. *E.g.*, Ex-1004, ¶¶122-123, ¶¶128, ¶131, ¶135, ¶224, ¶236, ¶245.

Therefore, when the controller functions as an NP, it executes in cooperation with CLDs. *See id.*; Ex-1003, ¶159.

Stroud intended the controller, NPs, and CLDs to work “cooperatively” to construct larger test systems of multiple cards. Ex-1004, ¶102. Various

subsystems, implemented across the controller, NPs, and CLDs, “operate simultaneously and cooperatively....” Ex-1004, ¶86, ¶177; Ex-1003, ¶160.

Stroud’s CLDs include sandboxes; they comprise PLDs/FPGAs (Ex-1004, ¶107, ¶¶110-112, ¶¶311-313) with portions for user-definable hardware, where the user programs and alters the connection pattern of the circuits, creating custom circuits for performing custom functions in hardware. Ex-1004, ¶107, ¶¶109-112, ¶117, ¶¶311-313; *supra* §V.A.3[1b]-[1c]. Therefore, the CLDs include sandbox portions designated for user-programmable electronic circuits.⁸ Stroud teaches the hardware modules configured with discrete functions Ex-1004, ¶462. Each CLD functions independently and its design is isolated from other CLDs. Ex-1003, ¶161.

6. Claim 13

[13pre] A tangible, non-transitory, computer-readable media...

Stroud teaches the controller is connected to a memory device that stores instructions for controlling the card’s operation. Ex-1004, ¶¶169-170 (“Solid State Disk Drive 109”). This was a tangible, non-transitory, computer-readable media with instructions that, when executed by the controller, configure and operate the

⁸ PO proposed a construction of “sandbox” as “portion designated for programmable electronic circuits.” Ex-1036, 000004.

hardware and software modules in Stroud's testing platform. Ex-1004, ¶158, ¶¶171-173. The instructions cause the controller to perform a method that comprises the subsequent limitations, discussed below. Ex-1003, ¶¶164-167.

[13a] configuring a plurality of swappable pods or cards in one or more chassis, coupled through a messaging interface network, to implement a data flow processing architecture;

[13a] recites overlapping language with [1a] and [1d] (Ex-1003, ¶168 (claim comparison)) that is satisfied for the same reasons discussed above. *Supra* §§V.A.3[1a], [1d]. Stroud teaches configuring cards to implement a data-flow processing architecture by configuring the plurality of hardware modules (e.g., CLDs) and software modules (e.g., controller and NPs). Ex-1004, ¶¶80-81, ¶¶63-64, ¶¶689-700, ¶¶106-107, ¶158, ¶¶170-173, Figs. 51-52; *supra* §V.A.3[1c]; Ex-1003, ¶¶168-170.

[13b] configuring user-definable hardware modules or user definable software modules in each of one or more of the plurality of swappable pods or cards;

[13b] recites overlapping language with [1c]-[1d] (Ex-1003, ¶171) that is satisfied for the same reasons. *Supra* §§V.A.3[1c]-[1d]; Ex-1003, ¶¶171-174.

The previous analysis explained how Stroud teaches **each of** the cards including user-definable hardware (e.g., CLD) modules and user-definable software (e.g., controller or NP) modules, configured by users to implement data-flow processing architectures. *Supra* §§V.A.3[1c]-[1d]; Ex-1004, ¶¶63-64, ¶¶689-

700, ¶¶106-107, Figs. 51-52, Ex-1004, ¶158, ¶¶170-173. “Each architecture” of “[e]ach card” provides configurable hardware/software modules. Ex-1004, ¶¶99-102, ¶¶80-81; Ex-1003, ¶¶172-173.

[13c] communicating a data flow in accordance with the implemented data flow processing architecture among one or more hardware modules or one or more software modules in each of the plurality of swappable pods or cards, including the user-definable hardware modules or user definable software modules; and
--

Stroud teaches **communicating a data flow in accordance with the implemented data flow processing architecture**, by, e.g., teaching a network testing application for generating “synthetic data flows,” with numerous examples, e.g., “HTTP Authenticated” and “HTTP Flash Video.” Ex-1004, ¶298, Fig. 13D. The network testing includes communicating the data flow in accordance with the implemented data flow processing architecture because that is one of the purposes of the network testing platform: to simulate the network traffic by generating and then sending a data flow (e.g., HTTP, video, and other network data flows) using the parallel architecture of the platform. *Id*; Ex-1003, ¶175.

Additionally Stroud’s network testing platform communicates the data flow (e.g., the simulated network traffic) as it would pass through a simulated network, based on the data flow processing architecture implemented in the platform (e.g., network traffic processing for file servers, web servers, mail servers, and database servers in data centers; firewalls; LTE networks; etc.). Ex-1004, ¶¶80-93. Some

simple applications for testing include: an “application server...with multiple available services (e.g., mail, Web, SQL, and file sharing)” and a network “switch.” Ex-1004, ¶¶96-97, Figs. 2F-2G:

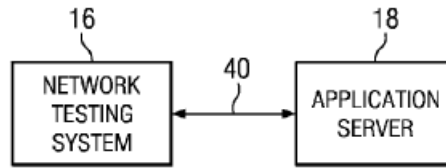


FIG. 2F

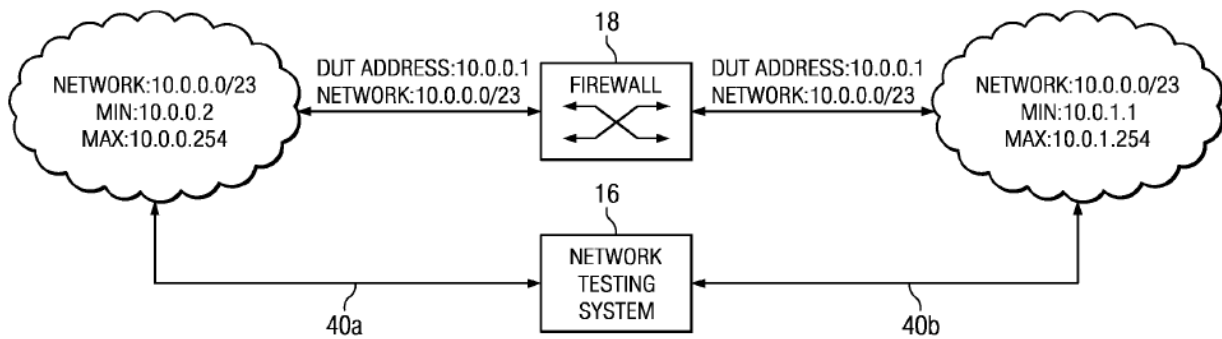
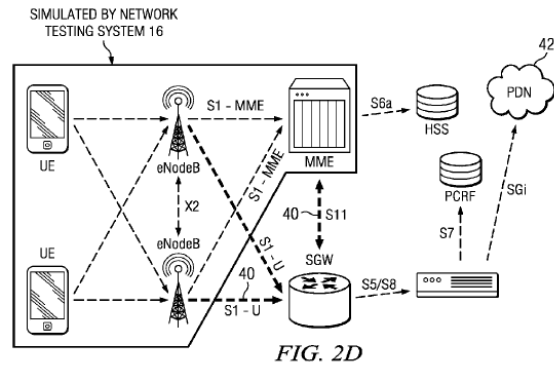
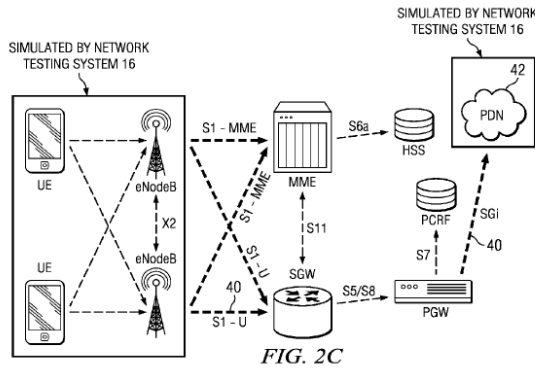


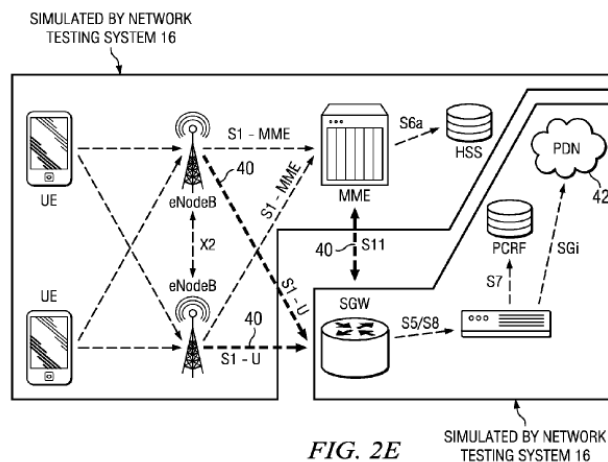
FIG. 2G

Ex-1003, ¶176.

Figures 2C through 2E illustrate examples of more complex architectures where hardware and software modules are configured to “simulate[] various components of an LTE network.” Ex-1004, ¶¶91-92. This includes simulating “user equipment (UEs),” “eNodeB interfaces,” and “MME”. Ex-1004, ¶¶93-94, Figs. 2C-2D:



The architecture of Figure 2E “simulate[s] all components of the LTE network” in order to test the performance of devices monitoring the data flows across the network segments. Ex-1004, ¶95, Fig. 2E:

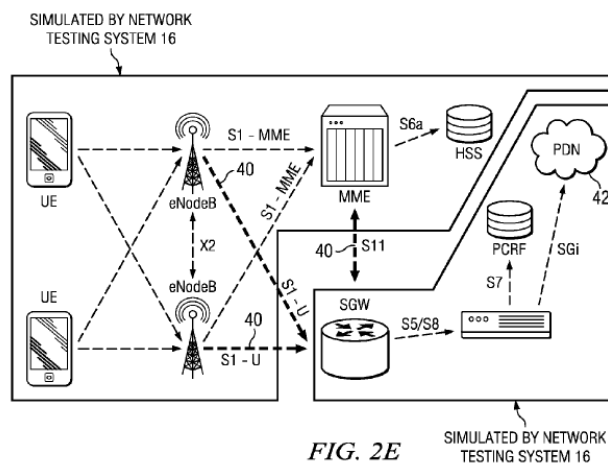


In all these scenarios, data flows are generated and communicated in accordance with the network topology of the simulated network architecture, implemented on the data flow processing architecture of Stroud’s platform. Ex-1003, ¶¶177-178.

Stroud teaches these communications and architectures are **among one or more hardware modules or one or more software modules in each of the plurality of swappable pods or cards**, by, e.g., “distribut[ing] workloads” across

hardware and software modules within a card. Ex-1004, ¶¶321. Packets are communicated among different nodes simulating devices on the network to test and measure the processing. Ex-1004, ¶¶80-81; Ex-1003, ¶179.

Stroud further teaches carrying out such communication across modules between different cards, because it is designed to provide scalable data processing capacity. Ex-1004, ¶321 (“scalability of the FPGA pipelining and interconnected FPGA/network processor architecture utilizing Ethernet as a common internal communication channel”). Communication for the distributing processing and testing architectures occur across cards in multiple chassis. Ex-1004, ¶343 (“The hardware devices...may be...distributed across multiple devices of 16, as appropriate.”), Fig. 2E (S1-U and S1-11 between two chassis simulating different LTE network segments):



Ex-1003, ¶180.

Stroud explains several data flow processing applications and provides exemplary data flow processing architectures using the modules on the blades/cards. Ex-1004, ¶178, ¶114, Figs. 5-9B. For example, Figure 6A depicts an architecture for a “high-speed, high-resolution network packet capture subsystem,” using the controller, capture/offload CLD, and memory. Ex-1004, ¶15, ¶¶182-190, Fig. 6A. Stroud explains how data flows of network packets are captured, stored in memory, and communicated to the controller. Ex-1004, ¶¶191-196, ¶¶201-218 (“high-speed packet generation and measurement subsystem”, ¶¶220-250 (“application-level simulation and measurement subsystem”), ¶¶255-266 (“security and exploit simulation and analysis subsystem”), Figs. 6B-9B; Ex-1003, ¶181.

For these various data flow processing architectures, Stroud teaches that data flows are communicated among the hardware and software modules on the cards. As discussed above, the modules communicate among modules across different cards via the backplane Ethernet interface network. Ex-1004, ¶99, ¶¶113-115, ¶176, ¶377, ¶¶669-670, Figs. 3-5, 48; *supra* §V.A.3[1a]. Additionally, Stroud teaches data flows communicated directly between cards using the backplane serial bus. *Supra* §V.A.4; Ex-1003, ¶182.

The implemented data flow processing architecture is also among the modules on the blades/cards, as shown in the aforementioned examples and figures. For example, the relationship between the modules are shown in the

various architectures depicted in Figures 5 through 9B. These architectures function as intended because the modules are configured with the architecture defined among the hardware and software modules, thus Stroud teaches data flow processing architectures that are among the modules. As discussed above, Stroud's network testing platform has an architecture that is implemented among various hardware modules (e.g., routing CLD, capture/offload CLD, and through traffic generating CLD) and software modules (controller and NPs) that generate, capture, and process data flows per the architecture defined in the network testing system. *Supra* §§V.A.3[1pre]-[1b], §V.A.3[1d]; Ex-1003, ¶183.

Stroud teaches the modules on the blades/cards **includ[e] the user-definable hardware modules or user-definable software modules**. As discussed above, all the modules are user-definable hardware modules or user-definable software modules. *Supra* §V.A.3[1c]; Ex-1003, ¶184.

[13d] communicating the data flow through the messaging interface network...
--

As discussed above, Stroud teaches **communicating the data flow** among the modules on the cards, which passes **through the messaging interface network** (e.g., Ethernet backplane interface network). *Supra* §V.A.3[13c], §V.A.3[1a]. The remainder of [13d] recites overlapping language with [1e] (Ex-

1003, ¶186 (claim comparison)) that is satisfied for the same reasons. *Supra* §V.A.3[1e]; Ex-1003, ¶¶186-187.

7. Claim 14

Claim 14 recites overlapping language with claim 3 (Ex-1003, ¶189 (claim comparison)) that is satisfied for the same reasons. *Supra* §V.A.5. Stroud teaches allocating a processor (controller 106) to all four tasks recited in claim 3. Ex-1004, ¶¶356-359, ¶154, ¶346, Fig. 6A; Ex-1003, ¶¶189-190.

B. Ground 2

Each of Stroud's cards includes multiple types of memory, e.g., DRAM, SRAM, and disk. Cragon explains textbook approaches for using different types of memory in computers. Ground 2 applies these common memory arrangements to Stroud's cards, allowing a flexible and selectable arrangement. The motivation to combine is explained below. *Infra* §V.B.2[7d] (motivation subsection); Ex-1003, ¶¶192-193.

1. “Memory Systems and Pipelined Processors” (“Cragon”)

Cragon is §102(a)(1) prior art published in 1996. Ex-1022, ¶¶9-12. Cragon was publicly available because it was shelved and indexed in public libraries, including Penn State University Library and the Library of Congress. *Id.*; Ex-1037, 000001, 0000007; Ex-1003, ¶¶194-196.

Cragon is a textbook that describes common ways to arrange different memory types (Ex-1006, 000048, 000071-73), including as multi-level cache (Ex-

1006, 000219-220), interleaved memory (Ex-1006, 000496-497), and other common modes of operation (e.g., 000255-265, 000413-415, 000536-548); Ex-1003, ¶197.

Cragon is analogous art in the same field (*supra* §III) and reasonably pertinent to the '753 patent because it relates to memory architectures that improve the performance of data-processing systems and provides solutions for configuring memory suitable for a data-processing architecture. Ex-1006, 000040; Ex-1003, ¶198.

2. Claim 7

[7pre]. A method for data flow processing in a platform, comprising:

[7pre] recites overlapping language with [1pre] (Ex-1003, ¶200) that is satisfied for the same reasons. *Supra* §V.A.3[1pre]; Ex-1003, ¶¶200-202.

Stroud further applies its teachings to **a method** in the platform discussed for [1pre]. Stroud “relates to...*methods* for testing communications networks” (Ex-1004, ¶1) with a modular platform (Ex-1004, ¶¶85-87) for data flow processing. *Supra* §V.A.3[1pre]. The method comprises steps discussed for [7a]-[7d]. Ex-1003, ¶201.

[7a] configuring...

[7a] recites overlapping language with [1a] and [1d] (Ex-1003, ¶203 (claim comparison)) that is satisfied for the same reasons. *Supra* §§V.A.3[1a], [1c]-[1d];

Ex-1003, ¶¶203-205. As explained above, Stroud teaches configuring cards to implement a data flow processing architecture by configuring hardware and software modules on each card. Ex-1004, ¶¶80-81, ¶¶63-64, ¶¶689-700, ¶¶106-107, ¶158, ¶¶170-173, Figs. 51-52; *supra* §V.A.3[1c]; Ex-1003, ¶204.

[7b] configuring...

[7b] recites overlapping language with [1b], [1c] and [1d] (Ex-1003, ¶206) that is satisfied for the same reasons. *Supra* §§V.A.3[1b]-[1d]. The previous analysis explained how Stroud teaches configuring the modules **in each of** the cards. Additionally, Stroud teaches “[e]ach architecture” of “[e]ach card” provides the configurable functionality of the platform. Ex-1004, ¶¶99-102, ¶¶80-81; Ex-1003, ¶¶206-209.

[7c] communicating a data flow...

[7c] recites overlapping language with [13c] (Ex-1003, ¶210) that is satisfied for the same reasons. *Supra* §V.A.6[13c]. Ex-1003, ¶¶210-211.

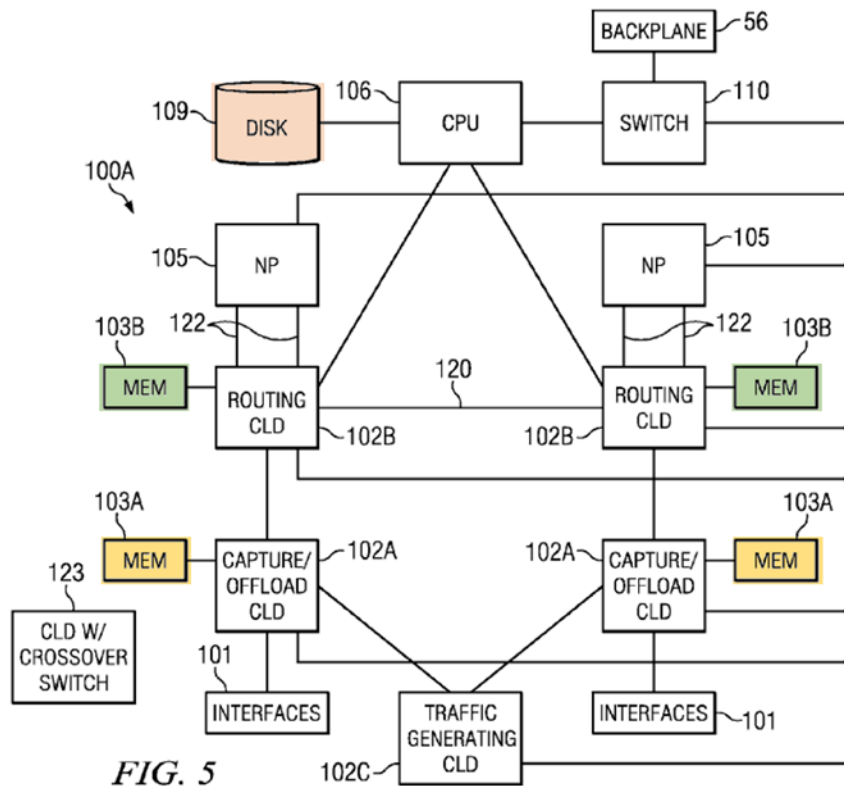
[7d] selecting a memory superposition in at least one of the plurality of swappable pods or cards having a plurality of types of memory with selectable arrangement of the plurality of types of memory as multilevel cache, interleaved memory, swapped memory over a specified address range, and memory allocated to specified address ranges according to parameterized task needs.

Selecting a memory superposition in at least one of the plurality of swappable pods or cards. Stroud teaches memory superposition (*supra* §IV.A)

with two or more types of memory on its cards (explained below) that are made available for data-flow processing because that is the functionality and purpose of Stroud's cards. *Supra* §V.A.3[1pre]; Ex-1003, ¶213.

Additionally, the memory on Stroud's cards is available for a data-processing operation—the general purpose of memory—because it is connected to processors, NPs, or CLDs configured to process data. *See* Ex-1004, ¶120, ¶127, ¶133, ¶143, ¶186, ¶¶190-198, ¶¶306-312, Figs. 6A-9A, 14A-B. Ground 2's teachings of selecting a memory superposition is discussed below. Ex-1003, ¶214.

At least one of the plurality of swappable pods or cards having a plurality of types of memory. Stroud teaches the cards having multiple types of memory, for example memory 103A “may comprise...DRAM, SRAM, or Flash memory, hard drive, or any other memory...” (Ex-1004, ¶127), and “103B may comprise any suitable memory device,” e.g., “QDR (quad data rate) SRAM” (*id.*, ¶169).



Ex-1003, ¶215.

Stroud's figures teach various card configurations with a plurality of memory types (e.g., DDR3 DIMMS, DDR2 DIMMS, and QDR SRAM). Ex-1004, ¶717, Figs. 6A-9A, 55A-59B, 14A-14B:

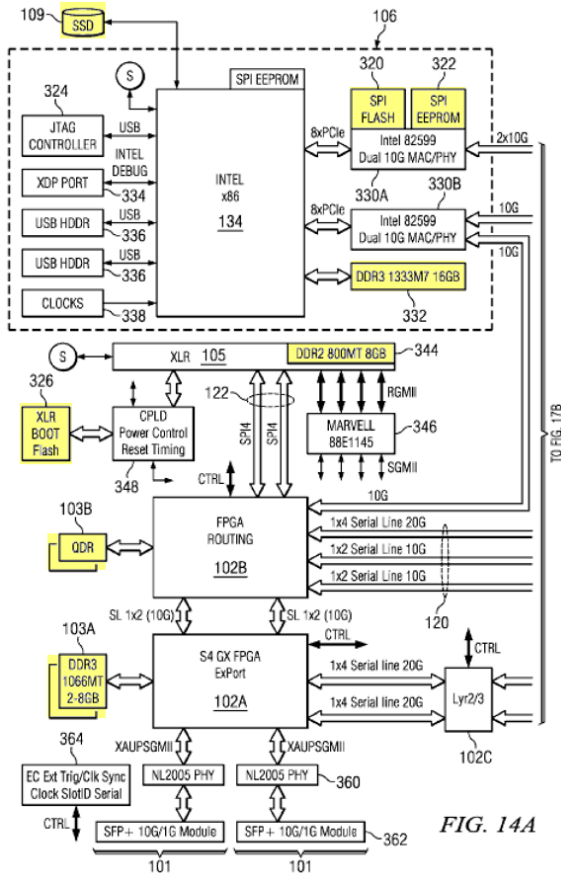


FIG. 14A

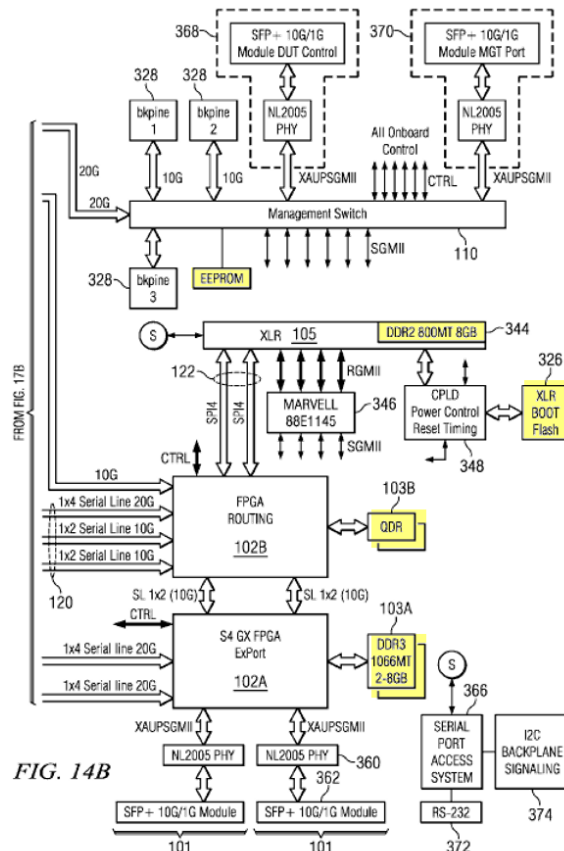


FIG. 14B

Ex-1003, ¶216.

A selectable arrangement of the plurality of types of memory. Cragon teaches textbook arrangements for different types of memory in computers, and selecting the arrangement of memory based on workload parameters. Ex-1006, 000413-414, 000536-548; Ex-1003, ¶217.

Applied to the memory on Stroud's cards, the combination therefore teaches a selectable arrangement of those memories depending on the needs of particular workloads. Since Stroud aims to provide a flexible platform with user-definable modules configured to perform a wide variety of applications (Ex-1004, ¶321,

¶¶106-113, ¶343, ¶373; *supra* §V.A.3[1b], §V.B.2[7b]), it was obvious to select between common memory arrangements to suit the particular application. Stroud's CLDs (e.g., FPGAs) include hardware circuitry the user can alter into custom functionality, including the ways in which the CLD uses and interacts with memory. Ex-1004, ¶¶106-112, ¶¶117-120, ¶¶311-313, ¶¶689-700, Figs. 51-52. Different functions have different memory requirements, and Cragon teaches several conventional memory arrangements for different purposes, e.g., depending on parameters. Ex-1006, 000536-548. Therefore, the combination teaches or suggests a selectable arrangement of the plurality of types of memory on Stroud's card based on the needs of the application running on the card, particularly in view of the flexible, reconfigurable nature of CLDs. *See id.*; Ex-1003, ¶218.

Ground 2 teaches the selectable arrangement including the arrangements listed in [7d], addressed in sequence below. Ex-1003, ¶¶219-233.

Multilevel cache. Cragon teaches multilevel cache with two levels. Ex-1006, 000219-220.



Figure 2.27
Two-level cache.

As Cragon explains, L1 cache comprises small, fast memory close to the processor core. Ex-1006, 000219. L1 was typically SRAM, which Stroud teaches. Ex-1004,

¶127, ¶169. L2 cache was larger and slower; therefore, it was obvious to use slower types of memory on Stroud's card. Ex-1004, ¶127. Multilevel cache was known in the art for accelerating access to recently-used data and was widely used for decades. *See* Ex-1006, 000067, 000073. Therefore, a POSITA would have found it obvious to use multiple types of memory (on-chip or on-card) in a multilevel cache. Ex-1003, ¶¶219-221.

Interleaved memory. Cragon teaches "high-order *interleaving*" in which different types of memory are alternately accessed to achieve an average access bandwidth. Ex-1006, 000496-497. Here, Cragon interleaves (alternates) access to different types of memory: as Cragon explains, "[h]igh-order interleaving...permits the easy *intermixing* of different memory technology such as a combination of RAM and ROM..." *Id.*, 000497. Applying Cragon's teachings to Stroud, it was obvious to arrange the multiple types of memory (on-chip or on-card) as high-order interleaved memory, because this was a textbook technique known since the 1990s for blending the characteristics (e.g., size and speed) of different memory types. *See id.* Additionally, high-order interleaving allowed "for easy memory expansion." Ex-1006, 000497; Ex-1003, ¶222.

Cragon teaches that "[i]nterleaved memories...distribut[e] the read or write accesses over a number of memory modules, which is a form of parallelism." Ex-1006, 000491. Therefore, alternately accessing different types of memory

arranged results in an access bandwidth that averages that of the different memory types. *See id.*; Ex-1003, ¶223 (Dr. Alpert’s explanation with numerical example). Therefore, when applying Cragon’s high-order interleaving to Stroud’s various on-chip and on-card memory, the resulting accesses achieve an average access bandwidth of those memories. Ex-1006, 000491-497; Ex-1003, ¶¶223-224.

To the extent PO argues a plain meaning of “interleaved memory,” Cragon literally teaches “[i]nterleaved memories.” Ex-1006, 000491-496, Fig. 5.1; Ex-1003, ¶225.

Swapped memory over a specified address range. Cragon teaches “swapping partitions in and out” over a specified address range, using “virtual memory.” Ex-1006, 000257, 000255, 000264-265, 000051. Data for certain address ranges are moved from main memory (e.g., DRAM) to non-volatile storage (e.g., disk) when not in use. Ex-1006, 000256. Cragon teaches “allocat[ing] its own address space” to a partition and swapping the location of the partitions from primary to secondary (e.g., non-volatile) memory and back. Ex-1006, 000257; Ex-1003, ¶226.

Cragon further teaches “paged allocation,” which splits the address space into pages and “mov[es] pages back and forth between real and auxiliary memory.” Ex-1006, 000265, 000262, 000352 (“page swap”). Thus each page is swapped in and out for a specific address range at fixed size intervals. Ex-1006, 00265 (“Page

is an allocation of fixed-length aggregations of addressable units.”), 000262 (e.g., “4 Kbytes”). Cragon thus teaches swapping memory with specified address ranges. Ex-1003, ¶227.

Applied to Stroud, the combination teaches a plurality of memory types (on-chip or on-card) as swapped memory over a specified address range, e.g., DRAM as main memory and disk as non-volatile storage for the swap file. *See* Ex-1004, ¶308, ¶195, ¶196, ¶269, Figs. 5-10, 14A-B, 20, 23, 30, 49. This aligns with the textbook arrangement for swapped memory. Alternatively, it would also have been obvious to use other memories on Stroud’s card for swapped memory, e.g., other DRAM or SRAM (Ex-1004, ¶127, ¶169), with disk as non-volatile storage. Ex-1006, 000255-257, 000264-265; Ex-1003, ¶¶228-229.

Memory allocated to specified address ranges according to parameterized task needs. Stroud teaches assigning specified address ranges to memory types based on their intended use (e.g., configuration memory, internal working memory, external memory). Ex-1004, ¶352. Cragon teaches memory allocated to specified address ranges in for each task, to avoid “interfer[ing] with another task.” Ex-1006, 000413. Cragon uses “bounds registers” that set the “upper address limit” and the “lower address limit” of contiguous memory allocated to a task. Ex-1006, 000413-414. As an example, Cragon explains how this task-based allocation is implemented paged memory systems, which as

explained above operates on specified address ranges. Ex-1006, 000414-417; Ex-1003, ¶¶230-233.

Additionally, Cragon teaches allocating memory to a task according to parameterized task needs, e.g., disk storage for interleaving or swapping, to meet “*requirements based upon workloads.*” Ex-1006, 000536. Cragon teaches five commonly-used parameters for task needs: “[c]apacity,” “[b]andwidth,” “[s]ervice rate,” “[r]esponse time,” and “cost.” Ex-1006, 000536. For each parameter, Cragon provides examples of different tasks and their differing requirements. Ex-1006, 000536-000548, 000041-63.

Level	Access Time	Access Time Ratio
Cache	50 ns	1
Main Memory	200 ns	4
Disk	20 ms	100,000

Id., 000054; Ex-1003, ¶232.

Motivation to Combine. Stroud seeks to provide a platform with scalable capacity and flexible modules for custom software and user-definable hardware (CLD) functions. Ex-1004, ¶382 (looking to “improve the scalability and reliability”), ¶321(discussing “scalability” of “interconnected FPGA/network processor architecture”), ¶399, ¶493, ¶128, ¶144. Therefore, a POSITA would have been motivated to provide flexible memory arrangements to support Stroud’s

flexible architecture and the variety of potential functions on Stroud's cards. Ex-1003, ¶234.

Cragon furthers Stroud's stated goals by teaching common memory arrangements, described earlier in this section, which were taught in undergraduate courses as part of memory hierarchies. Ex-1006, 000048. A POSITA would have been motivated to apply Cragon's memory arrangements for their known benefits, and to select between them to suit a particular application, as explained above. Ex-1003, ¶235.

This would have been a combination of prior art elements according to known methods to yield predictable results—namely, the use of Cragon's basic memory modes with Stroud's cards, which already include a flexible variety of memory types. Ex-1004, ¶127 (“any suitable type of memory device”), ¶169. Stroud arranged memory for particular card applications (Ex-1004, ¶717, Figs. 55A-55B) and sought to optimize memory performance (e.g., Ex-1004, ¶127). Moreover, the prior art provides motivation to improve Stroud's memory. Ex-1028, ¶2 (“Finding ways to improve system memory is therefore a motivation of computing system engineers”). Ex-1003, ¶¶236-237.

A POSITA looking to implement Stroud's network simulation platform with different memory types would naturally have looked for references with the details for using different types of memory together. Cragon teaches these details. Ex-

1006, 000066; Ex-1003, ¶236. For example, Stroud allocates different memory to address ranges. Ex-1004, ¶352. Cragon provides implementation details, e.g., translation tables and bound registers. Ex-1006, 000413-415. Cragon teaches “how” to do what Stroud asks for, and a POSITA would have appreciated the benefits of selecting between different memory arrangements based on application needs. Ex-1003, ¶237.

Furthermore, the combination would not have changed the principle of operation for either reference because it simply applies Cragon’s teachings to select memory arrangements ways that had been known/used for decades. Ex-1003, ¶238.

A POSITA would have had a reasonable expectation of success. The combination simply uses Cragon’s teachings for selecting between textbook memory modes. E.g., Ex-1006, 000219-225, 000255-431, 000491-597. These were routine details from undergraduate curriculums that a POSITA would have readily been able to implement, particularly since these techniques were widely used with various memory types. *See, e.g.*, Ex-1032, Abs., Figs. 1A, 1B, 2; Ex-1033, Abs. Moreover, Stroud teaches CLDs/FPGAs; a POSITA would have been more than capable of implementing CLD/FPGA designs to use textbook memory arrangements. *Supra* §III. Implementing Cragon’s memory teachings would have

been straightforward and well within the level of ordinary skill. Ex-1003, ¶¶239-240.

3. Claim 9

**9. The method of claim 7, wherein the communicating the data flow comprises:
bypassing the messaging interface network and communicating through a direct connection among two or more pods, cards or components.**

Claim 9 recites overlapping language with claim 2 (Ex-1003, ¶241 (claim comparison)) that is satisfied for the same reasons. *Supra* §V.A.4; Ex-1003, ¶241.

Ground 2 further teaches communicating data flows through a direct connection between Stroud's cards or components. For example, Stroud teaches several direct connections between cards, including direct serial connections whose purpose was to allow cards (and NPs and CLDs on the cards) to communicate between each other. *Supra* §V.A.4. These serial connections (including serial bus for high-speed MLVDS connections) provide alternative paths to communicate data flows of network traffic being generated and processed for simulation. Ex-1004, ¶101, ¶320, ¶¶669-671, Abs., ¶¶6-8, Fig. 46, 14B; Ex-1003, ¶242.

Additionally, it was obvious to send data flows through the front-panel Ethernet because Stroud explains how the front panels are for testing external networks, and Stroud teaches some data flows used for this purpose. *See* Ex-1004,

¶¶114-116, ¶¶121-125, ¶¶128-132, ¶145, Figs. 2C-2E, 6A-9B; *supra* §V.B.2[7c] (explaining data flows); Ex-1003, ¶243.

4. Claim 10

Claim 10 recites overlapping language with claim 3 (Ex-1003, ¶245 (claim comparison)) that is satisfied for the same reasons. *Supra* §V.A.5. Stroud teaches allocating a processor (controller 106) to all four tasks recited in claim 3. Ex-1004, ¶¶356-359, ¶154, ¶346, Fig. 6A; Ex-1003, ¶¶245-246.

C. Ground 3

To the extent PO argues it was not obvious to select an appropriate memory arrangement—despite Cragon’s teachings on the topic—Saulsbury provides additional teachings for selecting between memory arrangements. Each of Stroud’s cards includes multiple types of memory. Cragon teaches common modes for using different memories. Saulsbury provides additional teachings for selecting between memory modes, for example using the memories as a multilevel cache or directly-accessible memory. Ex-1007, Abs. Ground 3 combines these teachings to provide a flexible memory arrangement to Stroud’s cards. The motivation is explained below. *Infra* §V.C.2[7d]. Ex-1003, ¶¶248-250.

1. U.S. Patent App. Pub. 2002/0087821 (“Saulsbury”)

Saulsbury is §102(a)(1) prior art published in 2002. Ex-1003, ¶¶251-252. Saulsbury is directed to a “computer processing architecture with on-chip DRAM

usable as physical memory or cache memory.” Ex-1007, Title. Saulsbury

“control[s] the mode” to use DRAM in either mode. *Id.*, Abs., ¶¶67-68.

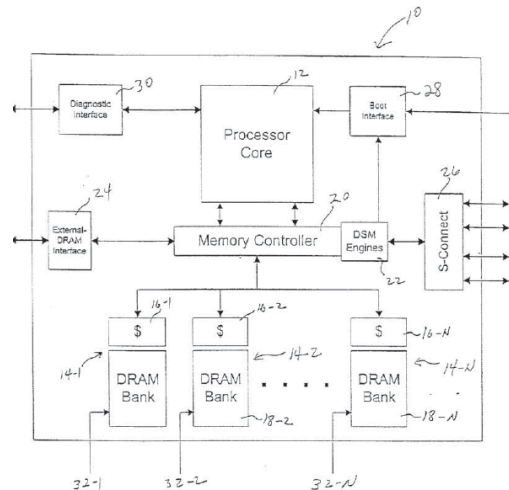


Fig. 1

Ex-1003, ¶253.

Saulsbury is in the same field (*supra* §III) and reasonably pertinent to the '753 patent because it teaches using multiple memory types in a “scalable computer processing architecture.” (Ex-1007, ¶8, ¶20, ¶¶22-26) and provides a solution for flexibility in memory configuration. Ex-1007, Abs., ¶3, ¶¶5-8, ¶¶67-70, Figs. 1, 5; Ex-1003, ¶254.

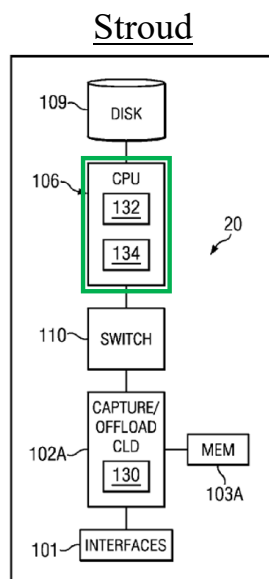
2. Claim 7

Limitations [7pre]-[7c] are taught by Stroud/Cragon for reasons explained above. *Supra* §V.B.2[7pre]-[7c]; Ex-1003, ¶255.

[7d] selecting a memory superposition...

Stroud and Cragon teach [7d], as discussed for Ground 2. *Supra* §V.B.2[7d]. Saulsbury's additional teachings are discussed below. Ex-1003, ¶256.

At least one of the plurality of swappable pods or cards having a plurality of types of memory. Stroud teaches each of its cards having a controller CPU and multiple types of memory. *Supra* §V.B.2[7d]; Ex-1004, ¶127, ¶¶169-170, Figs. 5-9A, 14A-14B, ¶717, Figs. 55A-59B. The combination with Saulsbury further adds on-chip DRAM inside the CPU. Ex-1007, ¶65, ¶¶2-3, ¶¶5-8, Fig. 1.



Ex-1003, ¶257-258.

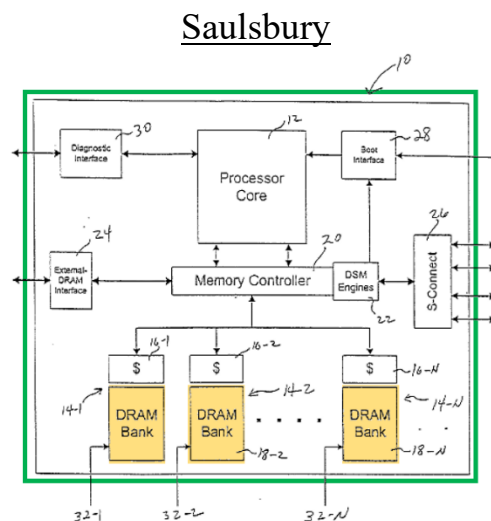


Fig. 1

A selectable arrangement of the plurality of types of memory. Saulsbury provides additional teachings for selecting a memory superposition in cards having a multiple types of memory, with a selectable arrangement of those memories.

Saulsbury teaches “*switch[ing] the memory mode* between physical memory and cache memory, depending on the particular application of the processor and memory chip.” Ex-1007, ¶3, ¶65, Fig. 1. The arrangement is selected using a control bit. *Id.*, ¶68; Ex-1003, ¶259.

Ground 3 applies Saulsbury’s teachings to Stroud’s cards. The combination therefore teaches a selectable arrangement of the plurality of types of memory on each card, where the mode is selectable (e.g., using control bits) depending on memory needs at any particular time. Ex-1007, ¶3, ¶¶65-68. Since Stroud aims to provide a flexible platform (Ex-1004, ¶321, ¶¶106-113, ¶343, ¶373), it was obvious to apply Saulsbury’s teachings to select between textbook memory arrangements as taught by Cragon for the memories on each card. *Supra* §V.B.2[7d]; Ex-1003, ¶260.

Additionally, Saulsbury adds on-chip DRAM to a CPU. Ex-1007, ¶3, ¶¶67-68, Figs. 1, 3. Applied to Stroud’s controller CPU, the combination teaches a selectable arrangement of the on-chip DRAM, disk, and other memory on the card. *Id.*

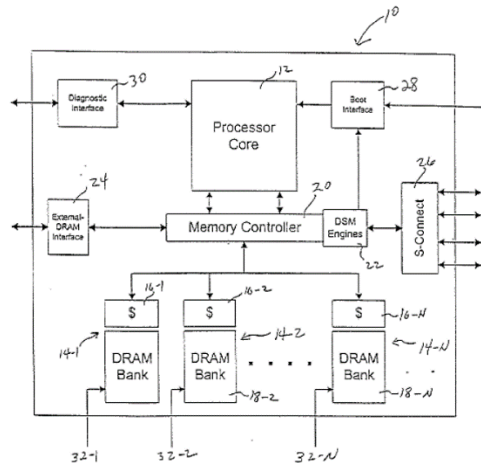


Fig. 1

Ex-1003, ¶261.

Multilevel cache. Cragon teaches multilevel cache. The combination with Saulsbury further teaches using on-chip DRAM as “cache memory.” Ex-1007, ¶¶2-3, ¶¶5-8, Fig. 1. Saulsbury also teaches SRAM-based instruction and data caches within the processor core, which was common by the 2000s. Ex-1007, ¶3, ¶26, ¶¶36-37, ¶¶45-47, Fig. 3. Therefore, Saulsbury teaches multi-level cache, e.g., L1 cache in the processor core and on-chip DRAM as a second level (L2), for storing recently-accessed data from disk (Ex-1004, ¶127, ¶170, ¶3 (“cache memory, and more particularly, second-level cache memory”), Figs. 5-9A, 14A-B) or external DRAM (Ex-1007, Fig. 1; Ex-1004, Figs. 14A-14B (“DDR3”)). This is consistent with Cragon’s teachings of small/fast L1 close to the processor core. Ex-1006, 000219; *see* Ex-1007, ¶26, ¶¶36-37, ¶¶45-47, Fig. 3. The second level cache is larger and slower than L1; therefore, it was obvious to use, e.g.,

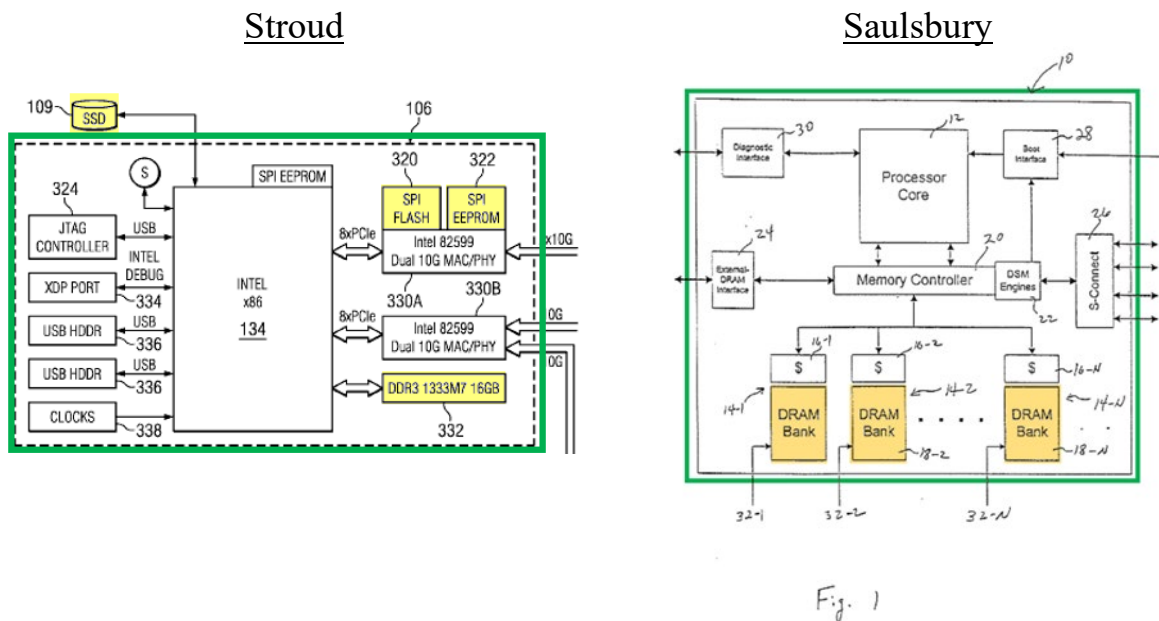
Saulsbury's on-chip DRAM or other larger SRAM or DRAM memories on the card. Ex-1007, ¶3, ¶5, ¶8, Fig. 1; Ex-1004, ¶127; Ex-1003, ¶¶262-263.

Interleaved memory. Saulsbury's on-chip DRAM provides another memory type for Stroud's cards, for use with interleaved memory, as Cragon teaches. *Supra* §V.B.2[7d]; Ex-1003, ¶264.

Swapped memory over a specified address range. Saulsbury's on-chip DRAM adds an additional memory type for use with swapped memory over a specified address range, as Cragon teaches. *Supra* §V.B.2[7d]. For example, Saulsbury teaches on-chip DRAM as direct physical memory; it was obvious to use on-chip DRAM as main physical memory (Ex-1007, ¶¶2-3, ¶¶7-8, ¶¶67-68, Fig. 1) and the disk attached to the CPU as non-volatile memory for the swap file (see Ex-1004, ¶308, ¶195, ¶196, ¶269, Figs. 5-10, 14A-B, 20, 23, 30, 49). This aligns with Cragon's teachings of DRAM as main memory and disk for the swap file. Ex-1006, 000256; Ex-1003, ¶265.

Memory allocated to specified address ranges according to parameterized task needs. Cragon allocates memory to specified address ranges for particular tasks according to parameterized task needs for those tasks (e.g., capacity, bandwidth, service rate, response time, and cost). *Supra* §V.B.2[7d]; Ex-1006, 000536-541. Saulsbury's on-chip DRAM provides another memory type to be allocated to a specified address range, for example, to a range for tasks that

require quick response but small capacity (Ex-1006, ¶¶65), with external DDR3 DRAM allocated to a range for tasks requiring larger capacity, while tolerating higher latency, and disks allocated to ranges for tasks that require even larger capacity or non-volatile storage. *See* Ex-1006, 000536-541.



Ex-1003, ¶¶266-268.

Motivation to Combine. Stroud seeks to provide a platform with scalable capacity and flexible modules for custom software and user-definable hardware (CLD) functions performing a wide variety of applications. Ex-1004, ¶¶382, ¶321, ¶399, ¶493, ¶128, ¶144. Therefore, a POSITA would have been motivated to provide flexible memory arrangements to support Stroud’s flexible architecture and the variety of potential functions on Stroud’s cards. Ex-1003, ¶269. This was the motivation explained for combining Stroud and Cragon, which likewise applies here. *Supra* §V.B.2[7d] (motivation subsection); Ex-1003, ¶269.

For additional flexibility, each of Stroud's cards includes a controller CPU. Ex-1004, ¶688. Saulsbury provides teachings for that CPU and Stroud's cards more generally. Saulsbury teaches two techniques that further Stroud's stated goals of scalability and flexibility: (1) the addition of on-chip DRAM to a CPU and (2) techniques for selecting the arrangement of a memory hierarchy. As Saulsbury explains, its teachings are directed to "scalable computer processing architecture[s]" (Ex-1007, ¶8, ¶16) and provide flexible memory arrangements "depending on the particular application." *Id.*, ¶3. Therefore, a POSITA would have been motivated to apply Saulsbury's teachings to Stroud's CPU and cards. This would have been a combination of prior art elements according to known methods to yield the predictable result of a CPU with on-chip DRAM and selectable modes, as Saulsbury teaches. Ex-1003, ¶270.

The combination uses Saulsbury's known techniques to improve Stroud's similar devices the same way. Stroud provides motivation to combine with known CPU architectures: "[c]ontroller 106 may compare any suitable controller" including "a general purpose CPU with multiple cores" or "an off-the-shelf processor." Ex-1004, ¶154. Stroud applies its teachings across a broad range of CPU architectures, including various x86 and RISC processors. *Id.* ¶104, ¶110. Saulsbury originated from Sun, which was known for RISC CPUs. *See* Ex-1007.

Therefore, a POSITA would have been motivated to apply Saulsbury's CPU teachings to Stroud's controller CPU and card. Ex-1003, ¶271.

The combination offers many benefits. Saulsbury's on-chip DRAM provides "simple, physical memory" close to the processor core and high-performance cache. Ex-1007, ¶3. Moreover, having a selectable memory arrangement provides flexibility and allows memory to be optimized for particular applications at any given time to suit the function of the card, which reinforces the flexible nature of Stroud's architecture, which uses CLDs and NPs for a variety of applications with different memory requirements. *Id.*; Ex-1003, ¶272.

The prior art provides express motivation to combine by teaching the benefits and use of on-chip DRAM and selectable memory arrangements. Ex-1031, Abs., Figs. 3, 4B, 5; Ex-1026, 1:48-58; Ex-1032, Abs., Fig. 8; Ex-1027, [54] Title, Abs., Fig. 4; Ex-1028, (54) Title, ¶5, Fig. 2; Ex-1029, Abs., Fig. 6; Ex-1030, Abs., ¶8. This type of flexible memory arrangement increased performance at lower cost. *See* Ex-1029, 1:23-33; Ex-1032, ¶21-22, ¶49. Consistent with the combination of Ground 3, selecting to cache between different types of memory was suggested in the art. Ex-1030, ¶35. The art also teaches on-chip memory and suggests using selectable memory arrangements when "more memory may be required than is available on the chip" (Ex-1031, ¶6), which remedied disadvantages of fixed memory systems (Ex-1031, ¶¶7-8). Selectable memory

arrangements increased the longevity of non-volatile memory (Ex-1033, Abs) and were used to selectively “combine[] the nonvolatility and reduced per-bit price of flash memory with the speed and durability of DRAM.” Ex-1033, 2:6-16. On-chip memory increased access speed. Ex-1031, ¶36, ¶11, Fig. 1; Ex-1034, 1:13-32; Ex-1032, Abs., Fig. 3. Additionally, a POSITA would have been motivated to improve Stroud’s memory as recognized in the art. Ex-1028, ¶2. A POSITA would have recognized such benefits and would therefore have been motivated to apply Saulsbury’s techniques to Stroud’s cards. Ex-1003, ¶¶273-274.

The combination would not have changed the principle of operation for any of these reference because it simply applies Saulsbury’s teachings for on-chip DRAM and a selectable arrangement of memory modes to Stroud and Cragon. Stroud’s cards/chassis would still have been used for a network testing platform, as Stroud teaches. Ex-1004, ¶¶85-86. Cragon’s teachings would have been used to arrange memory in the way that was taught by Cragon. Ex-1006, 000048, 000071-73, 000219-220, 000496-497, 000255-265, 000413-415, 000536-548. And Saulsbury’s on-chip DRAM would still have applied to a CPU, while Saulsbury’s selection of memory modes would still have been used to control the operation and arrangement of memory in a memory hierarchy. Ex-1007, Abs., ¶8; Ex-1003, ¶275.

A POSITA would have had a reasonable expectation of success in combining the teachings of Stroud and Cragon (explained for Ground 2, *supra* §V.B.2[7d]) and in further combining Saulsbury. Stroud teaches flexible configurations for its cards that encompasses a wide range of possible CPUs as the card controller. Ex-1004, ¶154, ¶110. Cragon teaches various memory arrangements commonly used with CPUs. Ex-006, 000040-43, 000496-497, 000255-265, 000413-415, 000536-548. Saulsbury teaches a general purpose processor and applies its teachings to CPUs. *See* Ex-1007, ¶¶16-22, Figs. 3-5; Ex-1003, ¶276.

Additionally, the combination would not have changed the principle of operation, as explained above. It simply adds Saulsbury's on-chip DRAM and techniques for controlling the memory mode. Ex-1007, ¶68, ¶8. That technique—using control bits—had been conventional for decades, and on-chip DRAM was also known. Ex-1028, ¶22, ¶14, Fig. 7; Ex-1034, 2:3-13. Given the express teachings of numerous prior art references, a POSITA would have been confident in the success of applying Saulsbury's techniques to Stroud and Cragon. *See, e.g.*, Ex-1031, Abs., Figs. 3, 4B, 5; Ex-1026, 1:48-58; Ex-1032, Abs., Fig. 8; Ex-1027, Title, Abs., Fig. 4; Ex-1028, Title, ¶5, Fig. 2; Ex-1029, Abs., Fig. 6; Ex-1030, Abs., ¶8; Ex-1003, ¶277.

A POSITA would have been more than capable in making the combination. The design and use of CPUs, memory types and hierarchies, and DRAM was part of undergraduate EE/CE education, and work experience in processing architectures would have provided experience to the use of different memory arrangements and configurations, as well as to the settings and controls for computer systems. Education and work experience in user-defined hardware and software modules would have encompassed the types of PLDs, FPGAs, and NPs used in Stroud's cards. *Supra* §III; Ex-1003, ¶278.

3. Claims 9-10

Saulsbury provides additional teachings for claim 7 but does not otherwise alter claims 9-10, which are obvious based on Stroud and Cragon. *Supra* §V.B.3-4; Ex-1003, ¶279.

D. Ground 4

Claims 1 and 7 were directed to separate aspects of a platform: the packet header used for network communications in claim 1 versus the memory arrangement inside the swappable card/pod in claim 7. Several dependent claims (claims 4, 8, 15) amalgamate these limitations together. Ex-1003, ¶280.

Accordingly, Ground 4 combines the teachings from Ground 1 (which addressed the packet header of claim 1) with Ground 3 (which addressed the memory limitations of claim 7). Both Grounds were based on Stroud. Ground 1

combined Stroud with DaSilva’s chassis ID teachings. Ground 3 combined Stroud with the selectable memory arrangements taught by Saulsbury and Cragon. Those same motivations apply here. *Supra* §V.A.3[1e] (motivation for Ground 1), §V.C.2[7d] (motivation for Ground 3). Since the combinations were directed at separate aspects of Stroud’s platform—network packet details in Ground 1 versus hardware memory arrangements in Ground 3—the combinations would not have precluded each other. Ex-1003, ¶281.

1. Claim 4

Ground 1 teaches claim 1. *Supra* §§V.A.3[1pre]-[1e]. The combination with Saulsbury and Cragon teaches the limitations of claim 4, which are identical to [7d]. *Supra* §V.C.2[7d]; Ex-1003, ¶282.

2. Claim 8

Ground 2 teaches claim 7. *Supra* §§V.B.2[7pre]-[7d]. The combination with DaSilva further teaches the limitations of claim 8, which are identical to [13d] and shares language with [1e]. *Supra* §V.A.3[1e], §V.A.6[13d]; Ex-1003, ¶283.

3. Claim 15

Ground 1 teaches claim 13. *Supra* §§V.A.6[13pre]-[13d]. The combination with Cragon and Saulsbury further teaches the limitations of claim 15, which are identical to [7d]. *Supra* §V.C.2[7d]; Ex-1003, ¶284.

VI. STANDING

Petitioner certifies that Petitioner is not barred or estopped from requesting this inter partes review, and the '753 patent is eligible for it.

VII. DISCRETIONARY ANALYSIS FOR REVIEW

Discretionary denial is not warranted. This Petition's merits are compelling, which "alone demonstrates that the PTAB should not discretionarily deny institution under *Fintiv*." See June 21, 2022 Director's Memo, 3-5. The Grounds present compelling obviousness theories that, "if unrebutted in trial, would plainly lead to a conclusion that one or more claims are unpatentable by a preponderance of the evidence." *Id.*, 4.

Fintiv. Even if the Board were to address *Fintiv*, the factors favor institution. The district court case is in its early stages: claim construction initial disclosures were exchanged, but no claim construction briefing has occurred. Trial is currently May 18, 2026. However, the magistrate judge has recommended transfer to the Northern District of California, which will cause the case to be rescheduled.

Advanced Bionics. Denial under §325(d) is likewise unwarranted because Stroud, DaSilva, Saulsbury, and Cragon were not cited or considered during prosecution. This is the only IPR filed against the challenged claims to date. *General Plastic* factors do not weigh against institution.

VIII. MANDATORY NOTICES AND FEES

1. **Real Party In Interest:** The real parties-in-interest here are Advanced Micro Devices, Inc. and Pensando Systems, Inc. No other parties directed, controlled, or funded this IPR.

2. **Related Matters:** *Concurrent Ventures, LLC et al. v. Advanced Micro Devices, Inc. et al.*, Case No. 1-24-cv-00335 (W.D. Tex.)

3. **Lead and Backup Counsel:** Pursuant to 37 C.F.R. §42.8(b)(3) and 42.10(a), Petitioner designates Harper Batts (Reg. No. 56,160) as lead counsel and Chris Ponder (Reg. No. 77,167) and Jeffrey Liang (Reg. No. 69,043) as back-up counsel, each of Sheppard, Mullin, Richter & Hampton LLP.

4. **Service Information:** Petitioner consents to service by electronic mail addressed to:

HBatts@sheppardmullin.com;
CPonder@sheppardmullin.com;
JLiang@sheppardmullin.com; and
LegalTM-AMD-Concurrent@sheppardmullin.com

5. Petitioner's counsel may also be served by mail or hand delivery at Sheppard, Mullin, Richter & Hampton LLP, 1540 El Camino Real Suite 120, Menlo Park, CA 94025. Petitioner's counsel may be reached by telephone at (650) 815-2673 and by facsimile at (650) 815-4668.

6. **Fees:** The Office is authorized to charge fees for this Petition to Deposit Account 50-4561, Ref. 71EY-388593.

Respectfully submitted,

SHEPPARD, MULLIN,
RICHTER & HAMPTON LLP

/Harper Batts/

Harper Batts (Reg. No. 56,160)
Attorneys for Petitioner

CLAIM APPENDIX

[1pre]. A platform for data flow processing, comprising:

[1a] one or more swappable pods or cards in one or more chassis, coupled through a messaging interface network;

[1b] each of the one or more swappable pods or cards having one or more hardware modules or one or more software modules;

[1c] one or more of the plurality of swappable pods or cards having a portion for user-definable hardware modules or user-definable software modules;

[1d] the plurality of swappable pods or cards being user-configurable to implement data flow processing architectures; and

[1e] a network coupled to the one or more swappable pods or cards and supporting messaging-based communication using packets each having a header with a chassis identifier, a board identifier, a module identifier, an instance identifier, and a type identifier, so that each type of module, each instance of a type of module, and each module on each board in each chassis can be addressed through the header.

2. The platform of claim 1, further comprising:

at least a subset of the one or more swappable pods or cards having an option for direct connection to one or more pods, cards or components, bypassing the module messaging interface network.

3. The platform of claim 1, wherein the one or more swappable pods or cards comprises at least one processor card having a at least one processor allocatable for offloading hardware tasks to software, orchestrating data flows, orchestrating configurations of hardware modules, and executing in cooperation with a sandbox of one of the plurality of swappable pods or cards.

4. The platform of claim 1, further comprising:

at least one of the one or more swappable pods or cards having a plurality of types of memory and memory superposition with selectable arrangement of the plurality of types of memory as multilevel cache, interleaved memory, swapped memory over a specified address range, and memory allocated to specified address ranges according to parameterized task needs.

[7pre]. A method for data flow processing in a platform, comprising:

[7a] configuring a plurality of swappable pods or cards in one or more chassis, coupled through a messaging interface network, to implement a data flow processing architecture;

[7b] configuring user-definable hardware modules or user-definable software modules in each of one or more of the plurality of swappable pods or cards;

[7c] communicating a data flow in accordance with the implemented data flow processing architecture among one or more hardware modules or one or more

software modules in each of the plurality of swappable pods or cards, including the user-definable hardware modules or user-definable software modules; and

[7d] selecting a memory superposition in at least one of the plurality of swappable pods or cards having a plurality of types of memory with selectable arrangement of the plurality of types of memory as multilevel cache, interleaved memory, swapped memory over a specified address range, and memory allocated to specified address ranges according to parameterized task needs.

8. The method of claim 7, wherein the communicating the data flow comprises:

communicating the data flow through the messaging interface network, comprising a network coupled to the plurality of swappable pods or cards and supporting messaging-based communication using packets each having a header with a chassis identifier, a board identifier, a module identifier, an instance identifier, and a type identifier, so that each type of module, each instance of a type of module, and each module on each board in each chassis can be addressed through the header.

9. The method of claim 7, wherein the communicating the data flow comprises:

bypassing the messaging interface network and communicating through a direct connection among two or more pods, cards or components.

10. The method of claim 7, further comprising:

allocating at least one processor of a processor card, from among the plurality of swappable pods or cards, to offloading hardware tasks to software, orchestrating data flows, orchestrating configurations of hardware modules, or executing in cooperation with a sandbox of one of the plurality of swappable pods or cards.

[13pre]. A tangible, non-transitory, computer-readable media having instructions thereupon which, when executed by a processor, cause the processor to perform a method comprising:

[13a] configuring a plurality of swappable pods or cards in one or more chassis, coupled through a messaging interface network, to implement a data flow processing architecture;

[13b] configuring user-definable hardware modules or user-definable software modules in each of one or more of the plurality of swappable pods or cards;

[13c] communicating a data flow in accordance with the implemented data flow processing architecture among one or more hardware modules or one or more software modules in each of the plurality of swappable pods or cards, including the user-definable hardware modules or user-definable software modules; and

[13d] communicating the data flow through the messaging interface network, comprising a network coupled to the plurality of swappable pods or cards and

supporting messaging-based communication using packets each having a header with a chassis identifier, a board identifier, a module identifier, an instance identifier, and a type identifier, so that each type of module, each instance of a type of module, and each module on each board in each chassis can be addressed through the header.

14. The computer-readable media of claim 13, wherein the method further comprises:

allocating at least one processor of a processor card having a plurality of processors, from among the plurality of swappable pods or cards, to offloading hardware tasks to software, orchestrating data flows, orchestrating configurations of hardware modules, or executing in cooperation with a sandbox of one of the plurality of swappable pods or cards.

15. The computer-readable media of claim 13, wherein the method further comprises:

selecting a memory superposition in at least one of the plurality of swappable pods or cards having a plurality of types of memory with selectable arrangement of the plurality of types of memory as multilevel cache, interleaved memory, swapped memory over a specified address range, and memory allocated to specified address ranges according to parameterized task needs.

CERTIFICATE OF SERVICE

The undersigned hereby certifies that on February 12, 2025, true and correct copies of the foregoing Petition for *Inter Partes* Review of U.S. Patent No. 10,873,753 and Exhibits 1001-1044 were served in entirety on the Patent Owner at the following addresses of record as listed on Patent Center, via Priority Mail Express®:

Michael Gencarella
12875 Lucero Lane
Los Altos Hills, CA 94022
Official Correspondence Address

Concurrent Ventures LLC
c/o Beeson, Jesse D.
240 Woodscape Court
Johns Creek, GA, 30022-3244
Patent Owner Correspondence Address

XtreamEdge, Inc.
c/o The Corporation Trust Company
Corporation Trust Center
1209 Orange Street
Wilmington, DE 19801
Patent Owner Correspondence Address

The undersigned certifies that true and correct copies of the Petition for *Inter Partes* Review of U.S. Patent No. 10,873,753 and Exhibits 1001-1044 were also sent via electronic mail to the attorneys of record for Plaintiff in the concurrent litigation matter(s):

Michael Matulewicz-Crowley
mmatulewicz-
crowley@reichmanjorgensen.com
Christine E. Lehman
clehman@reichmanjorgensen.com;
myohannes@reichmanjorgensen.com

Ariane S. Mann
amann@reichmanjorgensen.com
Amy L. Ruhland
aruhland@reichmanjorgensen.com
Laura Carwile
lcarwile@reichmanjorgensen.com

SHEPPARD, MULLIN,
RICHTER & HAMPTON LLP

/Harper Batts/
Harper Batts (Reg. No. 56,160)
Attorneys for Petitioners

Date: February 12, 2025

**CERTIFICATE OF COMPLIANCE WITH
TYPE-VOLUME LIMITATION, TYPEFACE REQUIREMENTS,
AND TYPE STYLE REQUIREMENTS**

1. This Petition complies with the type-volume limitation of 14,000 words, comprising 13,315 words, as counted using the Microsoft Word software that was used to prepare this paper, excluding the parts exempted by 37 C.F.R. § 42.24(a).

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

Respectfully submitted,

SHEPPARD, MULLIN,
RICHTER & HAMPTON LLP

/Harper Batts/
Harper Batts (Reg. No. 56,160)
Attorney for Petitioner

Date: February 12, 2025

1540 El Camino Real Suite 120
Menlo Park, CA 94025
Tel: (650) 815-2673