Exhibit 753-04
Invalidity of the ’753 Patent by Stroud

The asserted claims of U.S. Patent No. 10,873,753 are anticipated and/or rendered obvious in view of U.S. Patent Application Publication 2013/0346667 (“Stroud”) (avail. at. PA035425-PA035549) alone or in combination with the knowledge of one of skill in the art or other prior art.

This claim chart is based on Defendants’ present understanding of the asserted claims, Plaintiffs’ vague and overbroad infringement contentions (and corresponding, implicit claim constructions), and Defendants’ investigation to date. Defendants are not adopting Plaintiffs’ apparent constructions, nor are Defendants admitting to the correctness of any particular construction. Defendants reserve all rights to amend this invalidity claim chart in light of any claim construction developments or any amendments to Plaintiffs’ infringement contentions. Further, as discovery is ongoing and Defendants continue to seek discovery from third parties regarding the references identified in Defendants’ invalidity contentions as well as other potential prior art, Defendants reserve the right to amend or supplement this claim chart in connection with Defendants’ continuing investigation. At a minimum, Defendants intend to revise its invalidity contentions as appropriate based on discovery from third parties as discovery progresses.

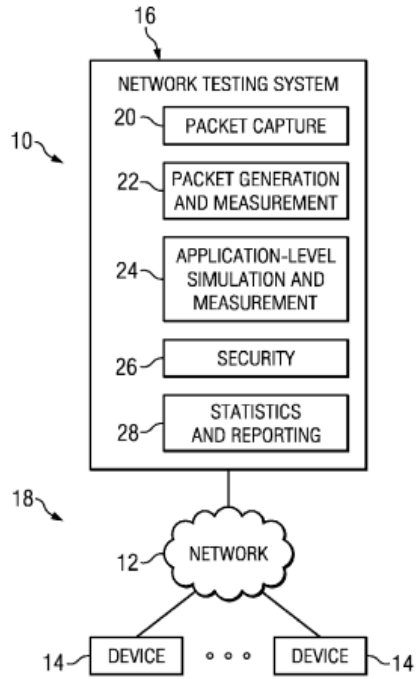
The claim chart below identifies where each limitation of each asserted claims of the ’753 patent is anticipated and/or rendered obvious in view of Stroud. The citations provided below are exemplary, rather than exhaustive, and Defendants reserve the right to rely upon any other portion of the cited references.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
[1.pre] A platform for data flow processing comprising:	Stroud discloses or renders obvious “[a] platform for data flow processing[.]” *See, e.g.*: A multi-card system includes a card having a plurality of serial port devices, an inter-card serial bus connection configured for connection to an inter-card serial bus, and an intra-card and inter-card serial port access system. (Stroud, Abstract) FIG. 1 illustrates a general block diagram of an arrangement 10 for testing the performance of a communications network 12 and/or one or more network devices 14 using a network testing system 16, according to certain embodiments of the present disclosure. Test devices 14 may be part of a network 12 tested by network testing system 16, or may be connected to network testing system 16 by network 12. Thus, network testing system 16 may be configured for

IPR2025-00486
Concurrent Ventures LLC, XtreamEdge Inc. EX2041

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	testing network 12 and/or devices 14 within or connected to network 12. For the sake of simplicity, the test network 12 and/or devices 14 are referred to herein as the test system 18. Thus, a test system 18 may comprise a network 12, one or more devices 14 within a network 12 or coupled to a network 12, one or more hardware, software, and/or firmware components of device(s) 14, or any other component or aspect of a network or network device. (Stroud, [0080]) Network testing system 16 may be configured to test the performance (e.g., traffic-handling performance) of devices 14, the security of a test system 18 (e.g., from security attacks), or both the performance and security of a test system 18. In some embodiments, network testing system 16 configured to simulate a realistic combination of business, recreational, malicious, and proprietary application traffic at sufficient speeds to test both performance and security together using the same data and tests. In some embodiments, network testing system 16 is configured for testing content-aware systems 18 devices 14 and/or content-unaware systems 18. (Stroud, [0081]) In some embodiments, network testing system 16 is configured to model and simulate network traffic. The network testing system 16 may act as virtual infrastructure and simulate traffic behavior of network devices (e.g., database server, Web server) running a specific application. The resulting network traffic originated from the network testing system 16 may drive the operation of a test system 18 for evaluating the performance and/or security of the system 18. Complex models can be built on realistic applications such that a system 18 can be tested and evaluated under realistic conditions, but in a testing environment. Simultaneously, network testing system 16 may monitor the performance and/or security of a test system 18 and may collect various metrics that measure performance and/or security characteristics of system 18. (Stroud, [0084]) For example, as shown in FIG. 1, network testing system 16 may comprise any or all of the following testing and simulation subsystems: a high-speed, high-resolution network packet capture subsystem 20, a high-speed packet generation and measurement subsystem 22, an

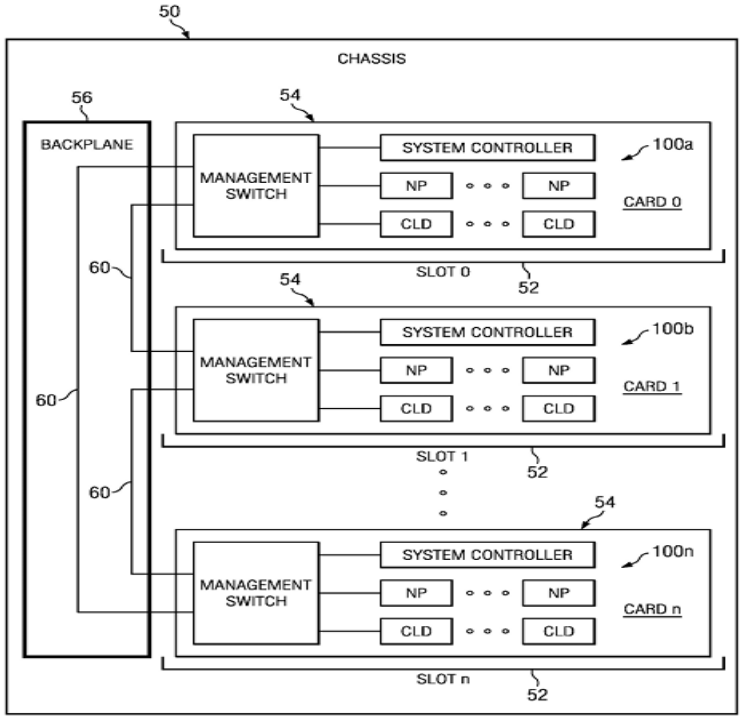
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	application-level simulation and measurement subsystem 24, a security and exploit simulation and analysis subsystem 26, and/or a statistics collection and reporting subsystem 28. Subsystems 20-28 are discussed below in greater detail. In some embodiments, the architecture of network testing system 16 may allow for some or all of subsystems 20-28 to operate simultaneously and cooperatively within the same software and hardware platform. Thus, in some embodiments, system 16 is configured to generate and analyze packets at line rate, while simultaneously capturing that same traffic, performing application simulation, and security testing. In particular embodiments, system 16 comprises custom hardware and software arranged and programmed to deliver performance and measurement abilities not achievable with conventional software or hardware solutions. (Stroud, [0086]) Thus, in some embodiments (e.g., embodiments that support hot swapping of cards 54), to utilize multiple control processors 105 and drives 109 available in a multi-card system 16, as well as to allow for each control processor 106 to run an independent operating system, while maintaining Ethernet connectivity to the backplane 56, system 16 may be configured such that local network processors 105 boot from the local control processor 106 using DHCP, NFS (Network File System), and TFTP (Trivial File Transfer Protocol). This task is divided by a special dynamic configuration for the DHCP server. (Stroud, [0378])

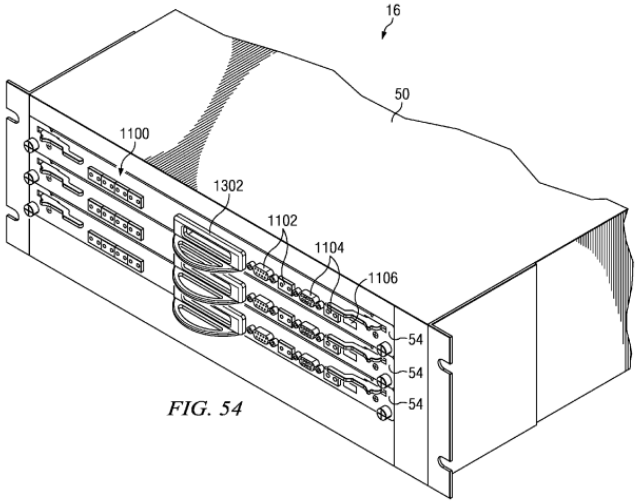
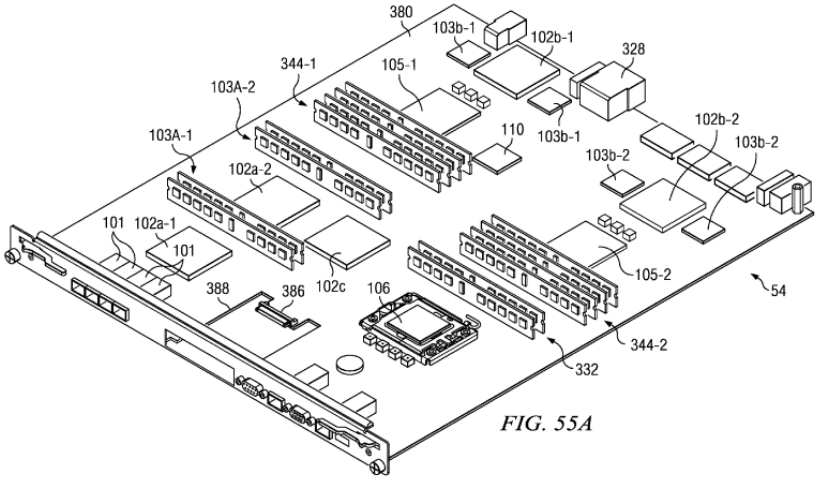
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 *FIG. 1* (Stroud, FIG. 1)
[1.a] one or more swappable pods or cards in one or more chassis, coupled through a messaging interface network;	Stroud discloses or renders obvious “one or more swappable pods or cards in one or more chassis, coupled through a messaging interface network[.]” *See, e.g.*: FIG. 3 illustrates an example configuration of a network testing system 16, according to example embodiments. Network testing system 16 may include a chassis 50 including any suitable number of slots 52, each configured to receive a modular card, or blade, 54. A card or blade 54 may comprise one or more printed circuit boards (e.g., PCB 380 discussed below). For example, as shown, chassis 50 may include Slot 0 configured to receive Card 0, Slot 1 configured to receive Card 1, . . . and Slot n configured to receive Card n, where n equals any suitable number, e.g., 1, 2, 3, 4, 5, 7, or more. For example, in some embodiments, chassis 50

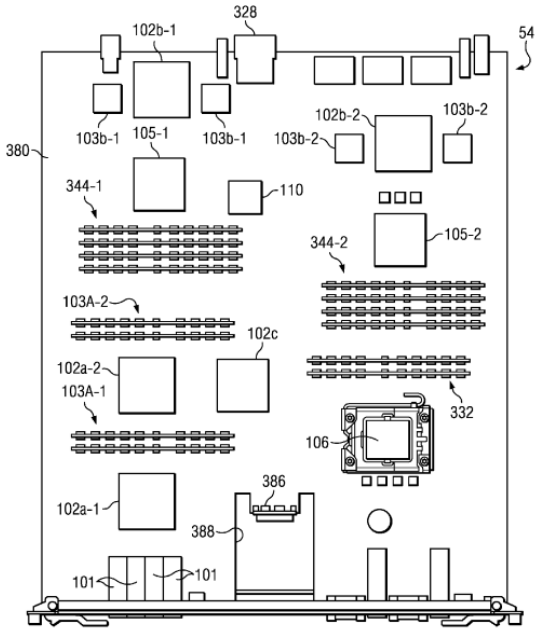
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	is a 3-slot chassis, a 5-slot chassis, or a 12-slot chassis. In other embodiments, system 16 comprises a single card 54. (Stroud, [0098]) Each card 54 may be plugged into a backplane 56, which may include physical connections 60 for communicatively connecting cards 54 to each other, as discussed below. While cards may be interconnected, each card is treated for some purposes as an independent unit. Communications within a card are considered to be “local” communications. Two different cards attached to the same backplane may be running different versions of software so long as the versions are compatible. (Stroud, [0099]) Each card 54 may include any architecture 100 of hardware, software, and/or firmware components for providing the functionality of network testing system 16. For example, card 0 may include an architecture 100 a, card 1 may include an architecture 100 b, . . . , and card n may include an architecture 100 n. The architecture 100 of each card 54 may be the same as or different than the architecture 100 of each other card 54, e.g., in terms of hardware, software, and/or firmware, and arrangement thereof. (Stroud, [0100]) Each architecture 100 may include a system controller, one or more network processors, and one or more CLDs connected to a management switch 110 (and any other suitable components, e.g., memory devices, communication interfaces, etc.). Cards 54 may be communicatively coupled to each other via the backplane 56 and management switches 110 of the respective cards 54, as shown in FIG. 3. In some embodiments, backplane 56 include physical connections for connecting each card 54 directly to each other card 54. Thus, each card 54 may communicate with each other card 54 via the management switches 110 of the respective cards 54, regardless of whether one or more slots 52 are empty or whether one or more cards 54 are removed. (Stroud, [0101]) In some embodiments, each card 54 may be configured to operate by itself, or cooperatively with one or more other cards 54, to provide any of the functionality discussed herein. (Stroud, [0102])

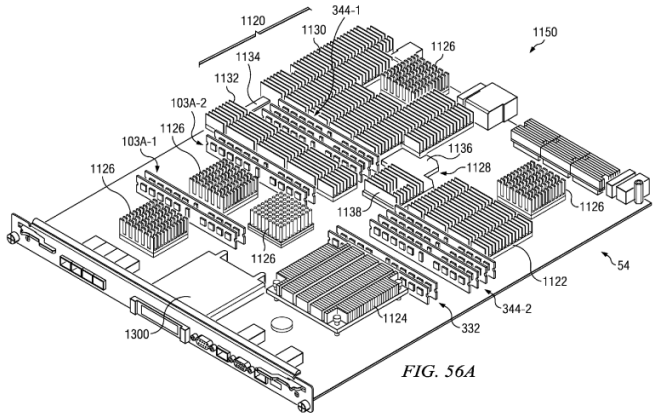
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	Management switch 110 allows and manages communications among the various components of testing architecture 100A, as well as communications between components of testing architecture 100A and components of one or more other cards 54 (e.g., via backplane 56 as discussed above with respect to FIG. 3). Management switch 110 may be a Ethernet layer 2 multi-port switch. (Stroud, [0113]) FIG. 5 is a more detailed illustration of the example testing and simulation architecture 100A shown in FIG. 4, according to an example embodiment. As shown, example testing and simulation architecture 100A includes controller 106; memory 109 coupled to controller 106; two network processors 105; various CLDs 102 (e.g., capture and offload CLDs 102A, router CLDs 102B, and a traffic generation CLD 102C); memory devices 103A and 103B coupled to CLDs 102A and 102B; respectively; management switch 110 coupled to network processors 105 and CLDs 102A, 102B, and 102C, as well as to backplane 56 (e.g., for connection to other cards 54); test interfaces 101 for connecting testing architecture 100A to a system 18 to be tested; and/or any other suitable components for providing any of the various functionality of network testing system 16 discussed herein or understood by one or ordinary skill in the art. (Stroud, [0114]) As discussed above, the components of example architecture 100A may be provided on a single blade 54, and multiple blades 54 may be connected together via backplane 54 to create larger systems. The various components of example architecture 100A are now discussed, according to example embodiments. (Stroud, [0115]) An CLD (Field Programmable Gate Array) is a reprogrammable device that can be modified to simulate many types of hardware. Being reprogrammable, it can be continually expanded to offer new acceleration and network analysis functionality with firmware updates. Example testing and simulation architecture 100A includes various CLDs designated to perform different functions, including two “capture and offload CLDs” 102A capturing data packets, two “router CLDs” 102B for routing data between components of architecture 100A, and a traffic generation CLD 102C for generating traffic that is delivered to the test system 18. (Stroud, [0117])

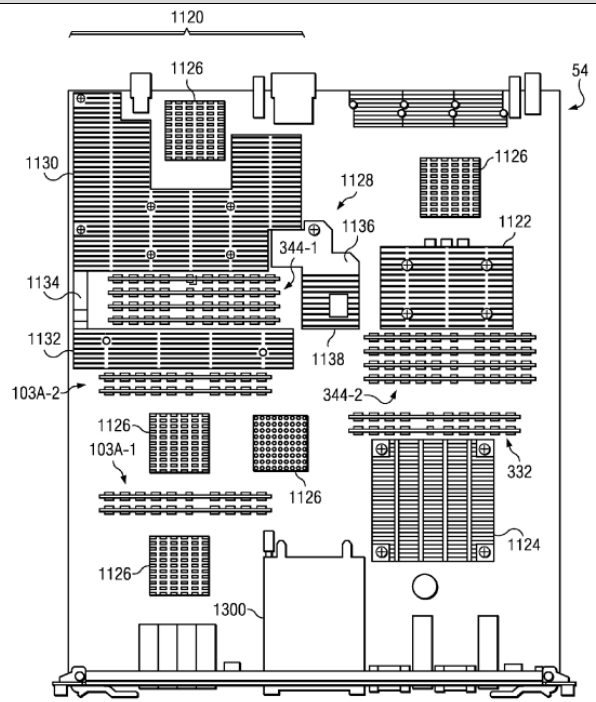
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	Each capture and offload CLDs 102A is connected to one or more test interfaces 101. Thus, CLDs 102A are the first and last device in the packet-processing pipeline. In some embodiments, Ethernet MACs (Media Access Controllers) required to support 10/100/1000 and 10000 Mbps Ethernet standards are implemented within CLDs 102A and interact with the physical-layer devices (PHYs) that implement with the test interfaces 101. (Stroud, [0119]) Each capture and offload CLDs 102A is connected to the traffic generation CLD 102C. In this capacity, the CLDs 102A is a pass-through interface; packets sent by the traffic generation CLD 102C are forwarded directly to an Ethernet test interface 101 for delivery to the test system 18. (Stroud, [0121]) Network testing system 16 may be configured to support any suitable number of cards or blades 54. In one embodiment, system 16 is configured to support between 1 and 14 cards 54 in a single chassis 50. Backplane 56 may provide a system for interconnecting the management Ethernet provided by the management switches 110 of multiple cards 54, as well as system monitoring connections for measuring voltages and temperatures on cards 54, and for debugging and monitoring CPU status on all cards 54, for example. Backplane 56 may also distribute clock signals between all cards 54 in a chassis 50 so that the time stamps for all CPUs and CLDs remain synchronized. (Stroud, [0176]) In some embodiments, network testing system 16 is configured such that cards 54 can boot and operate independently if desired, and be hot-swapped without affecting other the operation of the other cards 54, without the need for additional redundant hardware. Simultaneously, cards 54 can also communicate with each across the backplane 56. Such architecture may improve the scalability and reliability of the system, e.g., in high-slot-count systems. Further, the Ethernet-based architecture of some embodiments may simplify card layout and/or reduce costs. (Stroud, [0382])

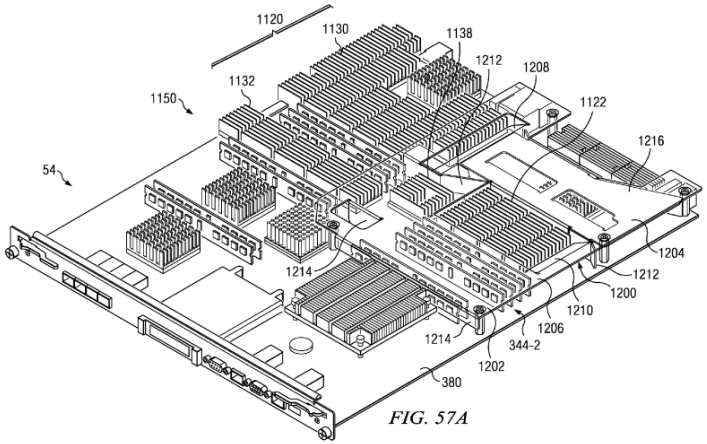
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 *FIG. 3* (Stroud, FIG.3)

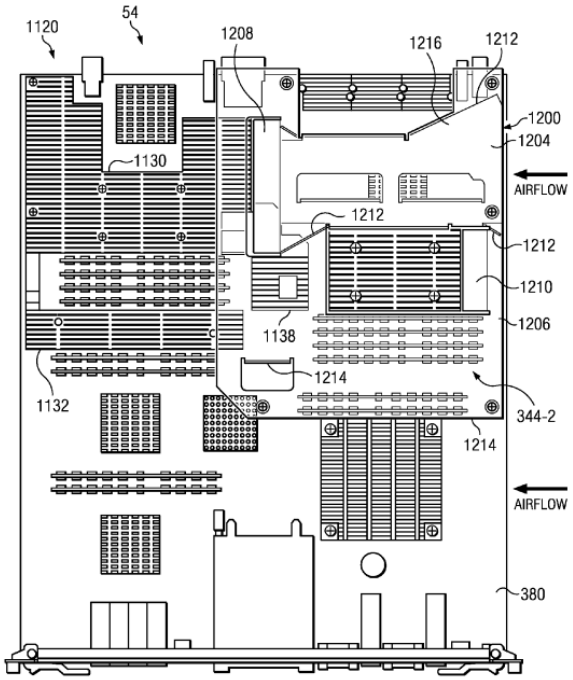
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 *FIG. 54* (Stroud, FIG.54)  *FIG. 55A* (Stroud, FIG.55A)

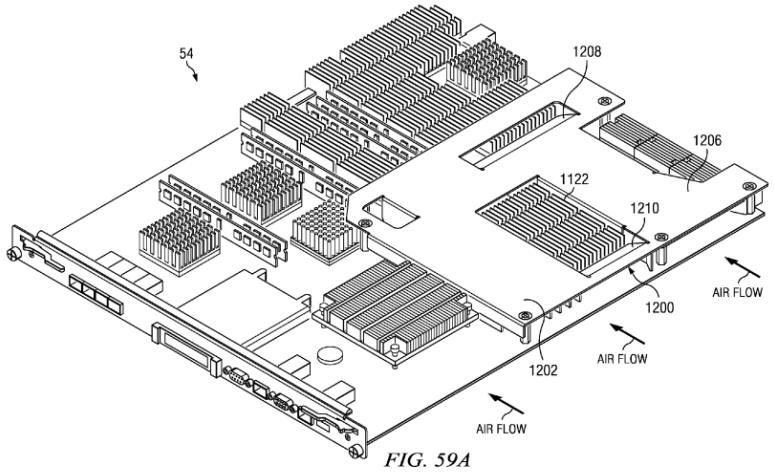
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 *FIG. 55B* (Stroud, FIG.55B)

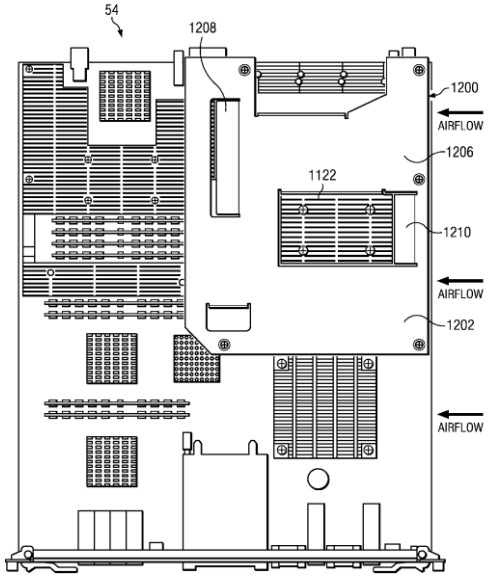
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 *FIG. 56A* (Stroud, FIG.56A)

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 *FIG. 56B* (Stroud, FIG.56B)

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 *FIG. 57A* (Stroud, FIG.57A)

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 *FIG. 57B* (Stroud, FIG.57B)

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 *FIG. 59A* (Stroud, FIG.59A)

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 *FIG. 59B* (Stroud, FIG.59B)
[1.b] each of the one or more swappable pods or cards having one or more hardware modules or one or more software modules;	Stroud discloses or renders obvious “each of the one or more swappable pods or cards having one or more hardware modules or one or more software modules[.]” *See, e.g.*: In some embodiments, network testing system 16 comprises a hardware- and software-based testing and simulation platform that includes of a number of interconnected subsystems. These systems may be configured to operate independently or in concert to provide a full-spectrum solution for testing and verifying network performance, application and security traffic scenarios. These subsystems may be interconnected in a manner to provide high-performance, highly-accurate measurements and deep integration of functionality. (Stroud, [0085]) FIG. 3 illustrates an example configuration of a network testing system 16, according to example embodiments. Network testing system 16 may include a chassis 50 including any

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	suitable number of slots 52, each configured to receive a modular card, or blade, 54. A card or blade 54 may comprise one or more printed circuit boards (e.g., PCB 380 discussed below). For example, as shown, chassis 50 may include Slot 0 configured to receive Card 0, Slot 1 configured to receive Card 1, . . . and Slot n configured to receive Card n, where n equals any suitable number, e.g., 1, 2, 3, 4, 5, 7, or more. For example, in some embodiments, chassis 50 is a 3-slot chassis, a 5-slot chassis, or a 12-slot chassis. In other embodiments, system 16 comprises a single card 54. (Stroud, [0098]) Each card 54 may include any architecture 100 of hardware, software, and/or firmware components for providing the functionality of network testing system 16…The architecture 100 of each card 54 may be the same as or different than the architecture 100 of each other card 54, e.g., in terms of hardware, software, and/or firmware arrangement thereof. (Stroud, [0100]) FIG. 4 is an high-level illustration of an example architecture 100A of a card 54 of network testing system 16, according to an example embodiment. As shown, example architecture 100A, referred to as a “testing and simulation architecture,” may include a controller 106, two network processors 105 and multiple CLDs 102 coupled to a management switch 110, and memory 103 coupled to the CLDs 102. (Stroud, [0103]) In general, network processors 105 are programmed to generate outbound network data in the form of one or more data packets and are programmed to receive and process inbound network data in the form of one or more data packets. In some embodiments, network processors 105 may be general purpose CPUs. In other embodiments, network processors 105 may be specialized CPUs with instruction sets and hardware optimized for processing network data. For example, network processors may be selected from the Netlogic XLR family of processors. (Stroud, [0105]) As used herein, the term configurable logic device (CLD) means a device that includes a set of programmable logic units, internal memory, and high-speed internal and external

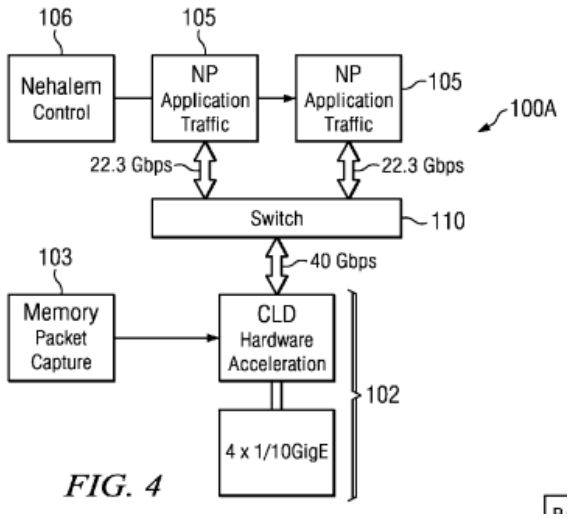
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	interconnections: Examples of CLDs include field programmable gate arrays (FPGAs) (e.g., ALTERA STRATIX family, XILINX VIRTEX family, as examples), programmable logic devices (PLDs), programmable array logic devices (PAL), and configurable programmable logic devices (CPLDs) (e.g., ALTERA MAXII, as an example). A CLD may include task-specific logic such as bus controllers, Ethernet media access controllers (MAC), and encryption/decryption modules. External interconnections on a CLD may include serial or parallel data lines or busses. External interconnections may be specialized to support a particular bus protocol or may be configurable, general-purpose I/O connections. Serial and parallel data connections may be implemented via specialized hardware or through configured logic blocks. (Stroud, [0107]) As used herein, the terms processor and CPU mean general purpose computing devices with fixed instruction sets or microinstruction sets such as x86 processors (e.g., the INTEL XEON family and the AMD OPTERON family, as examples only), POWERPC processors, and other well-known processor families. The terms processor and CPU may also include graphics processing units (GPUs) (e.g., NVIDIA GEFORCE family, as an example) and network processors (NPs) (e.g, NETLOGIC XLR and family, INTEL IXP family, CAVIUM OCTEON, for example). Processors and CPUs are generally distinguished from CLDs as defined above (e.g., FPGAs, CPLDs, etc.) Some hybrid devices include blocks of configurable logic and general purpose CPU cores (e.g., XILINX VIRTEX family, as an example) and are considered CLDs for the purposes of this disclosure. (Stroud, [0110]) As used herein, the term instruction executing device means a device that executes instructions. The term instruction executing device includes a) processors and CPUs, and b) CLDs that have been programmed to implement an instruction set. (Stroud, [0112]) As discussed above, the components of example architecture 100A may be provided on a single blade 54, and multiple blades 54 may be connected together via backplane 54 to create larger systems. The various components of example architecture 100A are now discussed, according to example embodiments. (Stroud, [0115])

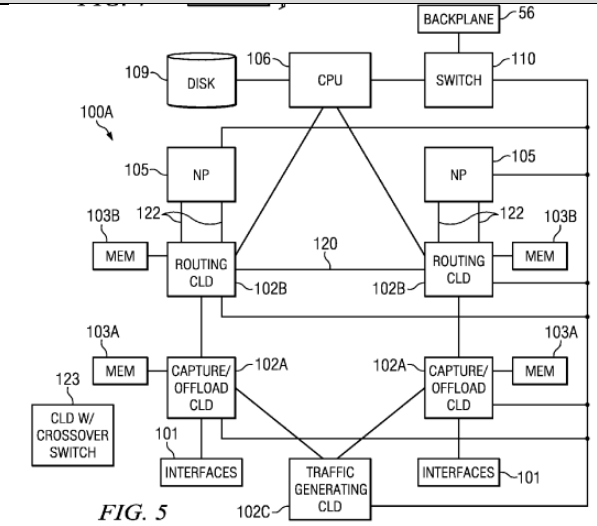
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	Each network processor (NP) 105 may be a general purpose CPU with multiple cores, security, and network acceleration engines. In some embodiments, each network processor 105 may be an off-the-shelf processor designed for network performance. However, it may be very flexible, and may be suitable to perform tasks ranging from low-level, high-speed packet generation to application and user-level simulation. Each network processor 105 may have the following relationships to other components of testing and simulation architecture 100A: (Stroud, [0144]) 1. The primary TCP/IP stack used for network traffic simulation executes on the network processor 105. (Stroud, [0149]) 2. IP and Ethernet-layer address allocation and routing protocols are handled by the network processor 105. (Stroud, [0150]) 3. User and application-layer simulation also run on the network processor 105. (Stroud, [0151]) 4. The network processor 105 works with software on the controller 106 to collect statistics, which may subsequently be used by the statistics and reporting engine 162 of subsystem 28. (Stroud, [0152]) 5. The network processor 105 may also collect statistics from CLDs 102A, 102B, and 102C and report them to the controller 106. In an alternative embodiment, the controller 106 itself is configured to collect statistics directly from CLDs 102A, 102B, and 102C. (Stroud, [0153]) Controller 106 may be the only processor connected directly to the removable disk storage 109. In some embodiments, all firmware or software used by the rest of the system 100A, except for firmware required to start the controller 106 itself (BIOS) resides on the disk drive 109. A freshly manufactured system 100A can self-program all other system components from the controller 106. (Stroud, [0158])

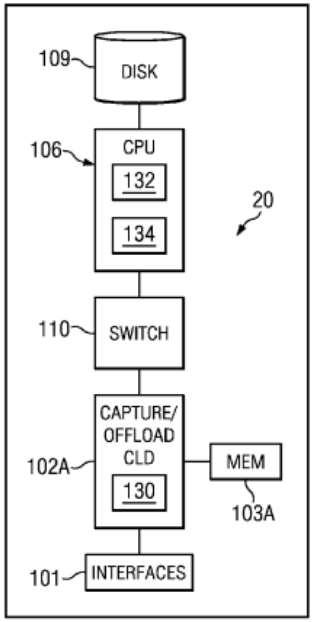
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	The management switch 110 connects to every CLD 102, network processor 105, and control CPU 106 in the system 100A. In some embodiments, management switch 110 comprises a management Ethernet switch configured to allow communication of for 1-10 Gbit traffic both between blades 54 and between the various processors 105, 106 and CLDs 102 on each particular blade 54. Management switch 110 may route packets based on the MAC address included in each packet passing through switch 110. Thus, management 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. In such embodiment, all subsystems are controllable via Ethernet, such that additional processors and CLDs may be added by simply chaining management switches 110 together. (Stroud, [0174]) Network testing system 16 may be configured to support any suitable number of cards or blades 54. In one embodiment, system 16 is configured to support between 1 and 14 cards 54 in a single chassis 50. Backplane 56 may provide a system for interconnecting the management Ethernet provided by the management switches 110 of multiple cards 54, as well as system monitoring connections for measuring voltages and temperatures on cards 54, and for debugging and monitoring CPU status on all cards 54, for example. Backplane 56 may also distribute clock signals between all cards 54 in a chassis 50 so that the time stamps for all CPUs and CLDs remain synchronized. (Stroud, [0176]) While high-speed packet generation and analysis can be used to illustrate raw network capacity, integrity and latency, modern networks also analyze traffic beyond individual packets and instead look at application flows. This is known as deep packet inspection. Also, it is often desired to measure performance of not only the network itself but individual devices, such as routers, firewalls, load balancers, servers, and intrusion detection and prevention systems, for example. (Stroud, [0219])

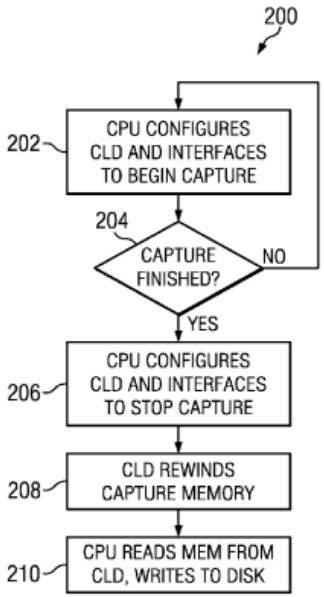
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	(d) Multiple capture and offload CLDs 102A and router CLDs 102B configured to route traffic between the Ethernet MACs 130 and the network processors 105 and to perform packet acceleration offload tasks. (Stroud, [0224]) In some embodiments, the network processors 105 execute software 142 that implements both the networking stack (Ethernet, TCP/IP, routing protocols, etc.) and the application stack that is typically present on a network device. In this sense, the software 142 can simulate network clients (e.g., Desktop PCs), servers, routers, and a whole host of different applications. This programmable “application engine” software 142 is given instructions on how to properly simulate a particular network or application by an additional software layer. This software layer may provide information such as: (Stroud, [0229]) 1. Addresses and types of hosts to simulate on the network, (Stroud, [0230]) 2. Addresses and types of hosts to target on the network, (Stroud, [0231]) 3. Types of applications to simulate, and/or (Stroud, [0232]) 4. Details on how to simulate a particular application (mid-level instructions for application interaction). (Stroud, [0233]) The details on how to simulate applications reside in software 144 that runs on the management processor 134 on controller 106. A user can model an application behavior in a user interface (see, e.g., FIGS. 13A-13D) that provides high-level application primitives, such as to make a database query or load a web page, for example. These high-level behaviors are translated by software 144 into low-level instructions, such as “send a packet, expect 100 bytes back,” which are then executed by the application engine 142 running on the network processor 105. New applications can be implemented by a user (e.g., a customer or in-house personnel), without any changes to the application engine 142 itself. Thus, it is possible to add new functionality without upgrading software. (Stroud, [0234])

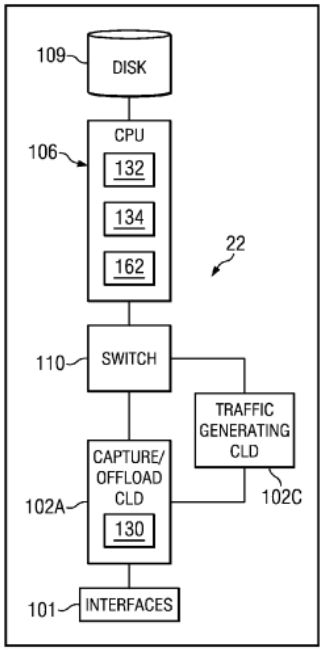
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	Network processor 105 provides software programmable computing that may be optimized for network applications. Network processor may be a NETLOGIC XLR processor. Network processor 105 is coupled to memory 344, boot flash 326, CPLD 348, and Ethernet transceiver 346. Memory 344 is a tangible, computer readable storage medium for storing the instructions and data of the operating system and application processes running on network processor 105. RAM 332 may be double data rate (DDR3) memory. Boot flash 326 is non-volatile memory storing the operating system image for network processor 105. Boot flash 326 may also store application software to be executed on network processor 105. CPLD 348 may provide glue logic between network processor 205 and boot flash 326 (e.g., because the network processor may be capable of interfacing flash memory directly). CPLD 348 may also provide reset and power sequencing for network processor 105. (Stroud, [0309]) Each CLD (e.g., FPGA 102) derives its own MAC address by reading some strapping JO pins on initialization. For example, a four-CLD system may have two pins that encode a binary number between 0 and 3. Strapping resistors are connected to these pins for each CLD, and the CLD reads the value to derive its MAC address. This technique allows system controller 106 to determine all of the encoded information based on the initial ARP (Address Resolution Protocol) request received from an Ethernet device on the internal Ethernet network. This flexibility allows new blades 54 to be defined that are compatible with existing devices without causing backwards compatibility problems. For example, if a new blade is designed that is compatible with an old blade, the model number stays the same. If the new blade adds a new CLD to system 16, then the new CLD is simply assigned a different CLD number for the MAC addressing. However, if a new blade is installed in system 16 that requires additional functionality on the system controller 106, the new blade may be assigned a new model number. Compatibility with existing blades can thus be preserved. (Stroud, [0373]) In some embodiments, software on a network processor assigns SPI4 interfaces to processor cores in the network processor such that all egress packets are sent on the assigned SPI4 interface. In some embodiments, processor cores with an odd number send data on SPI4-1 while those with an even number send data on SPI4-0. A simple bit mask operation can be

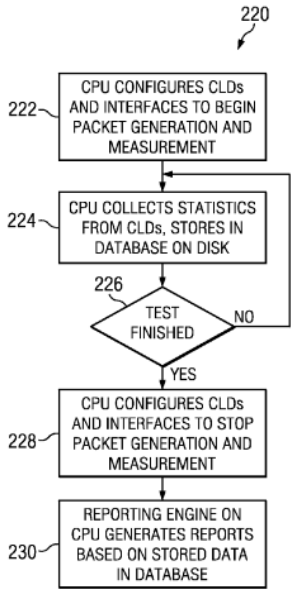
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	used to implement this approach: SPI4 Interface—CORE_ID & 0x1. This approach could be scaled to processors with additional SPI4 ports using a modulus function. (Stroud, [0464])  (Stroud, FIG. 4)

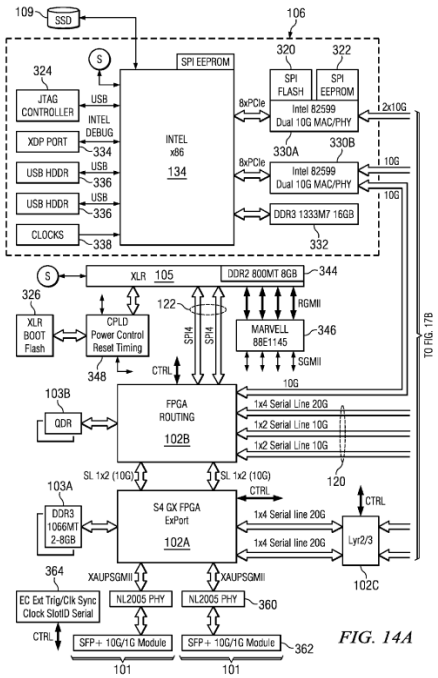
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 (Stroud, FIG. 5)

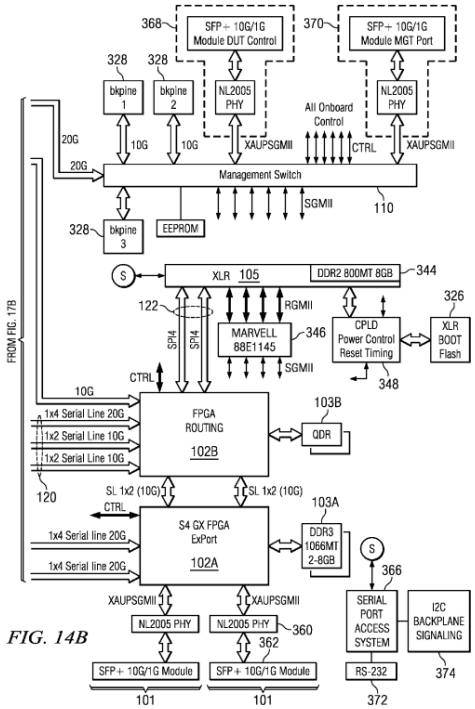
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 *FIG. 6A* (Stroud, FIG. 6A)

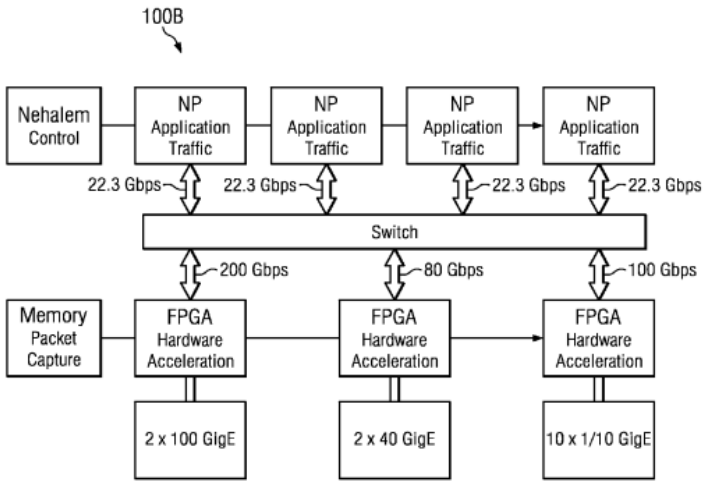
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 *FIG. 6B* (Stroud, FIG. 6B)

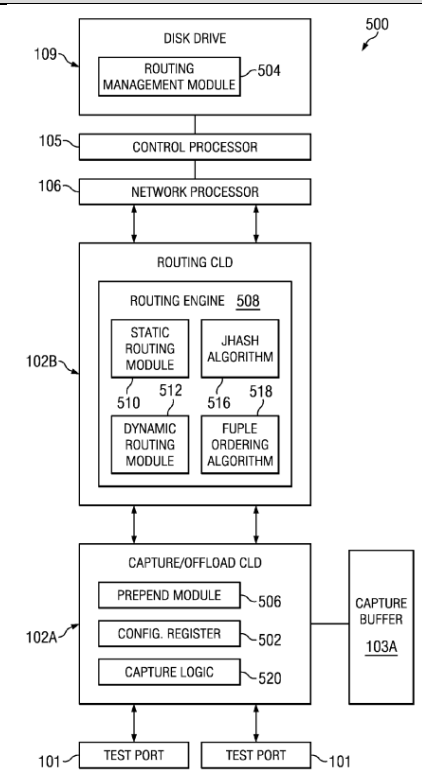
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 *FIG. 7A* (Stroud, FIG. 7A)

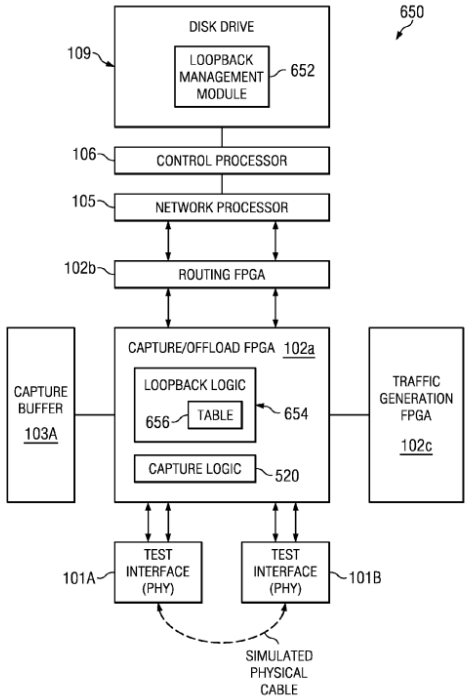
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 *FIG. 7B* (Stroud, FIG. 7B)

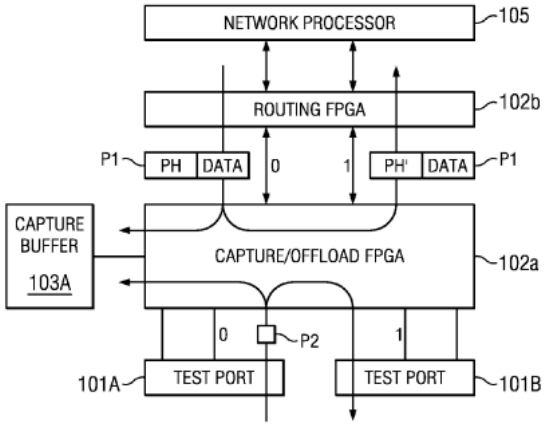
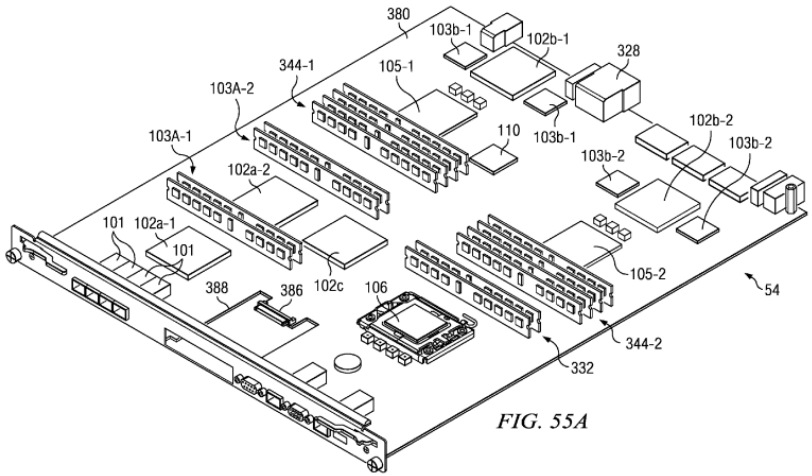
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 *FIG. 14A* (Stroud, FIG. 14A)

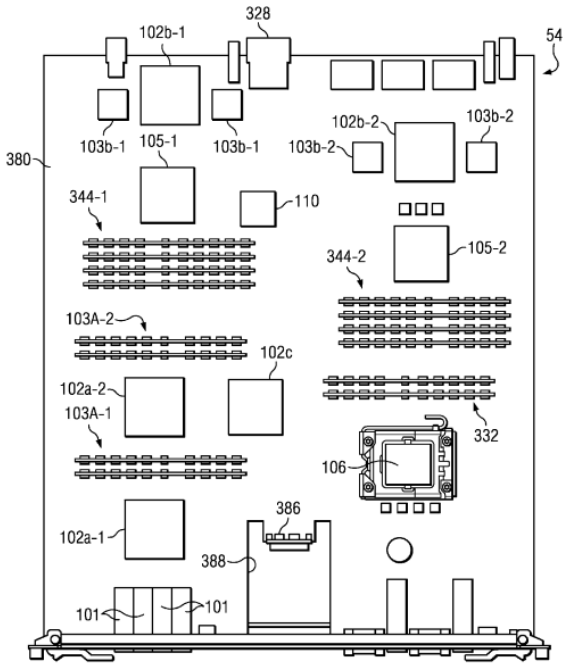
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 *FIG. 14B* (Stroud, FIG. 14B)

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 *FIG. 15* (Stroud, FIG. 15)

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 *FIG. 23* (Stroud, FIG. 23)

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 *FIG. 30* (Stroud, FIG. 30)

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 *FIG. 31* (Stroud, FIG. 31)  *FIG. 55A* (Stroud, FIG. 55A)

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 *FIG. 55B* (Stroud, FIG. 55B)

[1.c] one or more of the plurality of swappable pods or cards having a portion for user-definable hardware modules or user-	Stroud discloses or renders obvious “one or more of the plurality of swappable pods or cards having a portion for user-definable hardware modules or user-definable software modules[.]” *See, e.g.*: Devices 14 may include any type or types of network device, e.g., servers, routers, switches, gateways, firewalls, bridges, hubs, databases or data centers, workstations, desktop computers, wireless access points, wireless access devices, and/or any other type or types of devices

definable software modules;	configured to communicate with other network devices over a communications medium. Devices 14 may also include any hardware, software, and/or firmware components of any such network device, e.g., operating systems, applications, CPUs, configurable logic devices (CLDs), application-specific integrated circuits (ASICs), etc. (Stroud, [0083]) FIG. 4 is an high-level illustration of an example architecture 100A of a card 54 of network testing system 16, according to an example embodiment. As shown, example architecture 100A, referred to as a “testing and simulation architecture,” may include a controller 106, two network processors 105 and multiple CLDs 102 coupled to a management switch 110, and memory 103 coupled to the CLDs 102. (Stroud, [0103]) In general, network processors 105 are programmed to generate outbound network data in the form of one or more data packets and are programmed to receive and process inbound network data in the form of one or more data packets. In some embodiments, network processors 105 may be general purpose CPUs. In other embodiments, network processors 105 may be specialized CPUs with instruction sets and hardware optimized for processing network data. For example, network processors may be selected from the Netlogic XLR family of processors. (Stroud, [0105]) Configurable logic devices (CLDs) 102 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. (Stroud, [0106]) As used herein, the term configurable logic device (CLD) means a device that includes a set of programmable logic units, internal memory, and high-speed internal and external interconnections: Examples of CLDs include field programmable gate arrays (FPGAs) (e.g., ALTERA STRATIX family, XILINX VIRTEX family, as examples), programmable logic devices (PLDs), programmable array logic devices (PAL), and configurable programmable logic devices (CPLDs) (e.g., ALTERA MAXII, as an example). A CLD may include task-specific logic such as bus controllers, Ethernet media access controllers (MAC), and encryption/decryption modules. External interconnections on a CLD may include serial or parallel data lines or busses. External interconnections may be specialized to support a

	particular bus protocol or may be configurable, general-purpose I/O connections. Serial and parallel data connections may be implemented via specialized hardware or through configured logic blocks. (Stroud, [0107]) As used herein, the terms processor and CPU mean general purpose computing devices with fixed instruction sets or microinstruction sets such as x86 processors (e.g., the INTEL XEON family and the AMD OPTERON family, as examples only), POWERPC processors, and other well-known processor families. The terms processor and CPU may also include graphics processing units (GPUs) (e.g., NVIDIA GEFORCE family, as an example) and network processors (NPs) (e.g, NETLOGIC XLR and family, INTEL IXP family, CAVIUM OCTEON, for example). Processors and CPUs are generally distinguished from CLDs as defined above (e.g., FPGAs, CPLDs, etc.) Some hybrid devices include blocks of configurable logic and general purpose CPU cores (e.g., XILINX VIRTEX family, as an example) and are considered CLDs for the purposes of this disclosure. (Stroud, [0110]) An CLD (Field Programmable Gate Array) is a reprogrammable device that can be modified to simulate many types of hardware. Being reprogrammable, it can be continually expanded to offer new acceleration and network analysis functionality with firmware updates. Example testing and simulation architecture 100A includes various CLDs designated to perform different functions, including two “capture and offload CLDs” 102A capturing data packets, two “router CLDs” 102B for routing data between components of architecture 100A, and a traffic generation CLD 102C for generating traffic that is delivered to the test system 18. (Stroud, [0117]) Each capture and offload CLDs 102A is connected to one or more test interfaces 101. Thus, CLDs 102A are the first and last device in the packet-processing pipeline. In some embodiments, Ethernet MACs (Media Access Controllers) required to support 10/100/1000 and 10000 Mbps Ethernet standards are implemented within CLDs 102A and interact with the physical-layer devices (PHYs) that implement with the test interfaces 101. (Stroud, [0119]) Each capture and offload CLDs 102A is connected to the traffic generation CLD 102C. In this capacity, the CLDs 102A is a pass-through interface; packets sent by the traffic generation

	CLD 102C are forwarded directly to an Ethernet test interface 101 for delivery to the test system 18. (Stroud, [0121]) 1. The primary TCP/IP stack used for network traffic simulation executes on the network processor 105. (Stroud, [0149]) 2. IP and Ethernet-layer address allocation and routing protocols are handled by the network processor 105. (Stroud, [0150]) 3. User and application-layer simulation also run on the network processor 105. (*See e.g.,* Stroud, [0151]) 4. The network processor 105 works with software on the controller 106 to collect statistics, which may subsequently be used by the statistics and reporting engine 162 of subsystem 28. (*See e.g.,* Stroud, [0152]) 5. The network processor 105 may also collect statistics from CLDs 102A, 102B, and 102C and report them to the controller 106. In an alternative embodiment, the controller 106 itself is configured to collect statistics directly from CLDs 102A, 102B, and 102C. (Stroud, [0153])

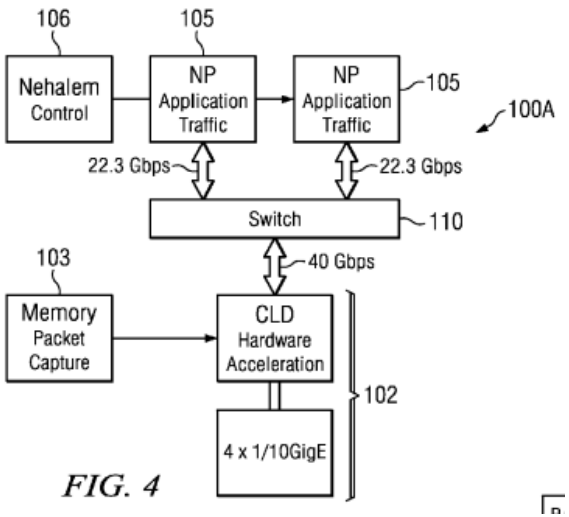


FIG. 4

(Stroud, FIG. 4)

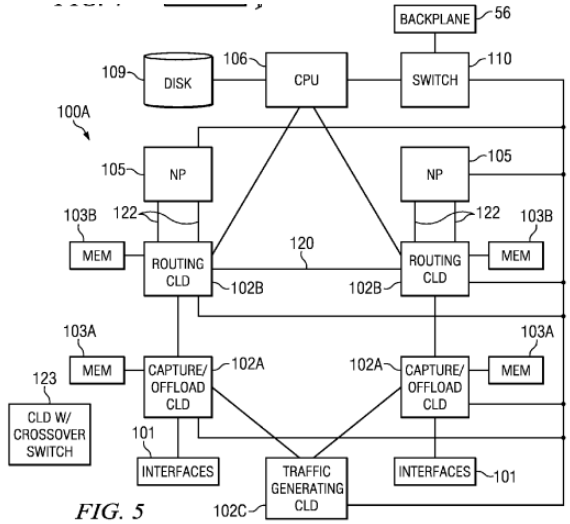
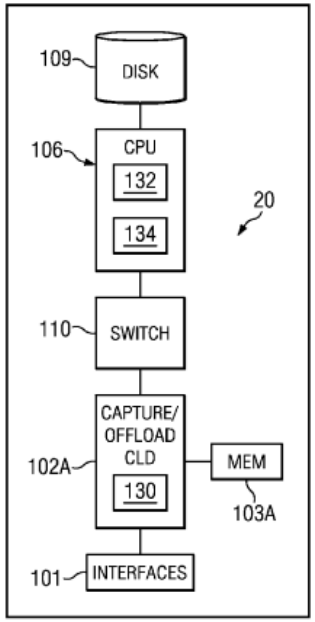
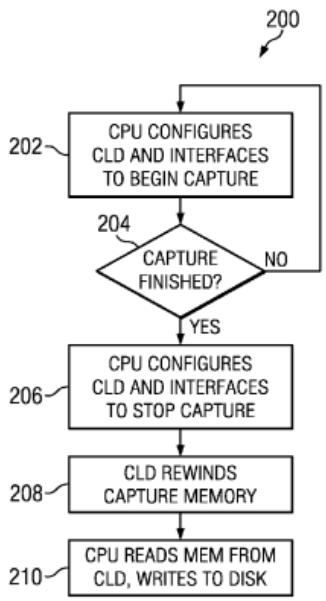
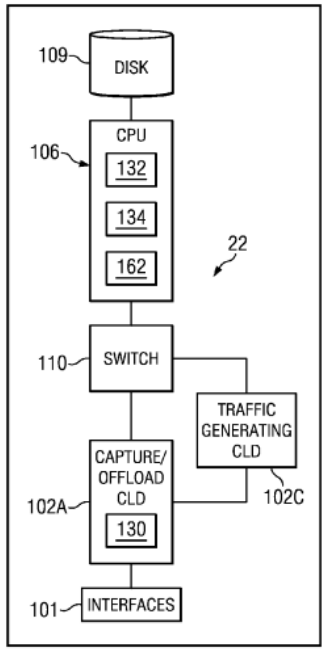


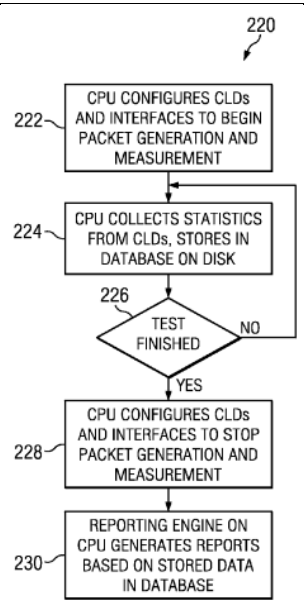
FIG. 5

(Stroud, FIG. 5)

	 *FIG. 6A* (Stroud, FIG. 6A)

	 *FIG. 6B* (Stroud, FIG. 6B)

	 *FIG. 7A* (Stroud, FIG. 7A)

	 *FIG. 7B* (Stroud, FIG. 7B)

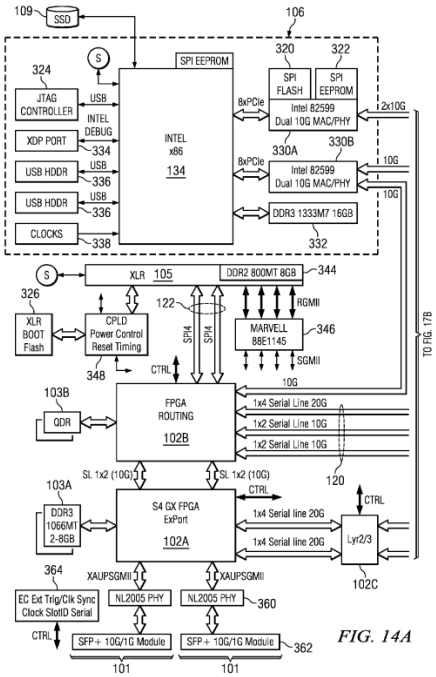
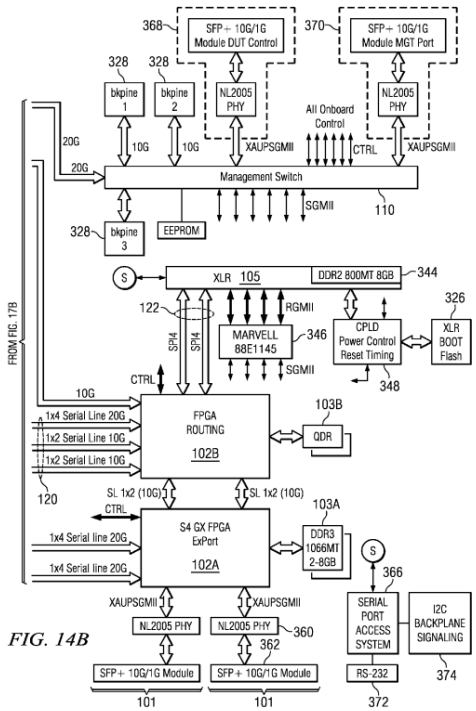


FIG. 14A

(Stroud, FIG. 14A)



(Stroud, FIG. 14B)

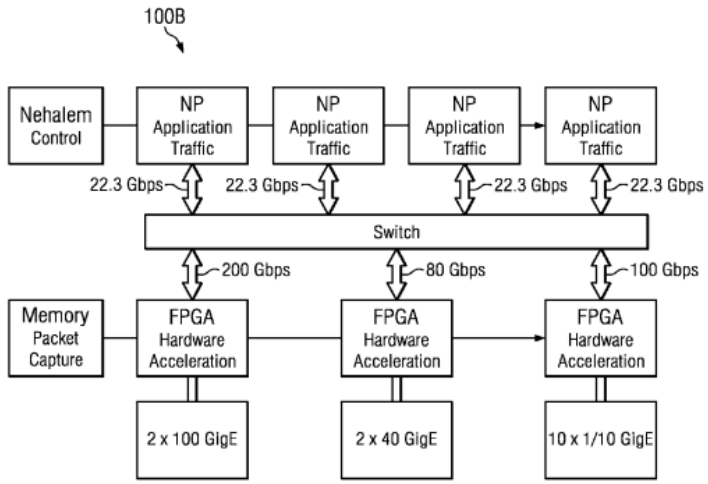


FIG. 15

(Stroud, FIG. 15)

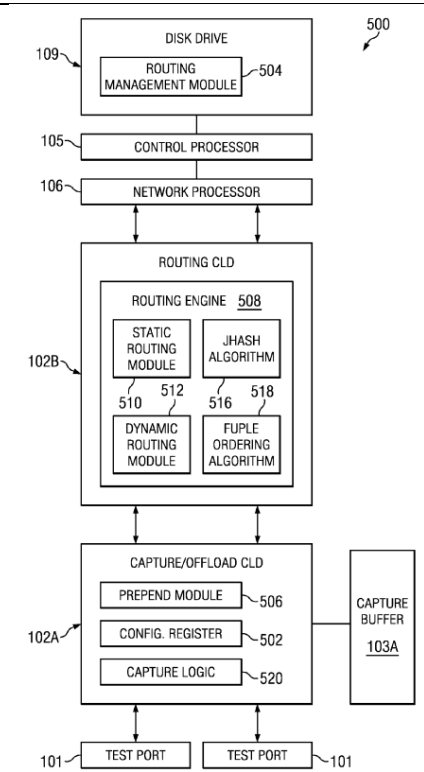


FIG. 23

(Stroud, FIG. 23)

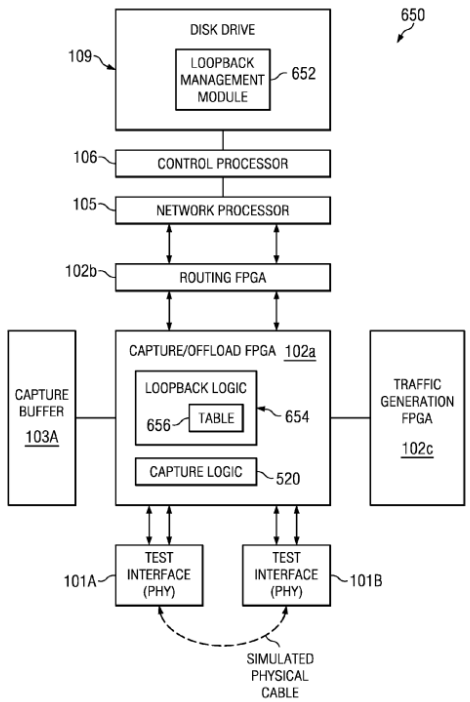


FIG. 30

(Stroud, FIG. 30)

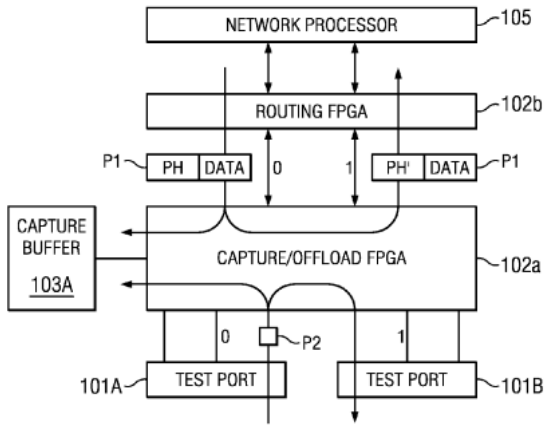


FIG. 31

(Stroud, FIG. 31)

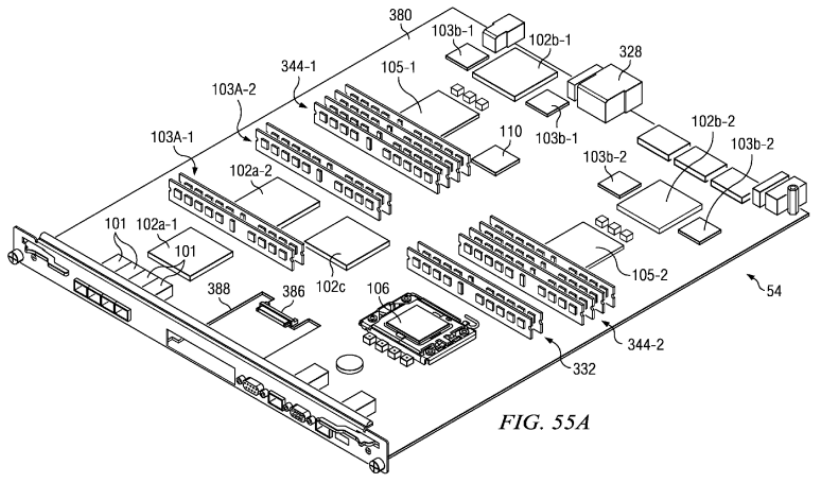
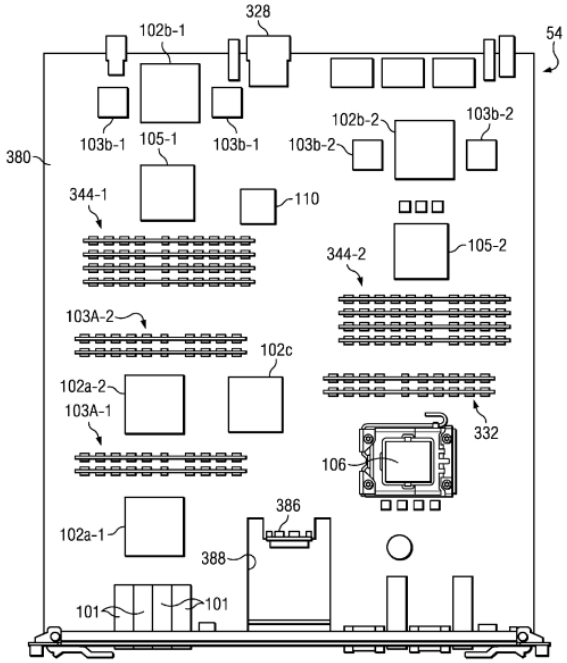


FIG. 55A

(Stroud, FIG. 55A)

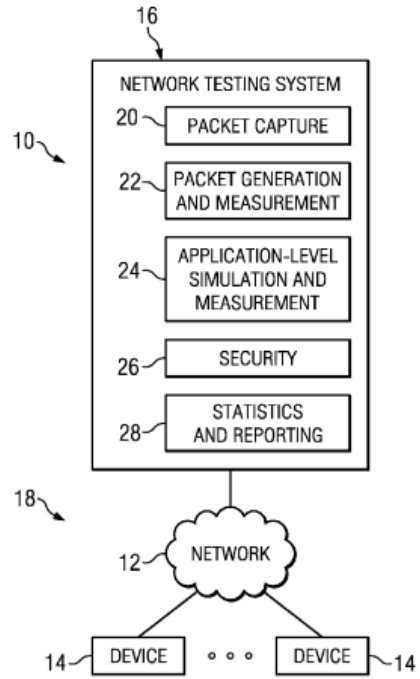
	 *FIG. 55B* (Stroud, FIG. 55B)
[1.d] the plurality of swappable pods or cards being user-configurable to implement data flow processing architectures; and	Stroud discloses or renders obvious “the plurality of swappable pods or cards being user-configurable to implement data flow processing architectures[.]” *See, e.g.*: FIG. 1 illustrates a general block diagram of an arrangement 10 for testing the performance of a communications network 12 and/or one or more network devices 14 using a network testing system 16, according to certain embodiments of the present disclosure. Test devices 14 may be part of a network 12 tested by network testing system 16, or may be connected to network testing system 16 by network 12. Thus, network testing system 16 may be configured for testing network 12 and/or devices 14 within or connected to network 12. For the sake of simplicity, the test network 12 and/or devices 14 are referred to herein as the test system 18. Thus, a test system 18 may comprise a network 12, one or more devices 14 within a network 12 or coupled to a network 12, one or more hardware, software, and/or firmware components

	of device(s) 14, or any other component or aspect of a network or network device. (Stroud, [0080]) Network testing system 16 may be configured to test the performance (e.g., traffic-handling performance) of devices 14, the security of a test system 18 (e.g., from security attacks), or both the performance and security of a test system 18. In some embodiments, network testing system 16 configured to simulate a realistic combination of business, recreational, malicious, and proprietary application traffic at sufficient speeds to test both performance and security together using the same data and tests. In some embodiments, network testing system 16 is configured for testing content-aware systems 18 devices 14 and/or content-unaware systems 18. (Stroud, [0081]) Devices 14 may include any type or types of network device, e.g., servers, routers, switches, gateways, firewalls, bridges, hubs, databases or data centers, workstations, desktop computers, wireless access points, wireless access devices, and/or any other type or types of devices configured to communicate with other network devices over a communications medium. Devices 14 may also include any hardware, software, and/or firmware components of any such network device, e.g., operating systems, applications, CPUs, configurable logic devices (CLDs), application-specific integrated circuits (ASICs), etc. (Stroud, [0083]) In some embodiments, network testing system 16 is configured to model and simulate network traffic. The network testing system 16 may act as virtual infrastructure and simulate traffic behavior of network devices (e.g., database server, Web server) running a specific application. The resulting network traffic originated from the network testing system 16 may drive the operation of a test system 18 for evaluating the performance and/or security of the system 18. Complex models can be built on realistic applications such that a system 18 can be tested and evaluated under realistic conditions, but in a testing environment. Simultaneously, network testing system 16 may monitor the performance and/or security of a test system 18 and may collect various metrics that measure performance and/or security characteristics of system 18. (Stroud, [0084])

	For example, as shown in FIG. 1, network testing system 16 may comprise any or all of the following testing and simulation subsystems: a high-speed, high-resolution network packet capture subsystem 20, a high-speed packet generation and measurement subsystem 22, an application-level simulation and measurement subsystem 24, a security and exploit simulation and analysis subsystem 26, and/or a statistics collection and reporting subsystem 28. Subsystems 20-28 are discussed below in greater detail. In some embodiments, the architecture of network testing system 16 may allow for some or all of subsystems 20-28 to operate simultaneously and cooperatively within the same software and hardware platform. Thus, in some embodiments, system 16 is configured to generate and analyze packets at line rate, while simultaneously capturing that same traffic, performing application simulation, and security testing. In particular embodiments, system 16 comprises custom hardware and software arranged and programmed to deliver performance and measurement abilities not achievable with conventional software or hardware solutions. (Stroud, [0086]) In general, network processors 105 are programmed to generate outbound network data in the form of one or more data packets and are programmed to receive and process inbound network data in the form of one or more data packets. In some embodiments, network processors 105 may be general purpose CPUs. In other embodiments, network processors 105 may be specialized CPUs with instruction sets and hardware optimized for processing network data. For example, network processors may be selected from the Netlogic XLR family of processors. (Stroud, [0105]) Configurable logic devices (CLDs) 102 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. (Stroud, [0106]) As used herein, the term configurable logic device (CLD) means a device that includes a set of programmable logic units, internal memory, and high-speed internal and external interconnections: Examples of CLDs include field programmable gate arrays (FPGAs) (e.g., ALTERA STRATIX family, XILINX VIRTEX family, as examples), programmable logic devices (PLDs), programmable array logic devices (PAL), and configurable programmable logic devices (CPLDs) (e.g., ALTERA MAXII, as an example). A CLD may include task-

	specific logic such as bus controllers, Ethernet media access controllers (MAC), and encryption/decryption modules. External interconnections on a CLD may include serial or parallel data lines or busses. External interconnections may be specialized to support a particular bus protocol or may be configurable, general-purpose I/O connections. Serial and parallel data connections may be implemented via specialized hardware or through configured logic blocks. (Stroud, [0107]) A configurable logic device may be configured, or programmed, at different times. In some circumstances, a configurable logic device may be programmed at the time of manufacture (of the configurable logic device or of a device containing the configurable logic device). This manufacture-time programming may be performed by applying a mask to the device and energizing a light or other electromagnetic wave form to permanently or semi-permanently program the device. A configurable logic device may also be programmed electronically at manufacture time, initialization time, or dynamically. Electronic programming involves loading configuration information from a memory or over an input/output connection. Some configurable logic devices may include onboard non-volatile memory (e.g., flash memory) for storing configuration information. Such an arrangement allows the configurable logic device to program itself automatically when power is applied. (Stroud, [0109]) An CLD (Field Programmable Gate Array) is a reprogrammable device that can be modified to simulate many types of hardware. Being reprogrammable, it can be continually expanded to offer new acceleration and network analysis functionality with firmware updates. Example testing and simulation architecture 100A includes various CLDs designated to perform different functions, including two “capture and offload CLDs” 102A capturing data packets, two “router CLDs” 102B for routing data between components of architecture 100A, and a traffic generation CLD 102C for generating traffic that is delivered to the test system 18. (Stroud, [0117]) Each capture and offload CLDs 102A is connected to one or more test interfaces 101. Thus, CLDs 102A are the first and last device in the packet-processing pipeline. In some embodiments, Ethernet MACs (Media Access Controllers) required to support 10/100/1000

	and 10000 Mbps Ethernet standards are implemented within CLDs 102A and interact with the physical-layer devices (PHYs) that implement with the test interfaces 101. (Stroud, [0119]) Each capture and offload CLDs 102A is connected to the traffic generation CLD 102C. In this capacity, the CLDs 102A is a pass-through interface; packets sent by the traffic generation CLD 102C are forwarded directly to an Ethernet test interface 101 for delivery to the test system 18. (Stroud, [0121]) 1. The primary TCP/IP stack used for network traffic simulation executes on the network processor 105. (Stroud, [0149]) 2. IP and Ethernet-layer address allocation and routing protocols are handled by the network processor 105. (Stroud, [0150]) 3. User and application-layer simulation also run on the network processor 105. (Stroud, [0151]) 4. The network processor 105 works with software on the controller 106 to collect statistics, which may subsequently be used by the statistics and reporting engine 162 of subsystem 28. (Stroud, [0152]) 5. The network processor 105 may also collect statistics from CLDs 102A, 102B, and 102C and report them to the controller 106. In an alternative embodiment, the controller 106 itself is configured to collect statistics directly from CLDs 102A, 102B, and 102C. (Stroud, [0153]) Thus, in some embodiments (e.g., embodiments that support hot swapping of cards 54), to utilize multiple control processors 105 and drives 109 available in a multi-card system 16, as well as to allow for each control processor 106 to run an independent operating system, while maintaining Ethernet connectivity to the backplane 56, system 16 may be configured such that local network processors 105 boot from the local control processor 106 using DHCP, NFS

	(Network File System), and TFTP (Trivial File Transfer Protocol). This task is divided by a special dynamic configuration for the DHCP server. (Stroud, [0378])  *FIG. 1* (Stroud, FIG. 1)

[1.e] a network coupled to the one or more swappable pods or cards and supporting	Stroud discloses or renders obvious “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[.]”

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.	Additionally, Plaintiffs have cited the use of Internet Protocol (IP) packets as allegedly satisfying limitation [1.e]. *See* Compl. at 55; ’753 Infringement Contentions at 44-45. Stroud discloses or renders obvious claim limitation [1.e] under Plaintiffs’ apparent interpretation of the claim. *See, e.g.*: FIG. 3 illustrates an example configuration of a network testing system 16, according to example embodiments. Network testing system 16 may include a chassis 50 including any suitable number of slots 52, each configured to receive a modular card, or blade, 54. A card or blade 54 may comprise one or more printed circuit boards (e.g., PCB 380 discussed below). For example, as shown, chassis 50 may include Slot 0 configured to receive Card 0, Slot 1 configured to receive Card 1, . . . and Slot n configured to receive Card n, where n equals any suitable number, e.g., 1, 2, 3, 4, 5, 7, or more. For example, in some embodiments, chassis 50 is a 3-slot chassis, a 5-slot chassis, or a 12-slot chassis. In other embodiments, system 16 comprises a single card 54. (Stroud, [0098]) Each card 54 may be plugged into a backplane 56, which may include physical connections 60 for communicatively connecting cards 54 to each other, as discussed below. While cards may be interconnected, each card is treated for some purposes as an independent unit. Communications within a card are considered to be “local” communications. Two different cards attached to the same backplane may be running different versions of software so long as the versions are compatible. (Stroud, [0099]) Each card 54 may include any architecture 100 of hardware, software, and/or firmware components for providing the functionality of network testing system 16. For example, card 0 may include an architecture 100 a, card 1 may include an architecture 100 b, . . . , and card n may include an architecture 100 n. The architecture 100 of each card 54 may be the same as or different than the architecture 100 of each other card 54, e.g., in terms of hardware, software, and/or firmware, and arrangement thereof. (Stroud, [0100])

	Each architecture 100 may include a system controller, one or more network processors, and one or more CLDs connected to a management switch 110 (and any other suitable components, e.g., memory devices, communication interfaces, etc.). Cards 54 may be communicatively coupled to each other via the backplane 56 and management switches 110 of the respective cards 54, as shown in FIG. 3. In some embodiments, backplane 56 include physical connections for connecting each card 54 directly to each other card 54. Thus, each card 54 may communicate with each other card 54 via the management switches 110 of the respective cards 54, regardless of whether one or more slots 52 are empty or whether one or more cards 54 are removed. (Stroud, [0101]) In some embodiments, each card 54 may be configured to operate by itself, or cooperatively with one or more other cards 54, to provide any of the functionality discussed herein. (Stroud, [0102]) Management switch 110 allows and manages communications among the various components of testing architecture 100A, as well as communications between components of testing architecture 100A and components of one or more other cards 54 (e.g., via backplane 56 as discussed above with respect to FIG. 3). Management switch 110 may be a Ethernet layer 2 multi-port switch. (Stroud, [0113]) As discussed above, the components of example architecture 100A may be provided on a single blade 54, and multiple blades 54 may be connected together via backplane 54 to create larger systems. The various components of example architecture 100A are now discussed, according to example embodiments. (Stroud, [0115]) Each capture and offload CLDs 102A may be programmed to implement the following functionality for packets received from test interfaces 101. (Stroud, [0124])

	For example, packets may include instructions for any one or more of the following offload operations: (a) insert a timestamp into the packet, (b) calculate checksums for the packet on up to 2 layers of IP and TCP/UDP/ICMP headers, and/or (c) split the packet into smaller TCP segments via TCP segmentation offload. Further, a capture and offload CLD 102A may forward a copy of each packet (or particular packets) for storage in the capture memory 103B attached to the CDL 102A, along with a timestamp indicating when each packet was sent. (Stroud, [0125]) In some embodiments, for packets sent from a network processor 105 or controller 106, the sending processor 105, 106 first specifies a target address in a special internal header in each packet.” Discussing the function of router CLDs in the invention. “The router CLD 102B may use the target address to determine where to send the packet next, e.g., it may direct the packet to the another router CLD 102B or to the nearest capture and offload CLD 102A. (Stroud, [0135]) The management switch 110 connects to every CLD 102, network processor 105, and control CPU 106 in the system 100A. In some embodiments, management switch 110 comprises a management Ethernet switch configured to allow communication of for 1-10 Gbit traffic both between blades 54 and between the various processors 105, 106 and CLDs 102 on each particular blade 54. Management switch 110 may route packets based on the MAC address included in each packet passing through switch 110. Thus, management 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. In such embodiment, all subsystems are controllable via Ethernet, such that additional processors and CLDs may be added by simply chaining management switches 110 together. (Stroud, [0174]) Network testing system 16 may be configured to support any suitable number of cards or blades 54. In one embodiment, system 16 is configured to support between 1 and 14 cards 54 in a single chassis 50. Backplane 56 may provide a system for interconnecting the

	management Ethernet provided by the management switches 110 of multiple cards 54, as well as system monitoring connections for measuring voltages and temperatures on cards 54, and for debugging and monitoring CPU status on all cards 54, for example. Backplane 56 may also distribute clock signals between all cards 54 in a chassis 50 so that the time stamps for all CPUs and CLDs remain synchronized. (Stroud, [0176]) Packets contain information about the source and destination address of the nodes, application information. A network packet capture is the observing and storage of packets on the network for later debugging analysis. (Stroud, [0179]) The CLD implementation of system 16 can be reconfigured to parse packets two headers deep. If all traffic has a single outer header, e.g., tunneled traffic, the system can look further to find unique identifiers in the packets. (Stroud, [0246]) FIG. 16 illustrates various sub-systems configured to provide various functions associated with system 16 as discussed herein. For example, control system 450 may include any or all of the following sub-systems: An Ethernet-based management system; a distributed DHCP, Addressing and Startup management system; a CLD-based packet routing system; a processor-specific routing system; a CLD pipeline system; a bandwidth management system; a packet capture error tracking system; an efficient packet capture system; a data loopback and capture system; a CLD-based hash function system; multi-key hash tables; a packet assembly subsystem; a packet segmentation offload system; an address compression system;

	a task management engine; a dynamic latency analysis system; a serial port access system; a USB device initialization system; a USB programming system; and a JTAG programming system. (Stroud, [0322]-[0342]) FIG. 17 illustrates the layout of the Ethernet packets containing CLD control messages according to certain embodiments of the present disclosure. The first portion of the packet is the IEEE standard header for Ethernet packets, including the destination MAC address, the source MAC address, and the Ethernet packet type field. The type field is set to value unused the IEEE standard to avoid conflicts with existing network protocols, especially within the networking stack on the control CPU. Immediately following the standard header is an access directive format including a sequence identifier, a count, a command field, and data to be used in executing the directive. The sequence number is an identifier used by the originator of the directive for tracking completion and/or timeout of individual directives. The count specifies the number of registers accessed by the command and the command field specifies the type of directive. (Stroud, [0349]) In some embodiments, the control CPU 106 of each blade 54 is the only component of system 16 with connectivity to external networks and is thus the public/external Ethernet interface of control CPU 106 is only component of system 16 that is assigned a globally unique “public” MAC address. Hardware and software of system 16 dynamically assigns each other Ethernet device in system 16, (including each network processor 105, each CLD 102, and local/internal Ethernet interfaces of control CPU 106) a MAC address that is unique within system 16, but need not be globally unique, as the internal Ethernet network of system 16 does not connect with external networks. In some embodiments, each of such Ethernet devices is dynamically assigned a unique MAC address based on a set of characteristics regarding that device and its location within the configuration of system 16. For example, in some embodiments, each network processor 105 and CLD in system 16 automatically derives a 6-byte MAC address for itself that has the following format:

	1st Byte: fixed (indicates a non-global MAC address). 2nd Byte: indicates chip type: e.g., processor, CLD, or other type of device. 3rd Byte: indicates processor type or model, or CLD type or model: e.g., 20 G, 10 G, or 1 G processor, router CLD, capture/offload CLD, etc. 4th Byte: indicates slot number. 5th Byte: indicates processor or CLD number, e.g., to distinguish between multiple instances of the same type of processor or CLD on the same card (e.g., two network processors 105 or two capture/offload CLDs 102 a). 6th Byte: indicates processor interface (each interface to the management switch has its own MAC address). (Stroud, [0366]) Thus, network processors 105 and CLDs 102 may generate their own MAC addresses according to a suitable algorithm. The MAC address for each device 102, 105, and 106 on a particular card 54 may identify the chassis slot 52 in which that card 54 is located, as well as other identifying information. In some embodiments, management switch 110 has no CPU and runs semi-independently. In particular, management switch 110 may have no assigned MAC address, and may rely on control CPU 106 for intelligence. (Stroud, [0381]) At step 532, a packet (e.g., part of a data stream from test network 18) is received at system 16 on a test interface 101 and forwarded to capture/offload CLD 102A via a physical interface. At step 534, prepend module 506 attaches a prepend header to the received packet. The prepend header may include one or more header fields that are presently populated, including a timestamp indicating the arrival time of the packet, and one or more header fields that may be populated later, e.g., a hash value to be subsequently populated by routing module 508 in routing CLD 102B, as discussed below. (Stroud, [0406]) The prepend header added by capture/offload CLD 102A to ingress packets arriving at interface 101 for deliver to a network processor may contain the following information according to certain embodiments of the present disclosure. (Stroud, [0414])

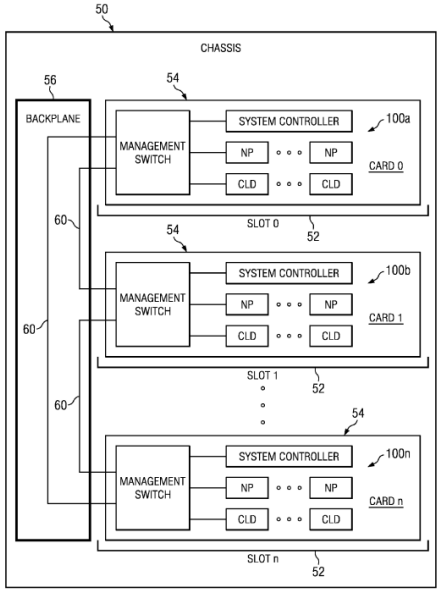
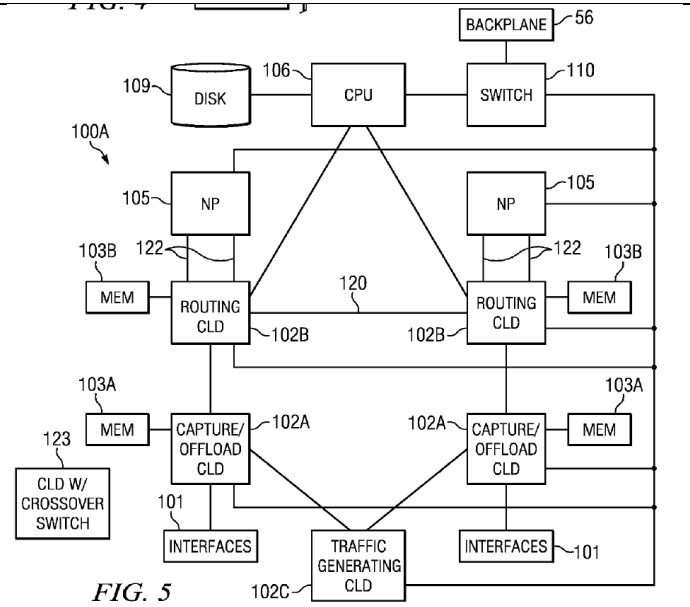
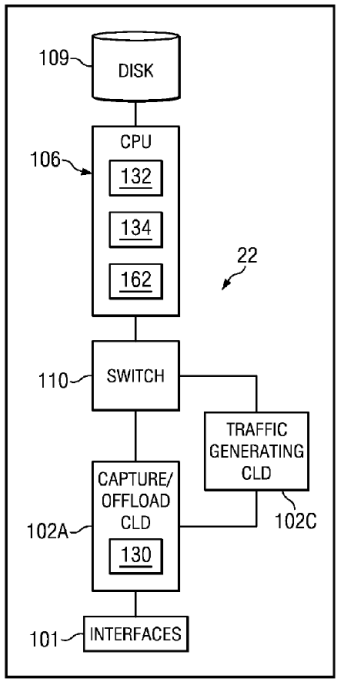


FIG. 3

(Stroud, FIG. 3)

	 *FIG. 5* (Stroud, FIG. 5)

	 *FIG. 7A* (Stroud, FIG. 7A)

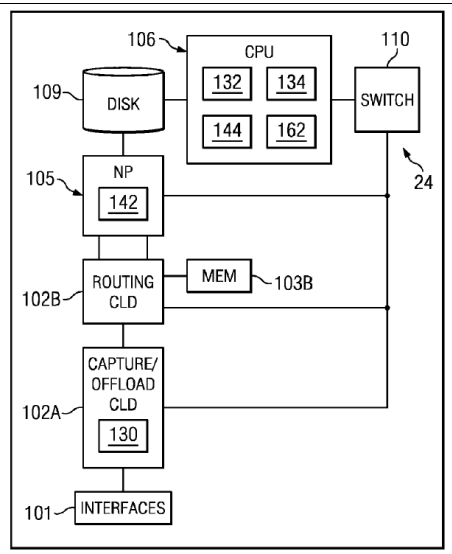


FIG. 8A

(Stroud, FIG. 8A)

IEEE STANDARD HEADER

DEST MAC, 6 BYTES	SRC MAC, 6 BYTES	EthType, 2 BYTES

CLD ACCESS DIRECTIVE FORMAT

SEQUENCE 2 BYTES	COUNT 2 BYTES	CMD 2 BYTES	DATA...

FIG. 17

(Stroud, FIG. 17)

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 discloses or renders obvious “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.” Additionally, Plaintiffs allege infringement of claim 2 based on multiple cards being deployed in a server. ’753 Infringement Contentions at 52 (“multiple AMD Pensando DSCs can be deployed within a single server and these AMD Pensando DSCs are connected to each other without using the module messaging interface network”). Stroud discloses or renders obvious claim 2 under Plaintiffs’ apparent interpretation of the claim. *See, e.g.*: Each card 54 may be plugged into a backplane 56, which may include physical connections 60 for communicatively connecting cards 54 to each other, as discussed below. While cards may be interconnected, each card is treated for some purposes as an independent unit. Communications within a card are considered to be “local” communications. Two different cards attached to the same backplane may be running different versions of software so long as the versions are compatible. (Stroud, [0099]) Each card 54 may include any architecture 100 of hardware, software, and/or firmware components for providing the functionality of network testing system 16. For example, card 0 may include an architecture 100 a, card 1 may include an architecture 100 b, . . . , and card n may include an architecture 100 n. The architecture 100 of each card 54 may be the same as or different than the architecture 100 of each other card 54, e.g., in terms of hardware, software, and/or firmware, and arrangement thereof. (Stroud, [0100]) Network testing system 16 may be configured to support any suitable number of cards or blades 54. In one embodiment, system 16 is configured to support between 1 and 14 cards 54 in a single chassis 50. Backplane 56 may provide a system for interconnecting the management Ethernet provided by the management switches 110 of multiple cards 54, as well as system monitoring connections for measuring voltages and temperatures on cards 54, and for debugging and monitoring CPU status on all cards 54, for example. Backplane 56 may also distribute clock signals

	between all cards 54 in a chassis 50 so that the time stamps for all CPUs and CLDs remain synchronized. (Stroud, [0176]) Serial port access system 366 provides direct data and/or control access to various system components via controller 106 or an external serial port 372, e.g., a physical RS-232 port on the front of the blade. Serial port access system 366 (illustrated in detail in FIG. 46 and discussed below) connects via serial line (illustrated in FIGS. 14A and 14B as an S in a circle) to each of: control processor 106, each network processor 105, external serial port 372, and an I2C backplane signaling system 374. As discussed below with respect to FIG. 46 I2C backplane signaling system 374 may be provided for managing inter-card serial connections, and may include a management microcontroller (or “environmental controller”) 954, I2C connection 958 to backplane 56, and an I2C IO expander 956. Serial lines may be multipoint low-voltage differential signaling (MLVDS). (Stroud, [0320]) FIG. 5 illustrates an internal network configuration for certain embodiments of the present disclosure. In FIG. 5, Ethernet switch 110 connects to CPU 105 and both NPs 105. In addition, Ethernet switch 110 connects to routing CLDs 102B, capture/offload CLDs 102A, and traffic generating CLD 102C. In this configuration, any CPU may communicate with any CLD directly using Ethernet packets. Ethernet switch 110 also connects to backplane 56 to extend connectivity to CPUs or CLDs on other boards. The approach of the present disclosure could also facilitate direct communication between any of the attached devices including CLDs, network processors, and control processors. (Stroud, [0353]) As discussed above, system 16 may be housed in a chassis 50 that interconnects multiple cards 54 via a backplane 56. In some embodiments, all cards 54 boot a single software image. In other embodiments, each card 54 runs a different software image, possibly with different revisions, in the same chassis 50. (Stroud, [0376]) One challenge results from the fact that the cards 54 in chassis 50 are physically connected to each other via Ethernet over the backplane 56. In addition, some processors in system 16 may obtain their operating system image from other processors across the shared Ethernet using DHCP. DHCP is a broadcast protocol, such that a request from any processor on any card 54 can be seen from any other card 54. Thus, without an effective measure to prevent it, any processor can boot from

any other processor that replies to its DHCP request quickly enough, including processors on other cards 54 from the requesting processor. This may be problematic in certain embodiments, e.g., embodiments that support hot swapping of cards 54. For example, if a CPU on card 1 boots from a CPU on card 2, and card 2 is subsequently removed from chassis 50, CPU 1 may crash. (Stroud, [0377])

Thus, in some embodiments (e.g., embodiments that support hot swapping of cards 54), to utilize multiple control processors 105 and drives 109 available in a multi-card system 16, as well as to allow for each control processor 106 to run an independent operating system, while maintaining Ethernet connectivity to the backplane 56, system 16 may be configured such that local network processors 105 boot from the local control processor 106 using DHCP, NFS (Network File System), and TFTP (Trivial File Transfer Protocol). This task is divided by a special dynamic configuration for the DHCP server. (Stroud, [0378])

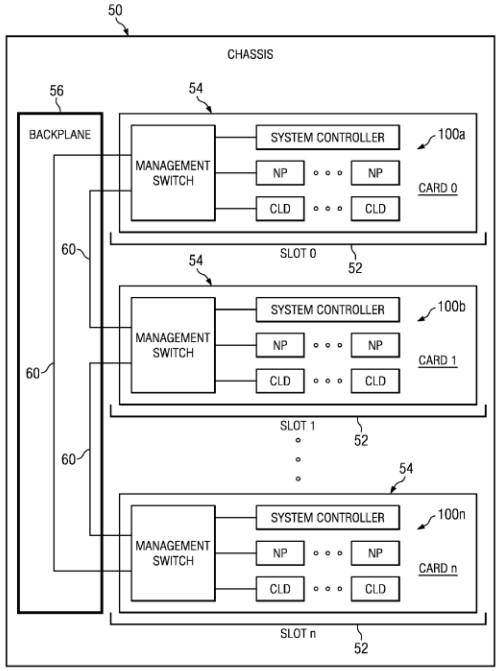
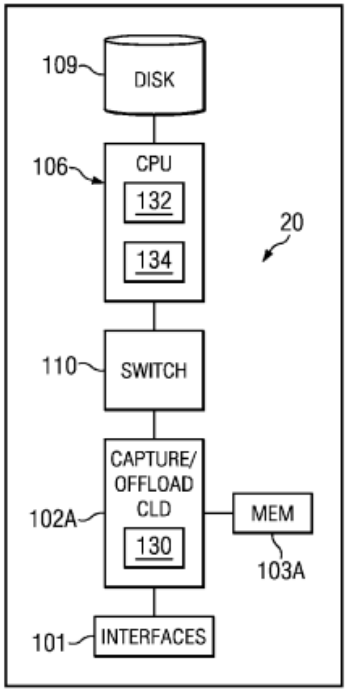


FIG. 3

	(Stroud, FIG. 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 discloses or renders obvious “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,” as shown in at least the exemplary citations below. Additionally, Plaintiffs cited the presence of processors that allegedly “work in cooperation with a sandbox” as allegedly satisfying claim 3. ’753 Infringement Contentions at 53. Stroud discloses or renders obvious claim 3 under Plaintiffs’ apparent interpretation of the claim. *See, e.g.*: Each card 54 may be plugged into a backplane 56, which may include physical connections 60 for communicatively connecting cards 54 to each other, as discussed below. While cards may be interconnected, each card is treated for some purposes as an independent unit. Communications within a card are considered to be “local” communications. Two different cards attached to the same backplane may be running different versions of software so long as the versions are compatible. (Stroud, [0099]) (e) Controller software 132 of controller 106 configured to start, stop and post-process packet captures. (Stroud, [0187]) In some embodiments, CLDs in the present disclosure contain embedded Ethernet controllers designed to parse incoming specially formatted packets as command directives for memory access to be executed. In this approach, the CLD directly interprets the incoming packets to make the access to internal CLD memory without intervention by an intermediate CPU or microcontroller processing the Ethernet packets. Simultaneous requests from multiple originating packet sources (e.g., CPUs) are supported through the use of a command FIFO that queues up incoming requests. After each command directive is completed by the CLD, a response packet is sent back to the originating source CPU containing the status of the operation. (Stroud, [0346])

	In certain embodiments, controller software 132 includes a software module called an CLD server. The CLD server provides a centralized mechanism for tracking failures and timeouts of Ethernet commands. The CLD server may be implemented as an operating system level driver that implements a raw socket. This raw socket is configured as a handler for Ethernet packets of the type created to implement the CLD control protocol. All other Ethernet packets left for handling by the controller's networking stack or other raw sockets. (Stroud, [0356]) At step 474, the network processor sends the directive packet to control CPU 106 via switch 110. The directive packet is received by the CLD server through a raw port on the network driver of the control server. The CLD server creates a record of the directive packet and includes in that record the current time and at least the source MAC address and the sequence number of the directive packet. The CLD server modifies the directive packet as follows. The source MAC address is set to the MAC address of control CPU 106 and the destination MAC address is set to the MAC address of traffic generating CLD 102C. In some embodiments, the CLD server replaces the sequence number with its own current sequence number. In some embodiments, the CLD server may keep a copy of the entire modified directive packet to allow later retransmission. (Stroud, [0358]) At step 476, the CLD server transmits the modified directive packet, via switch 110, to traffic generating CLD 102C for execution. (Stroud, [0359]) At a regular interval, the CLD server examines its records of previously sent directives to and determines whether any are older than a predetermined age threshold. This might indicate that a response from the destination CLD is unlikely due to an error in transmission or execution of the directive. If any directives are older than the threshold, then a timeout is recognized at step 478. (Stroud, [0360]) In the case of a timeout, the CLD server generates an error message at step 480 to send to the requesting network processor. In some embodiments, CLD server may resend the directive one or more times before giving up and reporting an error. The CLD server also deletes the record of the directive packet at this time. (Stroud, [0361])

	If a response is received prior to a timeout, CLD server removes the directive packet record and forwards the CLD response packet to the originating network processor at step 482. To forward the CLD response packet, the CLD server replaces the destination MAC address with the MAC address of the originating network processor. If the sequence number was replaced by the CLD server in step 474, the original sequence number may be restored. Finally the modified response packet is transmitted, via switch 110, to the originating network processor. (Stroud, [0362])  *FIG. 6A* (Stroud, FIG. 6A)

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.	Stroud discloses or renders obvious “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,” as shown in at least the exemplary citations below. Additionally, Plaintiffs allege infringement of claim 4 by citing support for “various types of memory to optimize performance.” ’753 Infringement Contentions at 53. Stroud discloses or renders obvious claim 4 under Plaintiffs’ apparent interpretation of the claim. *See, e.g.*: FIG. 4 is an high-level illustration of an example architecture 100A of a card 54 of network testing system 16, according to an example embodiment. As shown, example architecture 100A, referred to as a “testing and simulation architecture,” may include a controller 106, two network processors 105 and multiple CLDs 102 coupled to a management switch 110, and memory 103 coupled to the CLDs 102. (Stroud, [0103]) FIG. 5 is a more detailed illustration of the example testing and simulation architecture 100A shown in FIG. 4, according to an example embodiment. As shown, example testing and simulation architecture 100A includes controller 106; memory 109 coupled to controller 106; two network processors 105; various CLDs 102 (e.g., capture and offload CLDs 102A, router CLDs 102B, and a traffic generation CLD 102C); memory devices 103A and 103B coupled to CLDs 102A and 102B; respectively; management switch 110 coupled to network processors 105 and CLDs 102A, 102B, and 102C, as well as to backplane 56 (e.g., for connection to other cards 54); test interfaces 101 for connecting testing architecture 100A to a system 18 to be tested; and/or any other suitable components for providing any of the various functionality of network testing system 16 discussed herein or understood by one or ordinary skill in the art. (Stroud, [0114]) As discussed above, each capture and offload CLDs 102A may be connected to capture memory device 103A that the CLD 102A can write to and read from. Capture memory device 103A may comprise any suitable type of memory device, e.g., DRAM, SRAM, or Flash memory, hard dive, or any other memory device with sufficient bandwidth. In some embodiments, a high-speed double

	data rate SDRAM (e.g., DDR2 or DDR3) memory interface is provided between each capture and offload CLDs 102A and its corresponding capture memory device 103A. Thus, data may be written at near maximum-theoretical rates to maintain an accurate representation of all packets that arrived on the network, within the limits of the amount of available memory. (Stroud, [0127]) A buffer/reassembly memory device 103B may be coupled to each router CLDs 102B. Each memory device 103B may comprise any suitable memory device. For example, each memory device 103B may comprise high-speed, low-latency QDR (quad data rate) SRAM memory attached to the corresponding router CLD 103B for various offload purposes, e.g., statistics collection, packet buffering, TCP reassembly offload, etc. (Stroud, [0169]) Turning first to FIGS. 55A and 55B, card 54 includes a printed circuit board 380 that houses a pair of capture and offload CLDs 102 a-1 and 102 a-2 and associated DDR3 SDRAM memory modules (DIMMs) 103A-1 and 103A-2, a pair of routing CLDs 102 b-1 and 120 b-2 and associated QDR SRAMs 103 b-1 and 103 b-2, a traffic generation CLD 102C, a pair of network processors 105-1 and 105-2 and associated DDR2 SDRAM DIMMs 344-1 and 344-2, a control processor 106 and associated DDR3 SDRAM DIMMs 332, a management switch 110, four test interfaces 101, a backplane connector 328, a notch or bay 388 that locates a drive connector 386 for receiving a disk drive assembly 1300 that houses a disk drive 109, and various other components (e.g., including components shown in FIGS. 14A and 14B). As shown, DIMMS 103A-1, 103A-2, 344-1, 344-2, and 332 may be aligned in the same direction, e.g., in order to facilitate air flow from one or more fans across card 54 in that direction, e.g., in a direction from side-to-side across card 54. (Stroud, [0717])

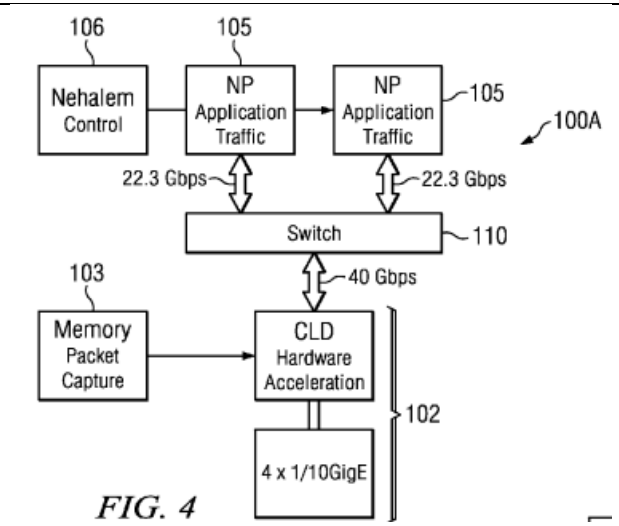


FIG. 4

(Stroud, FIG. 4)

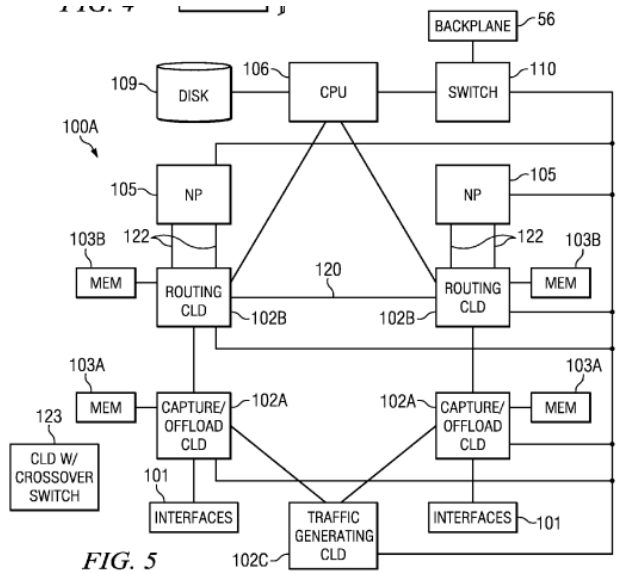


FIG. 5

(Stroud, FIG. 5)

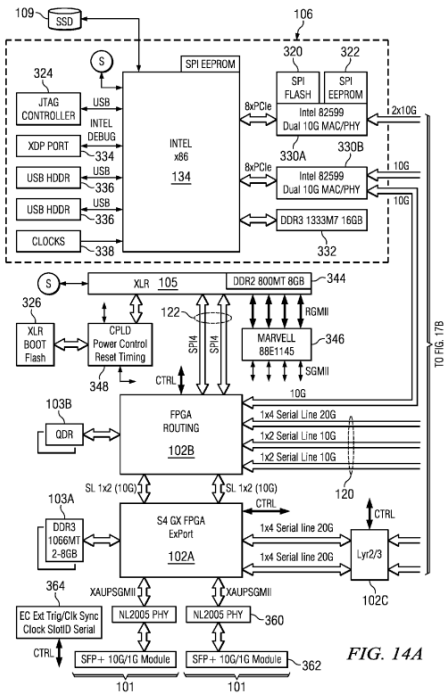


FIG. 14A

(Stroud, FIG. 14A)

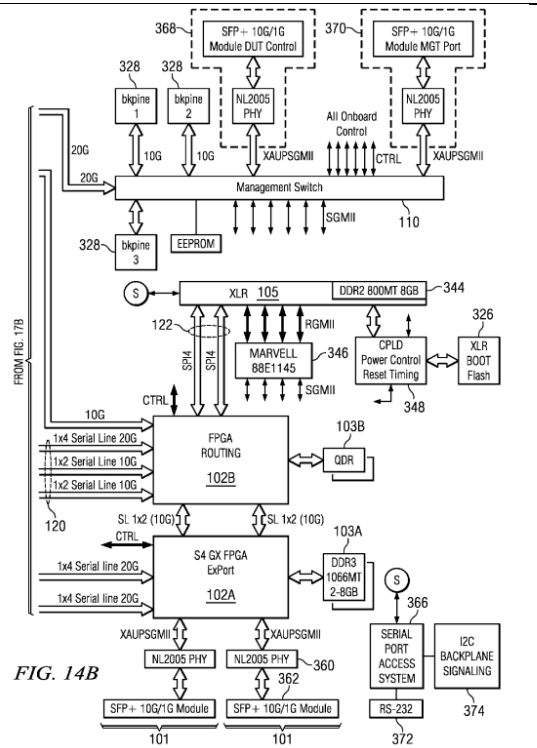
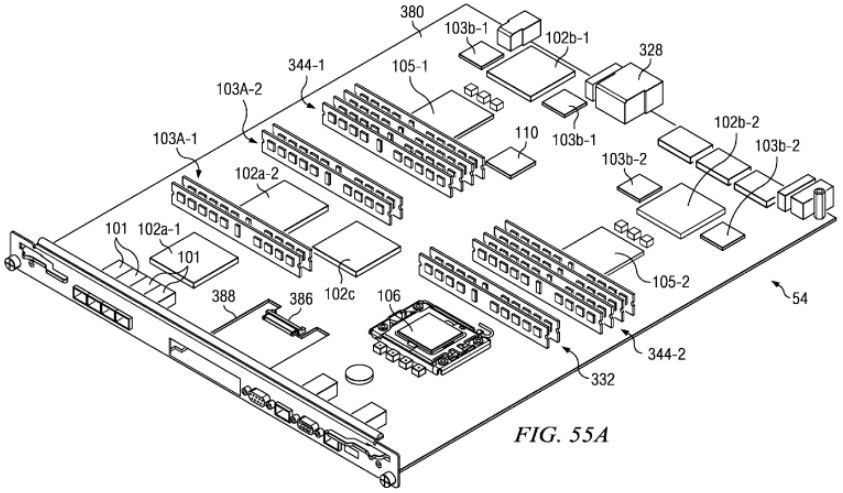
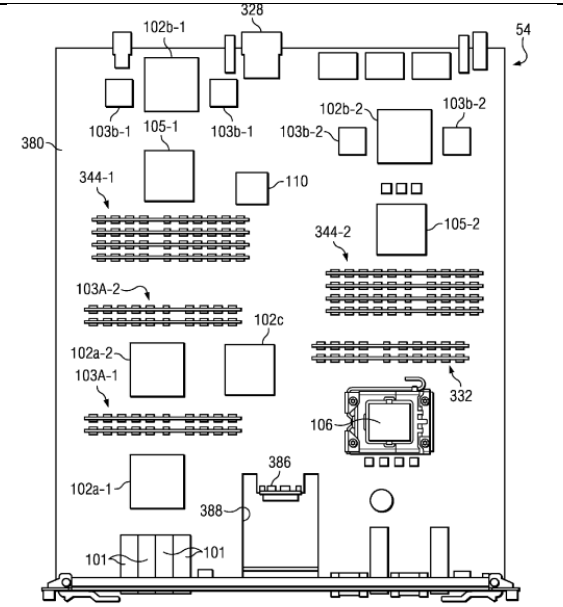


FIG. 14B

(Stroud, FIG. 14B)

	 (Stroud, FIG. 55A)

	 *FIG. 55B* (Stroud, FIG. 55B)

[7.pre] A method for data flow processing in a platform, comprising:	*See* claim element 1.pre.
[7.a]. 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;	*See* claim elements 1.pre and 1.a.

[7.b] configuring user-definable hardware modules or user-definable software modules in each of one or more of the plurality of swappable pods or cards;	*See* claim element 1.c.
[7.c] 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	*See* claim elements 1.c and 1.d.
[7.d] 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	*See* claim 4.

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	*See* claim element 1.e.

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.	*See* claim 2.
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	*See* claim 3.

with a sandbox of one of the plurality of swappable pods or cards.	
[13.pre] A tangible, non-transitory, computer-readable media having instructions thereupon which, when executed by a processor, cause the processor to perform a method comprising:	*See* claim elements 1.pre and 1.a-1.e.
[13.a] 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;	*See* claim elements 1.a-1.c.
[13.b] configuring user-definable hardware modules or user-definable software modules in each of one or more of the plurality of swappable pods or cards;	*See* claim element 1.c.
[13.c] communicating a data flow in accordance	*See* claim elements 1.c and 1.d.

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	
[13.d] 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	*See* claim elements 1.e.

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.	*See* claim 3.
15. The computer-readable media of claim 13, wherein the method further comprises:	*See* claim 4.

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.	