Exhibit 596-03

Invalidity of the ’596 Patent by Chavan ’197 and Other Prior Art

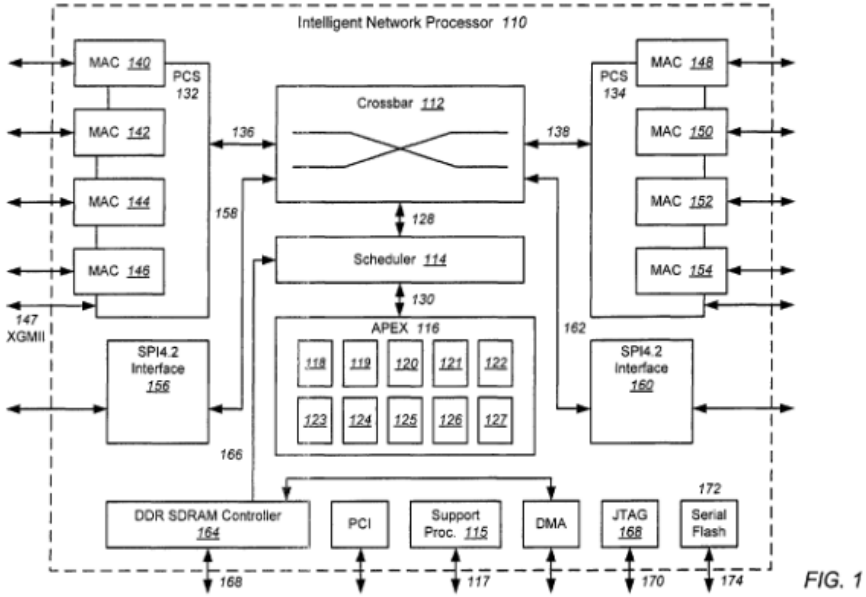
The asserted claims of U.S. Patent No. 8,924,596 are anticipated and/or rendered obvious in view of US7743197B2 (“Chavan ’197”) alone or in combination with knowledge of one of skill in the art or other prior art.

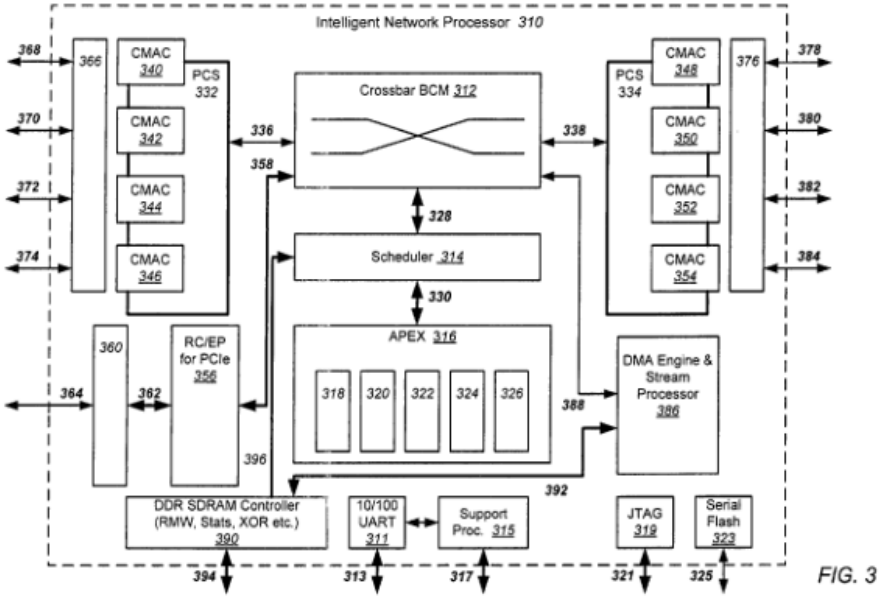
This claim chart is based on Defendants’ present understanding of the asserted claims, Plaintiffs’ vague and overbroad infringement contentions (and corresponding, implicit claim constructions), and Defendants’ investigation to date. Defendants are not adopting Plaintiffs’ apparent constructions, nor 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 these invalidity contentions as appropriate based on discovery from third parties as discovery progresses.

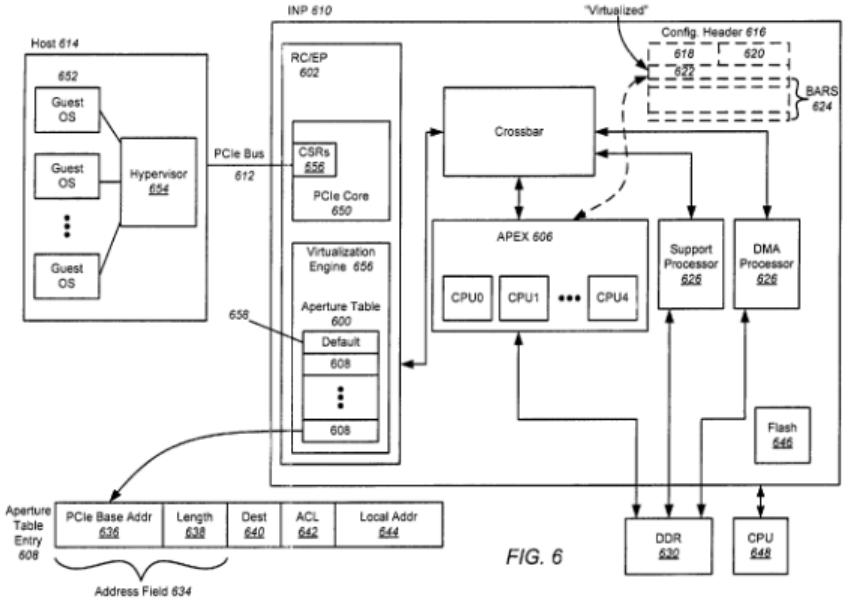
The claim chart below identifies where each limitation of each asserted claim of the ’596 patent can be found in Chavan ’197. This analysis is based on Plaintiff’s interpretation of the claims as indicated in their Infringement Contentions. The citations provided below are exemplary, rather than exhaustive, and Defendants reserve the right to rely upon any other portion of the cited references.

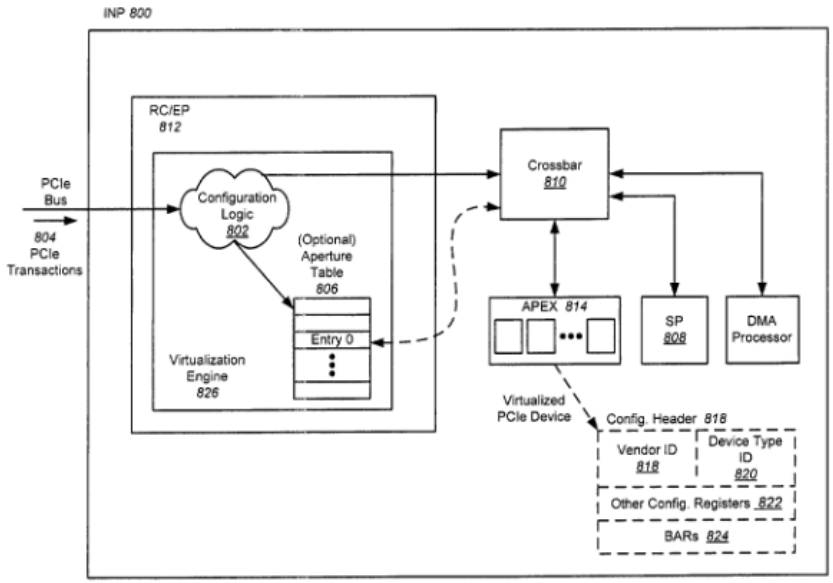
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
[1pre] A system for dividing and synchronizing a processing task across a plurality of processing elements comprising:	To the extent the preamble is considered a claim limitation, Chavan ’197 teaches “[a] system for dividing and synchronizing a processing task across a plurality of processing elements,” including at least the following exemplary citations. For example, Chavan ’197 teaches a system for dividing and synchronizing processing tasks by disclosing, as non–limiting examples, an APEX with ten core processors for executing firmware as scheduled by the scheduler in conjunction with a crossbar switch for synchronizing processing tasks. *See, e.g.:*

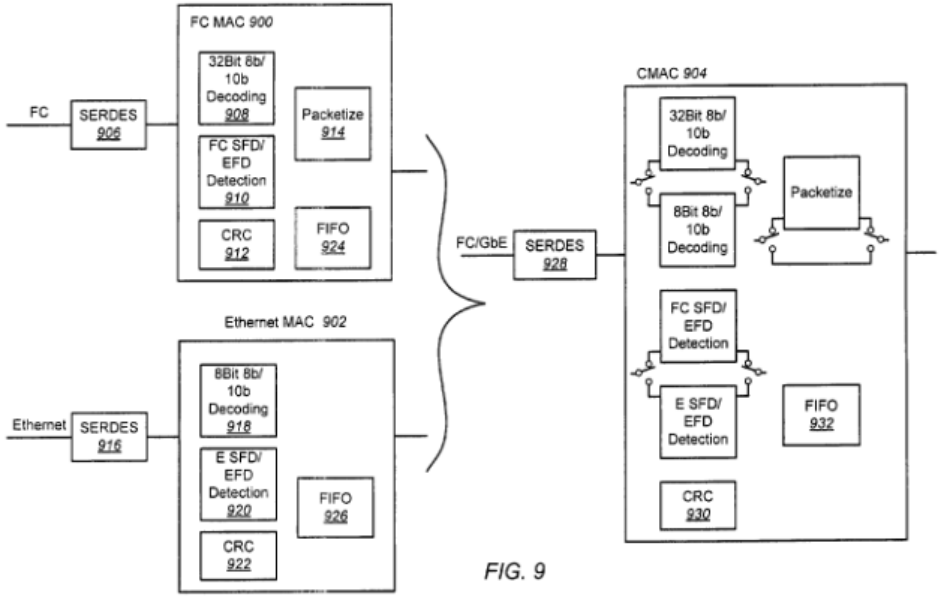
IPR2025-00478
Concurrent Ventures LLC, XtreamEdge Inc. EX2034

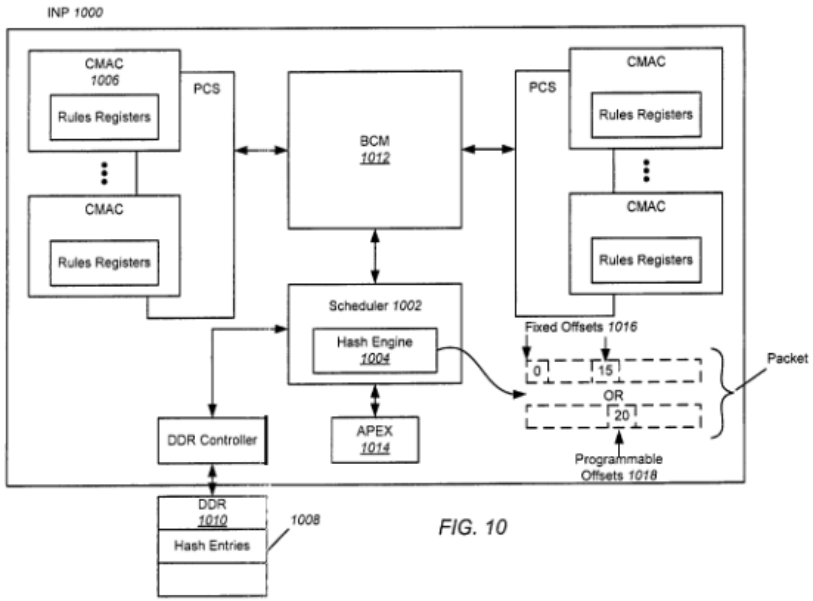
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan '197 at Figure 1.

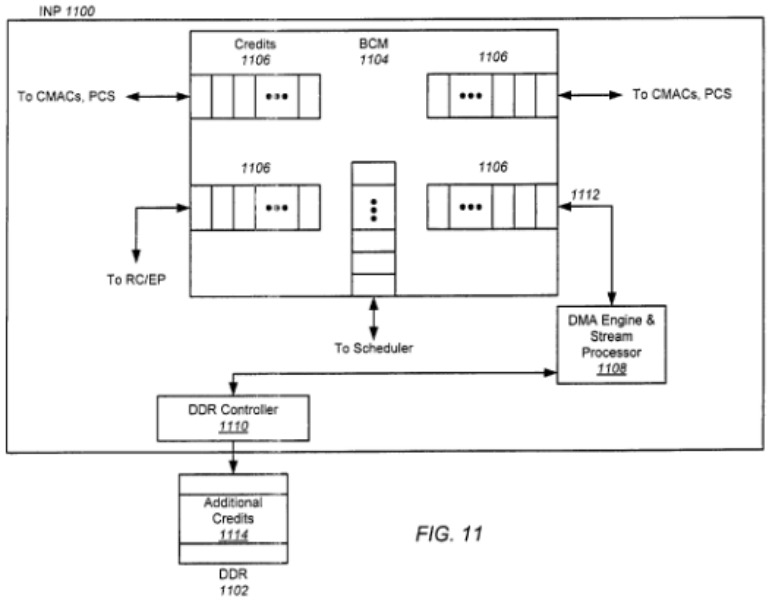
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan ’197 at Figure 3.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan '197 at Figure 6.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 8 Chavan ’197 at Figure 8.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan ’197 at Figure 9.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan ’197 at Figure 10.

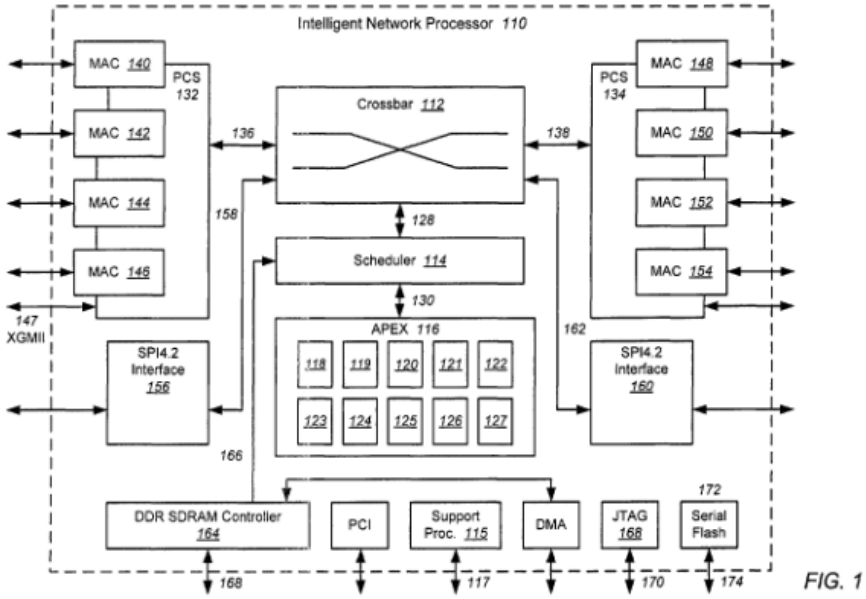
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 11 Chavan ’197 at Figure 11. “This invention relates to Intelligent Network Processors (INPs) for use in computer network systems, and more particularly to such INPs for providing Host Bus Adapter (HBA), router, and switch capabilities on Fibre Channel (FC) and Gigabit Ethernet (GbE) protocol networks connected to a host through a Peripheral Component Interconnect express (PCIe) host interface.” Chavan ’197 at col. 1, ll. 13–19. “INPs are used in computer network systems for providing HBA, router, and switch capabilities for particular protocol networks connected to a host through a host interface. An example of an INP 110 is shown in FIG. 1, which may be formed on a single silicon integrated circuit die packaged inside a ceramic, metal, or plastic housing to form a unitary component. INP 110 includes a shared memory crossbar module 112 which is controlled by a flow identification/scheduler module 114 through a bus 128. The flow identification/scheduler module 114 operates under the control of the Application Processor (APEX) module 116 acting through the bus 130, and maps the passages of crossbar locations inside the shared memory crossbar module 112. The APEX 116 in the example of FIG. 1 contains ten core processors for executing firmware, denoted as processors 118-127. The shared memory crossbar module 112 is connected to Protocol Coordination Stage (PCS) modules 132 and 134 through buses 136 and 138, respectively. PCS module 132 is connected to and controls the operation

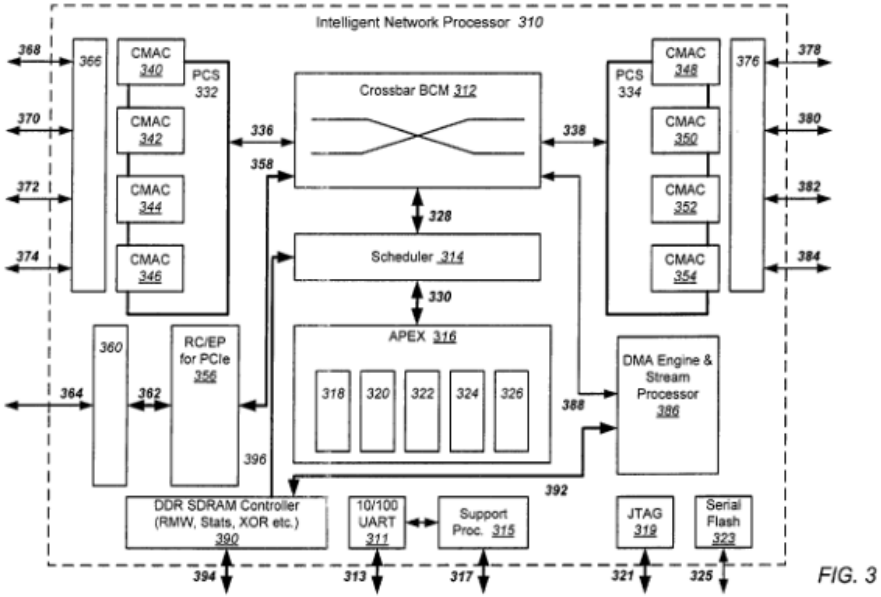
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	of Media Access Controller (MAC) modules 140, 142, 144, and 146. The MAC modules 140, 142, 144, and 146 are identical hardware interfaces capable of each communicating using the FC protocol at 1 Gigabits per second (Gbps) or 2 Gbps speeds. The speed and protocol settings for the for the MAC modules 140, 142, 144, and 146 are controlled by the PCS 132, which operates in response to control signals from a bus (not shown) connecting the PCS 132 to the APEX 116. A parallel 10 Gbit (X Gbit) Medium-Independent Interface (XGMII) port 147 may be provided from the PCS 132, as determined by control signals originating from the APEX 116. The 10 Gb XGMII interface is not coupled to any of the four MAC modules." Chavan '197 at col. 1, ll. 23–53. "The present invention is directed to INPs for providing HBA, router, and switch capabilities on FC and GbE protocol networks connected to a single host or multiple hosts (as in the case of multi-root) through a PCIe host interface. The INP includes a Bi-directional Crossbar Module (BCM) which is controlled by a flow identification and scheduler module. The scheduler operates under the control of the APEX, and maps the passages of crossbar locations inside the BCM. The APEX contains multiple core processors for executing firmware. The BCM is connected to PCS modules, which are connected to and control the operation of Combined Media Access Controller (CMAC) modules capable of each communicating using multiple protocols. The speed and protocol settings for the CMAC modules are controlled by the PCS. The PCS also has trunking modes in which it coordinates the operation of the CMAC modules together as a single trunk. A combined SERDES is coupled to each of the CMACs and selectively provides, under control of signals coupled from the APEX, electronic signals in one of multiple protocols (e.g. the FC and GbE protocols), or trunking protocols." Chavan '197 at col. 3, ll. 28–47. "An INP may also utilize programmable configuration logic that may replace the aperture table described above, or may work together with the aperture table. In this embodiment, the headers of PCIe transactions are evaluated by the configuration logic to determine the classification of the PCIe transaction (i.e. the type of cycle). Depending on the classification, the configuration logic may take certain actions. For example, if the configuration logic determines that a discovery or configuration request (i.e. a configuration cycle) is being presented, the discovery or configuration request is passed through the crossbar to the APEX. A free processor in the APEX then executes firmware to respond to and terminate the configuration cycle by providing information and indications back to the host indicating that virtual PCIe devices have been discovered and configured. The particular processor and firmware performs translations from the configuration space of the requestor to the virtual configuration space of the virtual PCIe device being emulated.

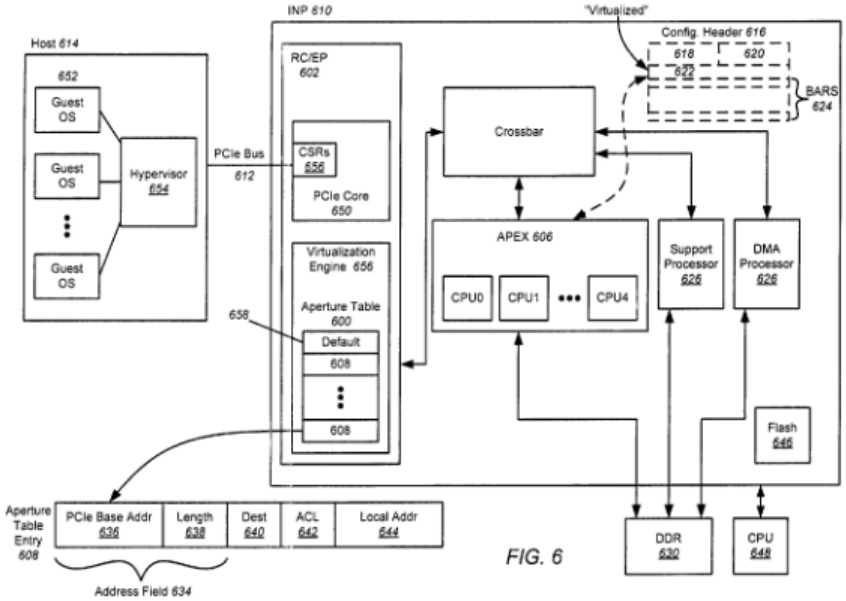
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	The common processing elements of FC and GbE MACs may be integrated into an efficient CMAC. Both protocols require SERDES, Cyclic Redundancy Checking (CRC) and First-In-First-Out (FIFO) buffers for rate matching, and thus the logic for these tasks is shared in the CMAC. However, because the logic (state machine) to perform 8b10b decoding, to detect Start of Frame Delimiters (SFDs) and End of Frame Delimiters (EFDs) for each protocol, and to packetize FC frames is different for the two protocols, this logic is duplicated and switched in or out as needed. To produce a FC/GbE CMAC, multiple bytes must be framed as packets. After the SFDs have been detected, logic specific to each protocol may be employed outside the CMAC and switched in or out depending on the protocol to parse the bytes in the packet. The CMACs are configurable by a processor to decode the incoming packets or frames as either FC "packets" or GbE packets." Chavan '197 at col. 5, l. 50 – col. 6, l. 16. "FIG. 3 is an illustration of an INP 310 implemented according to embodiments of the present invention. INP 310 may be composed of a single silicon integrated circuit die, which may be packaged inside a ceramic, metal, or plastic housing to form a unitary component. NP 310 includes a BCM 312 which is controlled by a flow identification and scheduler module 314 through a bus 328. The BCM 312 may be a multi-ported shared Random Access Memory (RAM) having memory locations mapped as passages, and operating at 224 Gbps. The scheduler 314 operates under the control of the APEX 316 acting through the bus 330, and maps the passages of crossbar locations inside the BCM 312. The scheduler 314 pre-processes received frames, packets, and PCIe requests before they are acted upon by APEX 316. The scheduler 314 also parses received FC frames into correct sequence, and parses received GbE packets (in Internet Protocol (IP) format) into a Transport Control Protocol (TCP) flow. The APEX 316 preferably contains five core processors for executing firmware, denoted as processors 318, 320, 322, 324, and 326. The processors 318, 322, 324, and 326 may be core model ARC750 as provided by ARC, Inc., which preferably each have the capability of executing 32 bit Reduced Instruction Set Computer (RISC) firmware with a clock speed of 500 MHz, and may each include 32 KB of dedicated instruction memory (not shown)." Chavan '197 at col. 7, l. 49 – col. 8, l. 6. "FIG. 6 is an illustration of an exemplary system employing an INP 610 according to embodiments of the present invention. Note that INP 610 is a simplified representation of the INP of FIG. 3, emphasizing the PCIe interface. FIG. 6 illustrates a so-called "single root" system configuration comprised of a single root complex (the host 614) and an INP 610 representing one of perhaps several endpoints. In the example of FIG. 6, the host is logically partitioned into multiple virtual servers, each running a separate "guest" OS 652. The

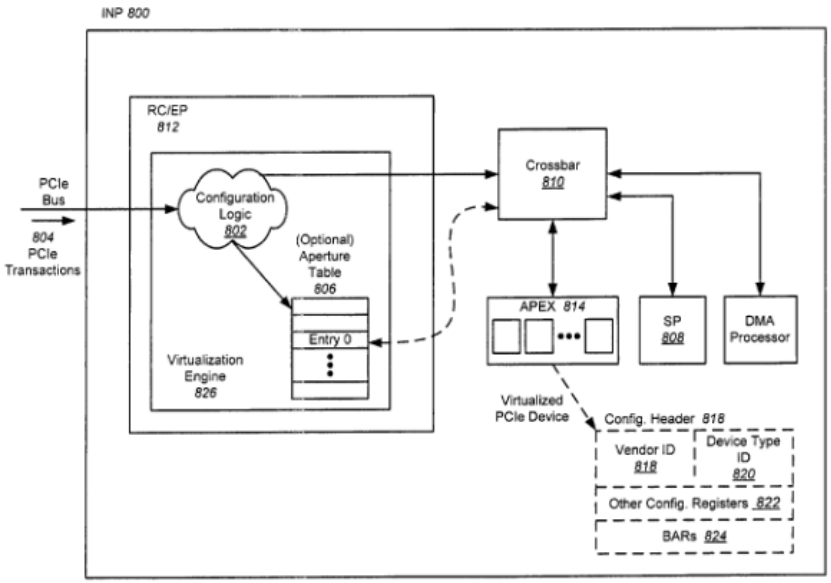
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	logical partitioning of the host 614 is managed by hypervisor software 654." Chavan '197 at col. 11, ll. 29–39. "FIG. 8 is an illustration of an exemplary system employing an INP 800 with virtualization logic 826 comprised of programmable configuration logic 802 according to embodiments of the present invention. In the alternative embodiment of FIG. 8, the configuration logic 802 may replace the aperture table described above, or may work together with the aperture table. Note that in this alternative embodiment there are no hardware configuration registers in the RC/EP 812 that terminate configuration cycles from the host. In the embodiment of FIG. 8, when PCIe transactions 804 are presented to the RC/EP 812, the headers are evaluated by the configuration logic 802 to determine the classification of the PCIe transaction (i.e. the type of cycle). Depending on the classification, the configuration logic 802 may take no further action with regard to the PCIe transaction, may direct the transaction to a processor via crossbar 810, or may perform an address translation. For example, if the configuration logic 802 determines that a discovery or configuration request (i.e. a configuration cycle) is being presented, the discovery or configuration request is passed through the crossbar 810 to the APEX 814. A free processor in the APEX 814 then executes firmware to respond to and terminate the configuration cycle by providing information and indications back to the host indicating that virtual PCIe devices have been discovered and configured. The particular processor and firmware performs translations from the configuration space of the requestor to the virtual configuration space of the virtual PCIe device being emulated." Chavan '197 at col. 16, ll. 6–33. "FIG. 10 illustrates an exemplary INP 1000 employing a hashing scheme for CMAC filtering according to embodiments of the present invention. Note that the representation of INP 1000 of FIG. 10 corresponds to the INP of FIG. 3, but is simplified to emphasize the hashing scheme. In the hashing scheme of FIG. 10, the hashing rules are made scalable without adding hardware, so each CMAC can have many more rules than their hardware registers would allow. In the embodiment of FIG. 10, the scheduler 1002 includes a hash engine 1004. Sophisticated hashing rules can be set for the hash engine 1004. For example, if a CMAC 1006 wants to accept any packet having one of 2 k destination MAC addresses, the hash engine 1004 can be configured such that for every one of the 2 k destination MAC addresses, a hash entry 1008 is created in off-chip DDR 1010. The DDR 1010 can store a large number of hash entries 1008 to achieve scalability. The hash entries 1008 are created at startup, during configuration, when a host configures the INP 1000 to receive the 2 k destination MAC addresses. After the hash entries 1008 are created, if a packet is subsequently received on CMAC 1006 containing one of the 2 k destination MAC addresses, it enters the BCM 1012 and is routed to the scheduler 1002. The

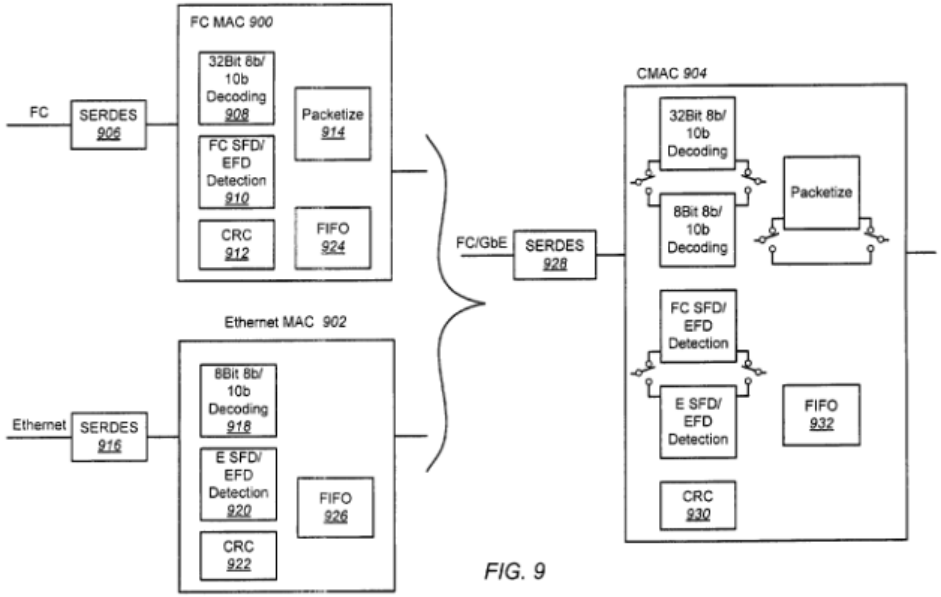
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	scheduler 1002 will do a hash of the destination MAC address in the packet, and attempt to match it with one of the hash entries 1008 in DDR 1010. This is performed by a hardware state machine, so no firmware is required. Although external memory is involved, hardware prefetching is utilized, and thus it may be done at near wire speeds. If a match is found, the packet is then forwarded to one of the APEX processors 1014. On the other hand, if a packet is received that does not have one of the 2 k destination MAC addresses, a bit in the packet is set indicating a "hash miss." The packet is thereafter still forwarded to the APEX block 1014, but the APEX firmware will detect the hash miss bit and know that the packet was not to be received, so the packet is discarded." Chavan '197 at col. 18, ll. 7–42. "74. An intelligent network processor (INP) for connection to a protocol fabric, comprising: a plurality of port interfaces for connection to a protocol fabric, and adjustable to be selectively compatible with FC or GbE; a bidirectional crossbar module (BCM) coupled to receive messages from each of the plurality of port modules, and having crossbar passageways compatible with both FC and GbE messages; a DDR memory interface for connecting to an external DDR memory, for storing and retrieving data messages in queues contained in the DDR memory, and connected to send and receive data messages directly from the BCM; and an application processor module (APEX) coupled to the BCM for controlling the connections between the port interfaces and the DDR memory interface through the BCM, and enabling direct memory access operations between the DDR memory interface and the BCM without data messages passing through the APEX." Chavan '197 at col. 28, l. 60 – col. 29, l. 11. "75. A method for connecting a plurality of port interfaces to a protocol fabric, comprising: providing a plurality of port interfaces for connection to a protocol fabric, the plurality of port interfaces being adjustable to be selectively compatible with FC or GbE; receiving messages from each of the plurality of port interfaces through a bidirectional crossbar module (BCM) having crossbar passageways compatible with both FC and GbE messages; connecting to an external DDR memory for storing and retrieving data messages in queues contained in DDR memory, and for sending and receiving data messages directly from the BCM; and controlling the connections between the port interfaces and the DDR memory interface through the BCM, and enabling direct memory access operations between the DDR memory interface and the BCM without data messages passing through the APEX." Chavan '197 at col. 29, l. 12 – col. 30, l. 14.

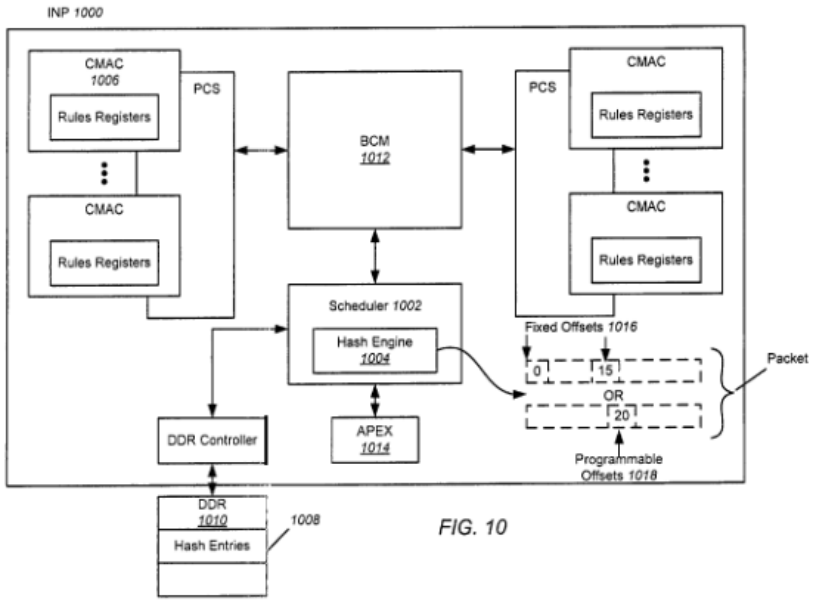
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
[1a] an input queue implemented in hardware;	Chavan ’197 teaches “an input queue implemented in hardware,” including at least the following exemplary citations. Plaintiffs have cited the use of an input queue implemented in memory through software as allegedly satisfying limitation [1a]. *See* Am. Compl. at ¶59; ’596 Infringement Contentions at 43–58. To the extent that Plaintiffs argue that Chavan ’197’s input queue is in memory Chavan ’197 teaches or renders obvious claim limitation [1a] under Plaintiffs’ apparent interpretation of the claim. For example, Chavan ’197 teaches an input queue implemented in hardware, by disclosing, as non–limiting examples, queues in external DDR. *See, e.g.:*  Chavan ’197 at Figure 1.

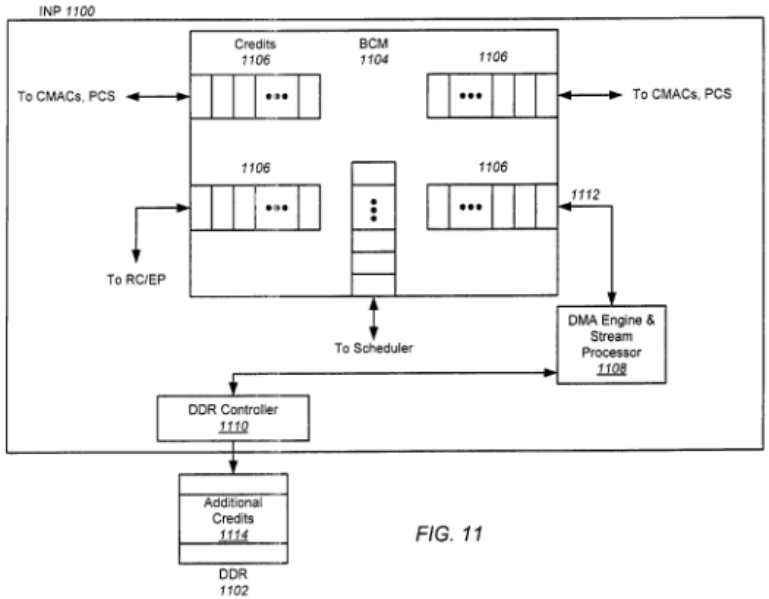
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan ’197 at Figure 3.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan '197 at Figure 6.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 8 Chavan ’197 at Figure 8.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan ’197 at Figure 9.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan ’197 at Figure 10.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 11 Chavan ’197 at Figure 11. “INPs are used in computer network systems for providing HBA, router, and switch capabilities for particular protocol networks connected to a host through a host interface. An example of an INP 110 is shown in FIG. 1, which may be formed on a single silicon integrated circuit die packaged inside a ceramic, metal, or plastic housing to form a unitary component. INP 110 includes a shared memory crossbar module 112 which is controlled by a flow identification/scheduler module 114 through a bus 128. The flow identification/scheduler module 114 operates under the control of the Application Processor (APEX) module 116 acting through the bus 130, and maps the passages of crossbar locations inside the shared memory crossbar module 112. The APEX 116 in the example of FIG. 1 contains ten core processors for executing firmware, denoted as processors 118-127. The shared memory crossbar module 112 is connected to Protocol Coordination Stage (PCS) modules 132 and 134 through buses 136 and 138, respectively. PCS module 132 is connected to and controls the operation of Media Access Controller (MAC) modules 140, 142, 144, and 146. The MAC modules 140, 142, 144, and 146 are identical hardware interfaces capable of each communicating using the FC protocol at 1 Gigabits per second (Gbps) or 2 Gbps speeds. The speed and protocol settings for the for the MAC modules 140, 142, 144, and 146 are controlled by the PCS 132, which operates in response to control signals from a bus (not shown)

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	connecting the PCS 132 to the APEX 116. A parallel 10 Gbit (X Gbit) Medium-Independent Interface (XGMII) port 147 may be provided from the PCS 132, as determined by control signals originating from the APEX 116. The 10 Gb XGMII interface is not coupled to any of the four MAC modules." Chavan '197 at col. 1, ll. 23–53. "A Double Data Rate (DDR) Synchronous Dynamic Random Access Memory (SDRAM) controller 164 is connected to the scheduler 114 through bus 166. The function of the controller 164 is to control the sending and receiving of data to and from an external DDR memory module (not shown) connected to an external bus 168 of the controller. A support processor 115 handles certain processing tasks not otherwise covered by the APEX 116, and has an external connection 117. A Joint Test Action Group (JTAG) module 168 has an external connection 170, and is used for boundary scan testing of the hardware contained within INP 110. Serial flash memory 172 has an external connection 174, and is a non-volatile reprogrammable memory which may be loaded through the connection to store firmware for operation of APEX 116 and support processor 115." Chavan '197 at col. 2, ll. 14–28. "The present invention is directed to INPs for providing HBA, router, and switch capabilities on FC and GbE protocol networks connected to a single host or multiple hosts (as in the case of multi-root) through a PCIe host interface. The INP includes a Bi-directional Crossbar Module (BCM) which is controlled by a flow identification and scheduler module. The scheduler operates under the control of the APEX, and maps the passages of crossbar locations inside the BCM. The APEX contains multiple core processors for executing firmware. The BCM is connected to PCS modules, which are connected to and control the operation of Combined Media Access Controller (CMAC) modules capable of each communicating using multiple protocols. The speed and protocol settings for the CMAC modules are controlled by the PCS. The PCS also has trunking modes in which it coordinates the operation of the CMAC modules together as a single trunk. A combined SERDES is coupled to each of the CMACs and selectively provides, under control of signals coupled from the APEX, electronic signals in one of multiple protocols (e.g. the FC and GbE protocols), or trunking protocols." Chavan '197 at col. 3, ll. 28–47. "A Root Complex and EndPoint (RC/EP) module is connected to the BCM, and is connected to a SERDES for use with PCIe bus signals. A Direct Memory Access (DMA) engine and stream processor is connected to the BCM, and is also connected to a DDR SDRAM controller. The function of the controller is to control the sending and receiving of data to and from an external DDR memory module. A data pathway from the DMA engine and stream processor to the BCM allows DMA memory operations to occur directly with the

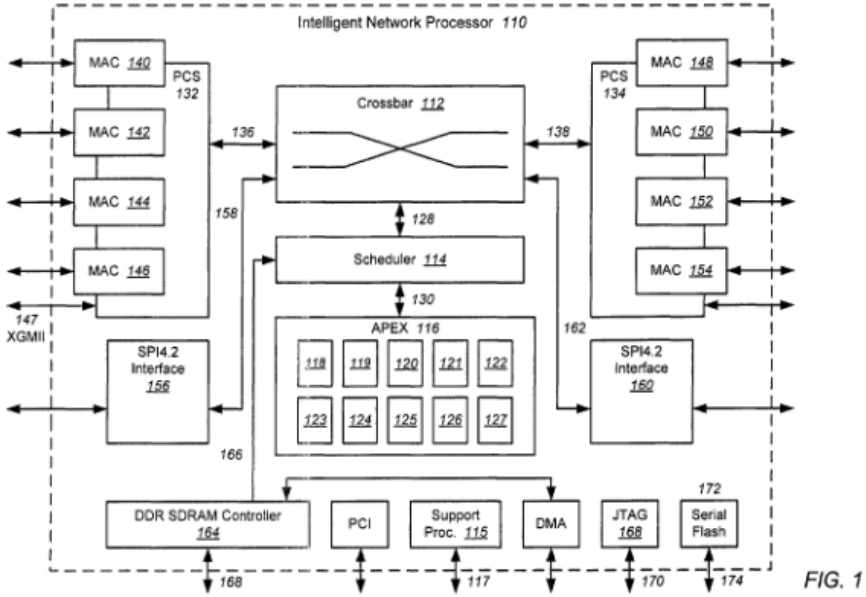
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	controller and its attached DDR memory module, without data flow through the APEX, for data being exchanged with other sources. The controller is also connected to the scheduler, and creates queues in the external DDR memory for storing data messages, and also writes to and reads from such queues." Chavan '197 at col. 3, ll. 48–62. "An INP may also utilize programmable configuration logic that may replace the aperture table described above, or may work together with the aperture table. In this embodiment, the headers of PCIe transactions are evaluated by the configuration logic to determine the classification of the PCIe transaction (i.e. the type of cycle). Depending on the classification, the configuration logic may take certain actions. For example, if the configuration logic determines that a discovery or configuration request (i.e. a configuration cycle) is being presented, the discovery or configuration request is passed through the crossbar to the APEX. A free processor in the APEX then executes firmware to respond to and terminate the configuration cycle by providing information and indications back to the host indicating that virtual PCIe devices have been discovered and configured. The particular processor and firmware performs translations from the configuration space of the requestor to the virtual configuration space of the virtual PCIe device being emulated. The common processing elements of FC and GbE MACs may be integrated into an efficient CMAC. Both protocols require SERDES, Cyclic Redundancy Checking (CRC) and First-In-First-Out (FIFO) buffers for rate matching, and thus the logic for these tasks is shared in the CMAC. However, because the logic (state machine) to perform 8b10b decoding, to detect Start of Frame Delimiters (SFDs) and End of Frame Delimiters (EFDs) for each protocol, and to packetize FC frames is different for the two protocols, this logic is duplicated and switched in or out as needed. To produce a FC/GbE CMAC, multiple bytes must be framed as packets. After the SFDs have been detected, logic specific to each protocol may be employed outside the CMAC and switched in or out depending on the protocol to parse the bytes in the packet. The CMACs are configurable by a processor to decode the incoming packets or frames as either FC "packets" or GbE packets." Chavan '197 at col. 5, l. 50 – col. 6, l. 16. "FIG. 3 is an illustration of an INP 310 implemented according to embodiments of the present invention. INP 310 may be composed of a single silicon integrated circuit die, which may be packaged inside a ceramic, metal, or plastic housing to form a unitary component. NP 310 includes a BCM 312 which is controlled by a flow identification and scheduler module 314 through a bus 328. The BCM 312 may be a multi-ported shared Random Access Memory (RAM) having memory locations mapped as passages, and operating at 224 Gbps. The scheduler 314 operates under the control of the APEX 316 acting through the bus 330, and maps the passages of crossbar locations inside the BCM 312. The scheduler 314 pre-processes received frames,

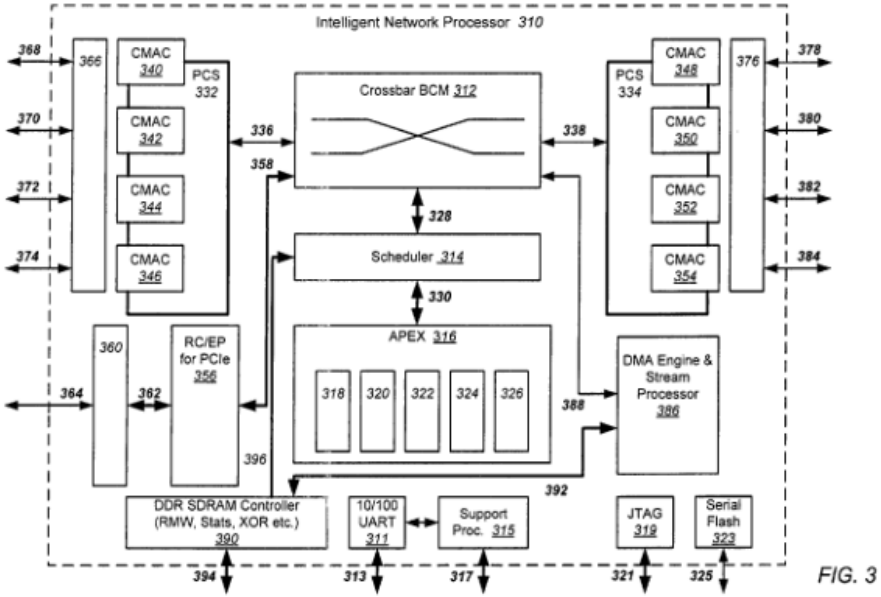
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	packets, and PCIe requests before they are acted upon by APEX 316. The scheduler 314 also parses received FC frames into correct sequence, and parses received GbE packets (in Internet Protocol (IP) format) into a Transport Control Protocol (TCP) flow. The APEX 316 preferably contains five core processors for executing firmware, denoted as processors 318, 320, 322, 324, and 326. The processors 318, 322, 324, and 326 may be core model ARC750 as provided by ARC, Inc., which preferably each have the capability of executing 32 bit Reduced Instruction Set Computer (RISC) firmware with a clock speed of 500 MHz, and may each include 32 KB of dedicated instruction memory (not shown)." Chavan '197 at col. 7, l. 49 – col. 8, l. 6. "The DMA engine and stream processor 386 is connected to the BCM 312 through bus 388, and is connected to the DDR SDRAM controller 390 through bus 392. The function of the controller 390 is to control the sending and receiving of data to and from an external DDR memory module (not shown) connected to the bus 394 of the controller 390. The controller 390 includes capabilities for providing RMW, statistics collection, XOR masking for data being transferred to or from the external DDR memory module. The data pathway through the bus 392 to the DMA engine and stream processor 386 and through the bus 388 to the BCM 312 allows DMA memory operations to occur directly with the controller 390 and its attached DDR memory module, without data flow through the APEX 316, for data being exchanged with other sources including the PCIe capabilities accessed through the bus 358 and RC/EP module 356, and the FC and GbE capabilities accessed through the buses 336 and 338 with the PCS modules 332 and 334. The ability to allow direct access for DMA operations through the BCM 312 with the DDR SDRAM controller 390 saves time that would be otherwise consumed by data being transferred into and out of the APEX 316. The controller 390 is also connected to the scheduler 314 through the bus 396. The controller 390 creates queues in the external DDR memory for storing data messages, and writes to and reads from such queues. The use of such queues, and the direct memory access to the crossbar BCM 312 (through the buses 388 and 392, controller 390, and DMA engine and stream processor 386) effectively increases the size of the crossbar BCM 312, and allows more switching pathways to be created than if the BCM 312 were used without the external DDR memory attached to connection 394." Chavan '197 at col. 8, l. 66 – col. 9, l. 29. "FIG. 6 is an illustration of an exemplary system employing an INP 610 according to embodiments of the present invention. Note that INP 610 is a simplified representation of the INP of FIG. 3, emphasizing the PCIe interface. FIG. 6 illustrates a so-called "single root" system configuration comprised of a single root complex (the host 614) and an INP 610 representing one of perhaps several endpoints. In the example of FIG. 6, the host is logically partitioned into multiple virtual servers, each running a separate "guest" OS 652. The

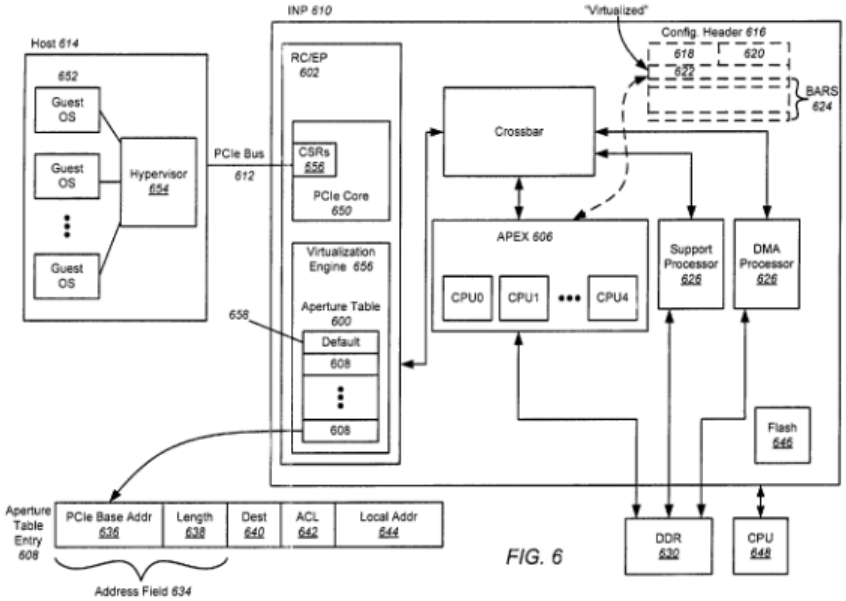
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	logical partitioning of the host 614 is managed by hypervisor software 654." Chavan '197 at col. 11, ll. 29–39. "FIG. 8 is an illustration of an exemplary system employing an INP 800 with virtualization logic 826 comprised of programmable configuration logic 802 according to embodiments of the present invention. In the alternative embodiment of FIG. 8, the configuration logic 802 may replace the aperture table described above, or may work together with the aperture table. Note that in this alternative embodiment there are no hardware configuration registers in the RC/EP 812 that terminate configuration cycles from the host. In the embodiment of FIG. 8, when PCIe transactions 804 are presented to the RC/EP 812, the headers are evaluated by the configuration logic 802 to determine the classification of the PCIe transaction (i.e. the type of cycle). Depending on the classification, the configuration logic 802 may take no further action with regard to the PCIe transaction, may direct the transaction to a processor via crossbar 810, or may perform an address translation. For example, if the configuration logic 802 determines that a discovery or configuration request (i.e. a configuration cycle) is being presented, the discovery or configuration request is passed through the crossbar 810 to the APEX 814. A free processor in the APEX 814 then executes firmware to respond to and terminate the configuration cycle by providing information and indications back to the host indicating that virtual PCIe devices have been discovered and configured. The particular processor and firmware performs translations from the configuration space of the requestor to the virtual configuration space of the virtual PCIe device being emulated." Chavan '197 at col. 16, ll. 6–33. "FIG. 9 is an illustration of how the common processing elements of MACs for FC 900 and MACs for GbE 902 can be integrated into an efficient CMAC 904 according to embodiments of the present invention. Conventionally, FC frames are received and deserialized by SERDES 906, and are then processed by a FC MAC 900. This processing typically includes 32-bit 8b10b decoding 908, detection of a SFD (a.k.a. a "k" character) and an EFD 910, CRC 912, generating packets from the FC frames 914, and buffering in a FIFO buffer 924. Similarly, GbE packets are received and deserialized by SERDES 916, and are then processed by a GbE MAC 902. This processing typically includes 8-bit 8b10b decoding 918, detection of a SFD and an EFD 920, CRC 922, and FIFO buffering 926. Because of the processing similarities, embodiments of the present invention integrate the data paths for both the FC and GbE protocols in a FC/GbE CMAC 904. Both protocols require SERDES 928, CRC 930 and FIFOs 932 for rate matching, and thus the logic for these tasks is shared in the CMAC 904. However, because the logic (state machine) to perform 8b10b decoding, to detect SFDs and EFDs for each protocol, and to packetize FC frames is different for the two protocols, this logic must be duplicated and switched in or out as

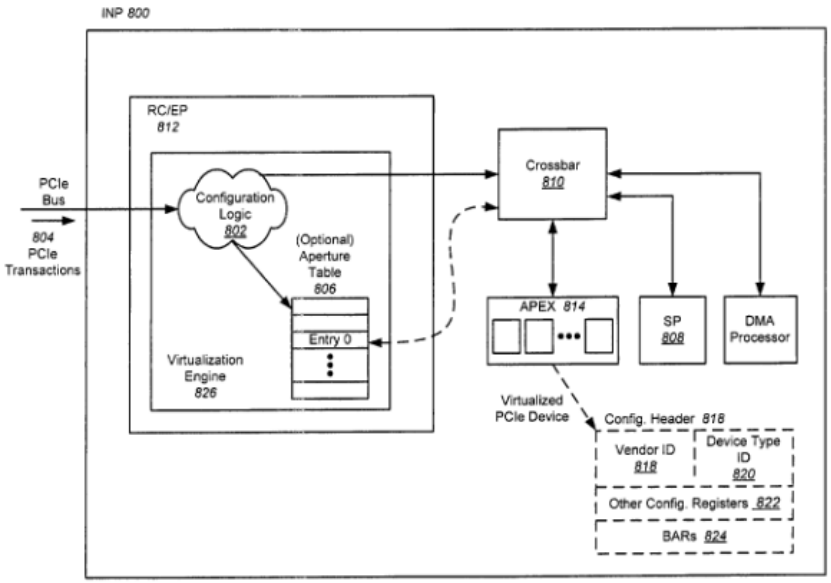
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	needed. To produce a FC/GbE CMAC, multiple bytes must be framed as packets (even though FC uses "frame" terminology). The bytes in a packet are well-defined in the FC and GbE standards. After the SFDs have been detected, logic specific to each protocol may be employed outside the CMAC and switched in or out depending on the protocol to parse the bytes in the packet. The CMACs are configurable by a processor to decode the incoming packets or frames as either FC "packets" or GbE packets." Chavan '197 at col. 17, ll. 22–53. "FIG. 10 illustrates an exemplary INP 1000 employing a hashing scheme for CMAC filtering according to embodiments of the present invention. Note that the representation of INP 1000 of FIG. 10 corresponds to the INP of FIG. 3, but is simplified to emphasize the hashing scheme. In the hashing scheme of FIG. 10, the hashing rules are made scalable without adding hardware, so each CMAC can have many more rules than their hardware registers would allow. In the embodiment of FIG. 10, the scheduler 1002 includes a hash engine 1004. Sophisticated hashing rules can be set for the hash engine 1004. For example, if a CMAC 1006 wants to accept any packet having one of 2 k destination MAC addresses, the hash engine 1004 can be configured such that for every one of the 2 k destination MAC addresses, a hash entry 1008 is created in off-chip DDR 1010. The DDR 1010 can store a large number of hash entries 1008 to achieve scalability. The hash entries 1008 are created at startup, during configuration, when a host configures the INP 1000 to receive the 2 k destination MAC addresses. After the hash entries 1008 are created, if a packet is subsequently received on CMAC 1006 containing one of the 2 k destination MAC addresses, it enters the BCM 1012 and is routed to the scheduler 1002. The scheduler 1002 will do a hash of the destination MAC address in the packet, and attempt to match it with one of the hash entries 1008 in DDR 1010. This is performed by a hardware state machine, so no firmware is required. Although external memory is involved, hardware prefetching is utilized, and thus it may be done at near wire speeds. If a match is found, the packet is then forwarded to one of the APEX processors 1014. On the other hand, if a packet is received that does not have one of the 2 k destination MAC addresses, a bit in the packet is set indicating a "hash miss." The packet is thereafter still forwarded to the APEX block 1014, but the APEX firmware will detect the hash miss bit and know that the packet was not to be received, so the packet is discarded." Chavan '197 at col. 18, ll. 7–42. "74. An intelligent network processor (INP) for connection to a protocol fabric, comprising: a plurality of port interfaces for connection to a protocol fabric, and adjustable to be selectively compatible with FC or GbE; a bidirectional crossbar module (BCM) coupled to receive messages from each of the plurality of port modules, and having crossbar passageways compatible with both FC and GbE messages;

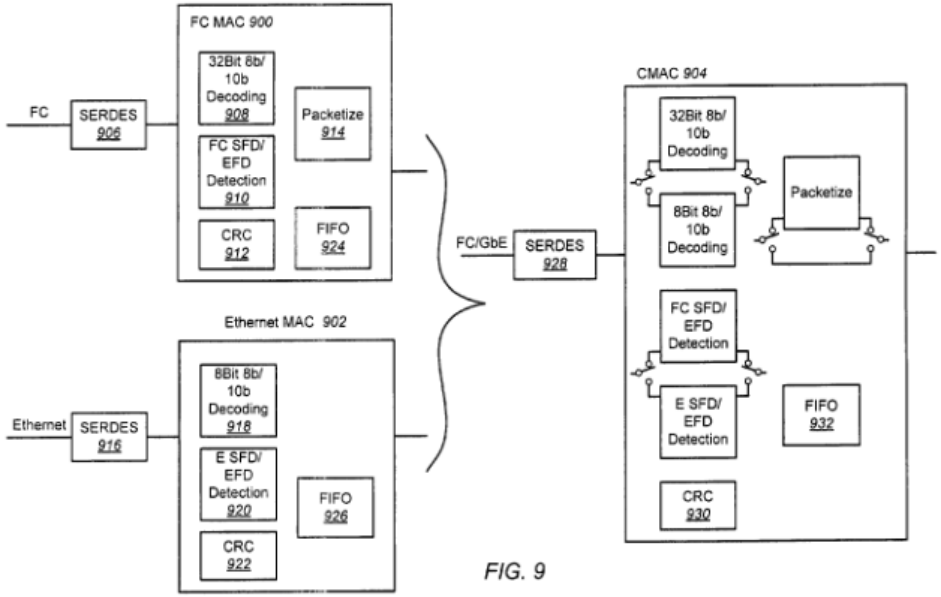
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	a DDR memory interface for connecting to an external DDR memory, for storing and retrieving data messages in queues contained in the DDR memory, and connected to send and receive data messages directly from the BCM; and an application processor module (APEX) coupled to the BCM for controlling the connections between the port interfaces and the DDR memory interface through the BCM, and enabling direct memory access operations between the DDR memory interface and the BCM without data messages passing through the APEX." Chavan '197 at col. 28, l. 60 – col. 29, l. 11. "75. A method for connecting a plurality of port interfaces to a protocol fabric, comprising: providing a plurality of port interfaces for connection to a protocol fabric, the plurality of port interfaces being adjustable to be selectively compatible with FC or GbE; receiving messages from each of the plurality of port interfaces through a bidirectional crossbar module (BCM) having crossbar passageways compatible with both FC and GbE messages; connecting to an external DDR memory for storing and retrieving data messages in queues contained in DDR memory, and for sending and receiving data messages directly from the BCM; and controlling the connections between the port interfaces and the DDR memory interface through the BCM, and enabling direct memory access operations between the DDR memory interface and the BCM without data messages passing through the APEX." Chavan '197 at col. 29, l. 12 – col. 30, l. 14.
[1b] an output queue implemented in hardware;	Chavan '197 teaches "an output queue implemented in hardware," including at least the following exemplary citations. Plaintiffs have cited the use of an output queue implemented in memory through software as allegedly satisfying limitation [1b]. *See* Am. Compl. at ¶59; '596 Infringement Contentions at 43–58. To the extent that Plaintiffs argue that Chavan '197's output queue is in memory Chavan '197 teaches or renders obvious claim limitation [1b] under Plaintiffs' apparent interpretation of the claim. For example, Chavan '197 teaches an output queue implemented in hardware, by disclosing, as non–limiting examples, queues in external DDR. *See, e.g.:*

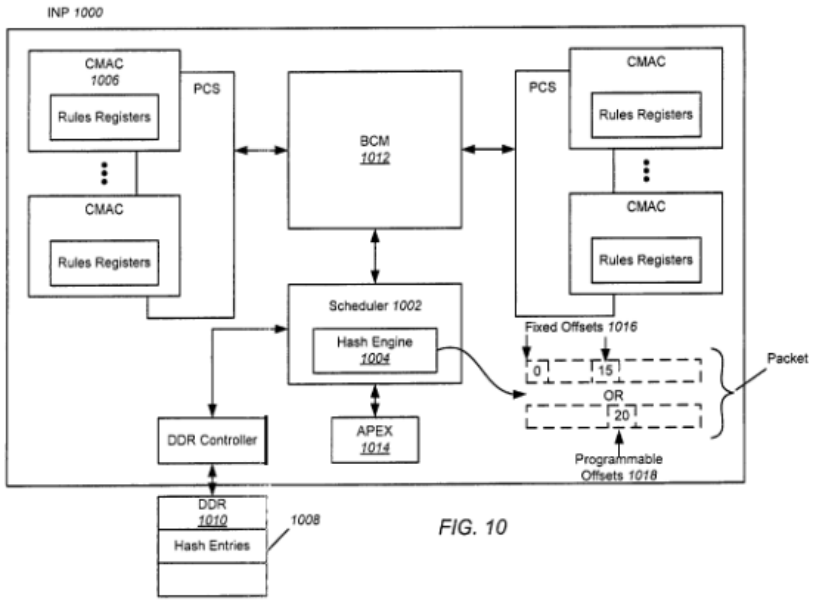
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan '197 at Figure 1.

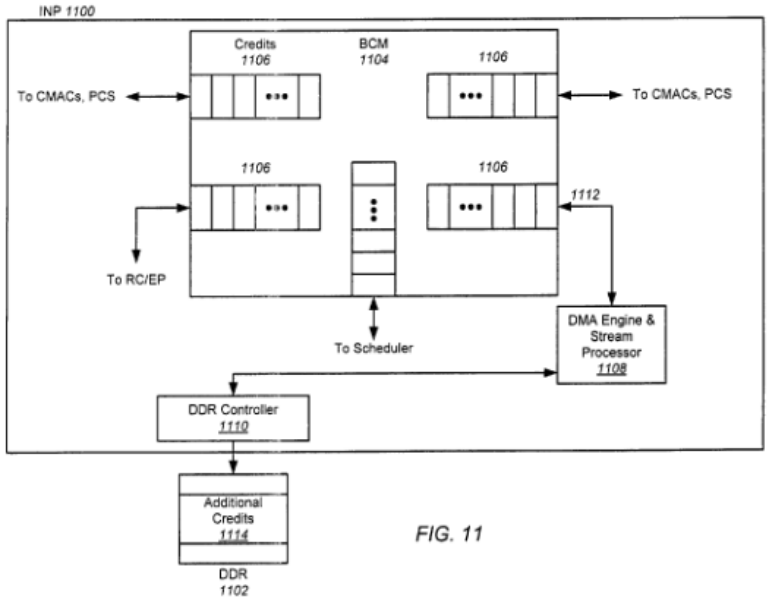
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan ’197 at Figure 3.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan '197 at Figure 6.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 8 Chavan ’197 at Figure 8.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan ’197 at Figure 9.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan ’197 at Figure 10.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 11 Chavan ’197 at Figure 11. “INPs are used in computer network systems for providing HBA, router, and switch capabilities for particular protocol networks connected to a host through a host interface. An example of an INP 110 is shown in FIG. 1, which may be formed on a single silicon integrated circuit die packaged inside a ceramic, metal, or plastic housing to form a unitary component. INP 110 includes a shared memory crossbar module 112 which is controlled by a flow identification/scheduler module 114 through a bus 128. The flow identification/scheduler module 114 operates under the control of the Application Processor (APEX) module 116 acting through the bus 130, and maps the passages of crossbar locations inside the shared memory crossbar module 112. The APEX 116 in the example of FIG. 1 contains ten core processors for executing firmware, denoted as processors 118-127. The shared memory crossbar module 112 is connected to Protocol Coordination Stage (PCS) modules 132 and 134 through buses 136 and 138, respectively. PCS module 132 is connected to and controls the operation of Media Access Controller (MAC) modules 140, 142, 144, and 146. The MAC modules 140, 142, 144, and 146 are identical hardware interfaces capable of each communicating using the FC protocol at 1 Gigabits per second (Gbps) or 2 Gbps speeds. The speed and protocol settings for the for the MAC modules 140, 142, 144, and 146 are controlled by the PCS 132, which operates in response to control signals from a bus (not shown)

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	connecting the PCS 132 to the APEX 116. A parallel 10 Gbit (X Gbit) Medium-Independent Interface (XGMII) port 147 may be provided from the PCS 132, as determined by control signals originating from the APEX 116. The 10 Gb XGMII interface is not coupled to any of the four MAC modules." Chavan '197 at col. 1, ll. 23–53. "A Double Data Rate (DDR) Synchronous Dynamic Random Access Memory (SDRAM) controller 164 is connected to the scheduler 114 through bus 166. The function of the controller 164 is to control the sending and receiving of data to and from an external DDR memory module (not shown) connected to an external bus 168 of the controller. A support processor 115 handles certain processing tasks not otherwise covered by the APEX 116, and has an external connection 117. A Joint Test Action Group (JTAG) module 168 has an external connection 170, and is used for boundary scan testing of the hardware contained within INP 110. Serial flash memory 172 has an external connection 174, and is a non-volatile reprogrammable memory which may be loaded through the connection to store firmware for operation of APEX 116 and support processor 115." Chavan '197 at col. 2, ll. 14–28. "The present invention is directed to INPs for providing HBA, router, and switch capabilities on FC and GbE protocol networks connected to a single host or multiple hosts (as in the case of multi-root) through a PCIe host interface. The INP includes a Bi-directional Crossbar Module (BCM) which is controlled by a flow identification and scheduler module. The scheduler operates under the control of the APEX, and maps the passages of crossbar locations inside the BCM. The APEX contains multiple core processors for executing firmware. The BCM is connected to PCS modules, which are connected to and control the operation of Combined Media Access Controller (CMAC) modules capable of each communicating using multiple protocols. The speed and protocol settings for the CMAC modules are controlled by the PCS. The PCS also has trunking modes in which it coordinates the operation of the CMAC modules together as a single trunk. A combined SERDES is coupled to each of the CMACs and selectively provides, under control of signals coupled from the APEX, electronic signals in one of multiple protocols (e.g. the FC and GbE protocols), or trunking protocols." Chavan '197 at col. 3, ll. 28–47. "A Root Complex and EndPoint (RC/EP) module is connected to the BCM, and is connected to a SERDES for use with PCIe bus signals. A Direct Memory Access (DMA) engine and stream processor is connected to the BCM, and is also connected to a DDR SDRAM controller. The function of the controller is to control the sending and receiving of data to and from an external DDR memory module. A data pathway from the DMA engine and stream processor to the BCM allows DMA memory operations to occur directly with the

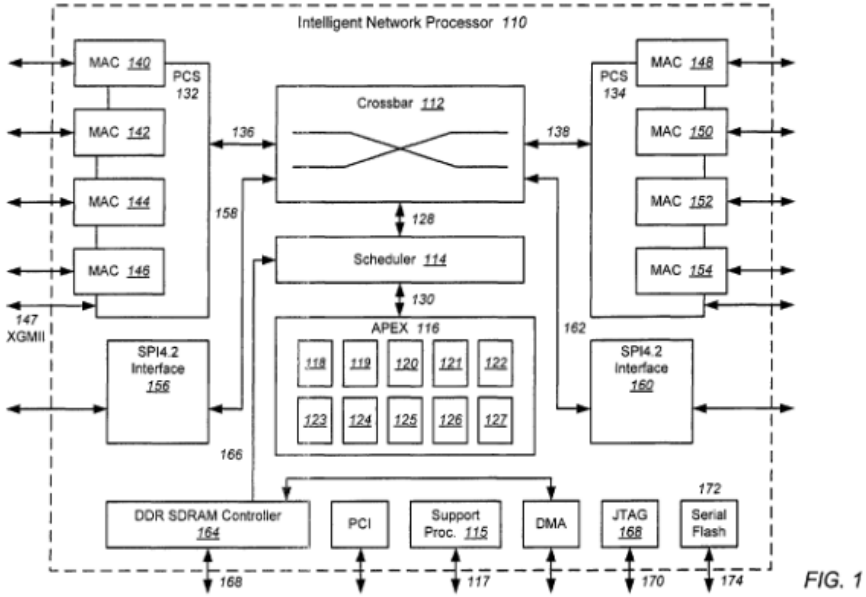
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	controller and its attached DDR memory module, without data flow through the APEX, for data being exchanged with other sources. The controller is also connected to the scheduler, and creates queues in the external DDR memory for storing data messages, and also writes to and reads from such queues." Chavan '197 at col. 3, ll. 48–62. "An INP may also utilize programmable configuration logic that may replace the aperture table described above, or may work together with the aperture table. In this embodiment, the headers of PCIe transactions are evaluated by the configuration logic to determine the classification of the PCIe transaction (i.e. the type of cycle). Depending on the classification, the configuration logic may take certain actions. For example, if the configuration logic determines that a discovery or configuration request (i.e. a configuration cycle) is being presented, the discovery or configuration request is passed through the crossbar to the APEX. A free processor in the APEX then executes firmware to respond to and terminate the configuration cycle by providing information and indications back to the host indicating that virtual PCIe devices have been discovered and configured. The particular processor and firmware performs translations from the configuration space of the requestor to the virtual configuration space of the virtual PCIe device being emulated. The common processing elements of FC and GbE MACs may be integrated into an efficient CMAC. Both protocols require SERDES, Cyclic Redundancy Checking (CRC) and First-In-First-Out (FIFO) buffers for rate matching, and thus the logic for these tasks is shared in the CMAC. However, because the logic (state machine) to perform 8b10b decoding, to detect Start of Frame Delimiters (SFDs) and End of Frame Delimiters (EFDs) for each protocol, and to packetize FC frames is different for the two protocols, this logic is duplicated and switched in or out as needed. To produce a FC/GbE CMAC, multiple bytes must be framed as packets. After the SFDs have been detected, logic specific to each protocol may be employed outside the CMAC and switched in or out depending on the protocol to parse the bytes in the packet. The CMACs are configurable by a processor to decode the incoming packets or frames as either FC "packets" or GbE packets." Chavan '197 at col. 5, l. 50 – col. 6, l. 16. "FIG. 3 is an illustration of an INP 310 implemented according to embodiments of the present invention. INP 310 may be composed of a single silicon integrated circuit die, which may be packaged inside a ceramic, metal, or plastic housing to form a unitary component. NP 310 includes a BCM 312 which is controlled by a flow identification and scheduler module 314 through a bus 328. The BCM 312 may be a multi-ported shared Random Access Memory (RAM) having memory locations mapped as passages, and operating at 224 Gbps. The scheduler 314 operates under the control of the APEX 316 acting through the bus 330, and maps the passages of crossbar locations inside the BCM 312. The scheduler 314 pre-processes received frames,

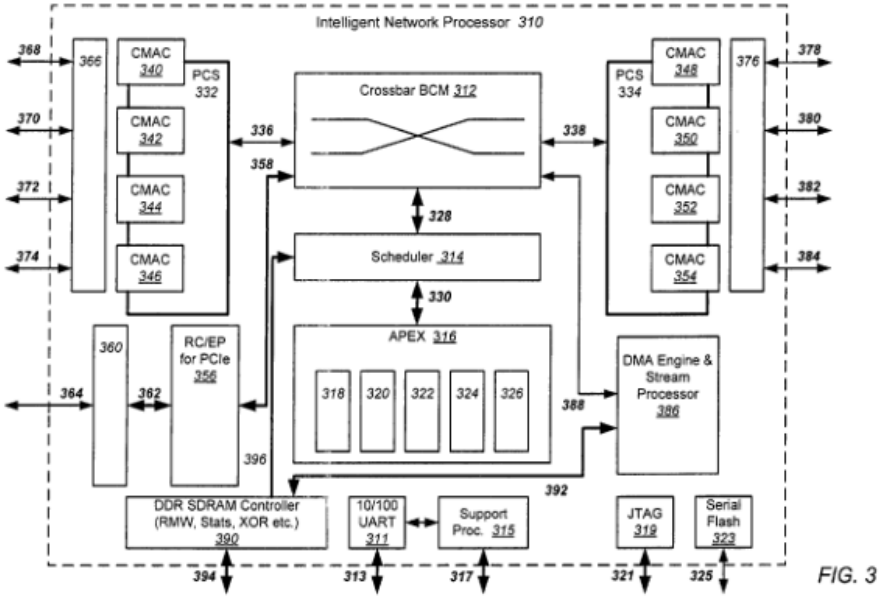
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	packets, and PCIe requests before they are acted upon by APEX 316. The scheduler 314 also parses received FC frames into correct sequence, and parses received GbE packets (in Internet Protocol (IP) format) into a Transport Control Protocol (TCP) flow. The APEX 316 preferably contains five core processors for executing firmware, denoted as processors 318, 320, 322, 324, and 326. The processors 318, 322, 324, and 326 may be core model ARC750 as provided by ARC, Inc., which preferably each have the capability of executing 32 bit Reduced Instruction Set Computer (RISC) firmware with a clock speed of 500 MHz, and may each include 32 KB of dedicated instruction memory (not shown)." Chavan '197 at col. 7, l. 49 – col. 8, l. 6. "The DMA engine and stream processor 386 is connected to the BCM 312 through bus 388, and is connected to the DDR SDRAM controller 390 through bus 392. The function of the controller 390 is to control the sending and receiving of data to and from an external DDR memory module (not shown) connected to the bus 394 of the controller 390. The controller 390 includes capabilities for providing RMW, statistics collection, XOR masking for data being transferred to or from the external DDR memory module. The data pathway through the bus 392 to the DMA engine and stream processor 386 and through the bus 388 to the BCM 312 allows DMA memory operations to occur directly with the controller 390 and its attached DDR memory module, without data flow through the APEX 316, for data being exchanged with other sources including the PCIe capabilities accessed through the bus 358 and RC/EP module 356, and the FC and GbE capabilities accessed through the buses 336 and 338 with the PCS modules 332 and 334. The ability to allow direct access for DMA operations through the BCM 312 with the DDR SDRAM controller 390 saves time that would be otherwise consumed by data being transferred into and out of the APEX 316. The controller 390 is also connected to the scheduler 314 through the bus 396. The controller 390 creates queues in the external DDR memory for storing data messages, and writes to and reads from such queues. The use of such queues, and the direct memory access to the crossbar BCM 312 (through the buses 388 and 392, controller 390, and DMA engine and stream processor 386) effectively increases the size of the crossbar BCM 312, and allows more switching pathways to be created than if the BCM 312 were used without the external DDR memory attached to connection 394." Chavan '197 at col. 8, l. 66 – col. 9, l. 29. "FIG. 6 is an illustration of an exemplary system employing an INP 610 according to embodiments of the present invention. Note that INP 610 is a simplified representation of the INP of FIG. 3, emphasizing the PCIe interface. FIG. 6 illustrates a so-called "single root" system configuration comprised of a single root complex (the host 614) and an INP 610 representing one of perhaps several endpoints. In the example of FIG. 6, the host is logically partitioned into multiple virtual servers, each running a separate "guest" OS 652. The

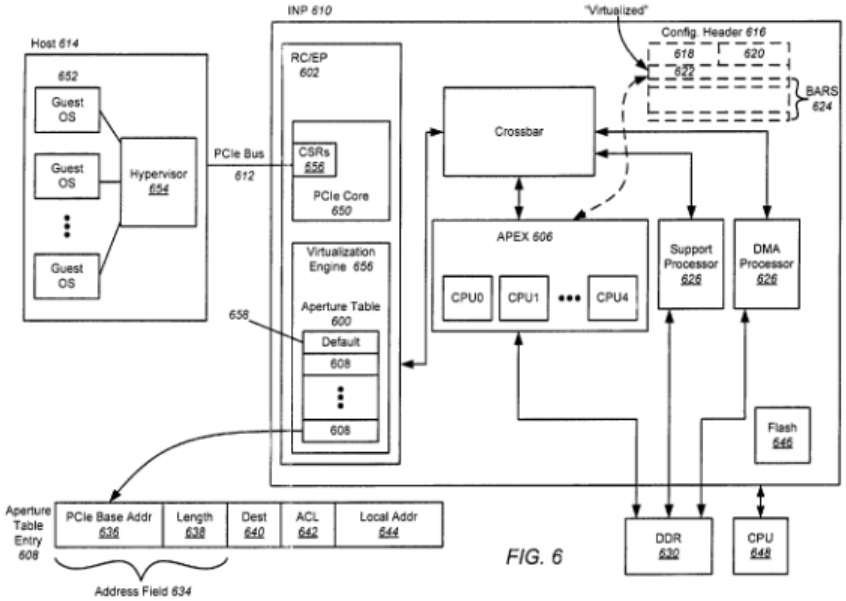
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	logical partitioning of the host 614 is managed by hypervisor software 654." Chavan '197 at col. 11, ll. 29–39. "FIG. 8 is an illustration of an exemplary system employing an INP 800 with virtualization logic 826 comprised of programmable configuration logic 802 according to embodiments of the present invention. In the alternative embodiment of FIG. 8, the configuration logic 802 may replace the aperture table described above, or may work together with the aperture table. Note that in this alternative embodiment there are no hardware configuration registers in the RC/EP 812 that terminate configuration cycles from the host. In the embodiment of FIG. 8, when PCIe transactions 804 are presented to the RC/EP 812, the headers are evaluated by the configuration logic 802 to determine the classification of the PCIe transaction (i.e. the type of cycle). Depending on the classification, the configuration logic 802 may take no further action with regard to the PCIe transaction, may direct the transaction to a processor via crossbar 810, or may perform an address translation. For example, if the configuration logic 802 determines that a discovery or configuration request (i.e. a configuration cycle) is being presented, the discovery or configuration request is passed through the crossbar 810 to the APEX 814. A free processor in the APEX 814 then executes firmware to respond to and terminate the configuration cycle by providing information and indications back to the host indicating that virtual PCIe devices have been discovered and configured. The particular processor and firmware performs translations from the configuration space of the requestor to the virtual configuration space of the virtual PCIe device being emulated." Chavan '197 at col. 16, ll. 6–33. "FIG. 9 is an illustration of how the common processing elements of MACs for FC 900 and MACs for GbE 902 can be integrated into an efficient CMAC 904 according to embodiments of the present invention. Conventionally, FC frames are received and deserialized by SERDES 906, and are then processed by a FC MAC 900. This processing typically includes 32-bit 8b10b decoding 908, detection of a SFD (a.k.a. a "k" character) and an EFD 910, CRC 912, generating packets from the FC frames 914, and buffering in a FIFO buffer 924. Similarly, GbE packets are received and deserialized by SERDES 916, and are then processed by a GbE MAC 902. This processing typically includes 8-bit 8b10b decoding 918, detection of a SFD and an EFD 920, CRC 922, and FIFO buffering 926. Because of the processing similarities, embodiments of the present invention integrate the data paths for both the FC and GbE protocols in a FC/GbE CMAC 904. Both protocols require SERDES 928, CRC 930 and FIFOs 932 for rate matching, and thus the logic for these tasks is shared in the CMAC 904. However, because the logic (state machine) to perform 8b10b decoding, to detect SFDs and EFDs for each protocol, and to packetize FC frames is different for the two protocols, this logic must be duplicated and switched in or out as

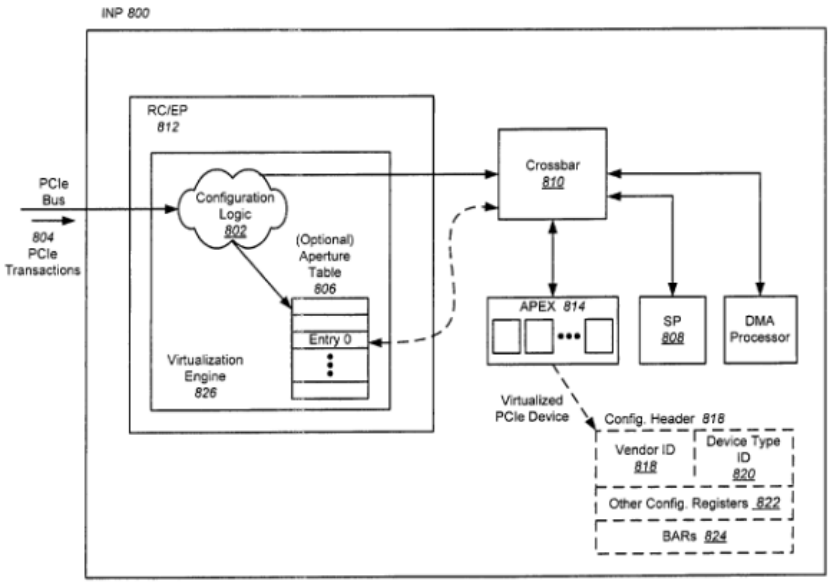
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	needed. To produce a FC/GbE CMAC, multiple bytes must be framed as packets (even though FC uses "frame" terminology). The bytes in a packet are well-defined in the FC and GbE standards. After the SFDs have been detected, logic specific to each protocol may be employed outside the CMAC and switched in or out depending on the protocol to parse the bytes in the packet. The CMACs are configurable by a processor to decode the incoming packets or frames as either FC "packets" or GbE packets." Chavan '197 at col. 17, ll. 22–53. "FIG. 10 illustrates an exemplary INP 1000 employing a hashing scheme for CMAC filtering according to embodiments of the present invention. Note that the representation of INP 1000 of FIG. 10 corresponds to the INP of FIG. 3, but is simplified to emphasize the hashing scheme. In the hashing scheme of FIG. 10, the hashing rules are made scalable without adding hardware, so each CMAC can have many more rules than their hardware registers would allow. In the embodiment of FIG. 10, the scheduler 1002 includes a hash engine 1004. Sophisticated hashing rules can be set for the hash engine 1004. For example, if a CMAC 1006 wants to accept any packet having one of 2 k destination MAC addresses, the hash engine 1004 can be configured such that for every one of the 2 k destination MAC addresses, a hash entry 1008 is created in off-chip DDR 1010. The DDR 1010 can store a large number of hash entries 1008 to achieve scalability. The hash entries 1008 are created at startup, during configuration, when a host configures the INP 1000 to receive the 2 k destination MAC addresses. After the hash entries 1008 are created, if a packet is subsequently received on CMAC 1006 containing one of the 2 k destination MAC addresses, it enters the BCM 1012 and is routed to the scheduler 1002. The scheduler 1002 will do a hash of the destination MAC address in the packet, and attempt to match it with one of the hash entries 1008 in DDR 1010. This is performed by a hardware state machine, so no firmware is required. Although external memory is involved, hardware prefetching is utilized, and thus it may be done at near wire speeds. If a match is found, the packet is then forwarded to one of the APEX processors 1014. On the other hand, if a packet is received that does not have one of the 2 k destination MAC addresses, a bit in the packet is set indicating a "hash miss." The packet is thereafter still forwarded to the APEX block 1014, but the APEX firmware will detect the hash miss bit and know that the packet was not to be received, so the packet is discarded." Chavan '197 at col. 18, ll. 7–42. "74. An intelligent network processor (INP) for connection to a protocol fabric, comprising: a plurality of port interfaces for connection to a protocol fabric, and adjustable to be selectively compatible with FC or GbE; a bidirectional crossbar module (BCM) coupled to receive messages from each of the plurality of port modules, and having crossbar passageways compatible with both FC and GbE messages;

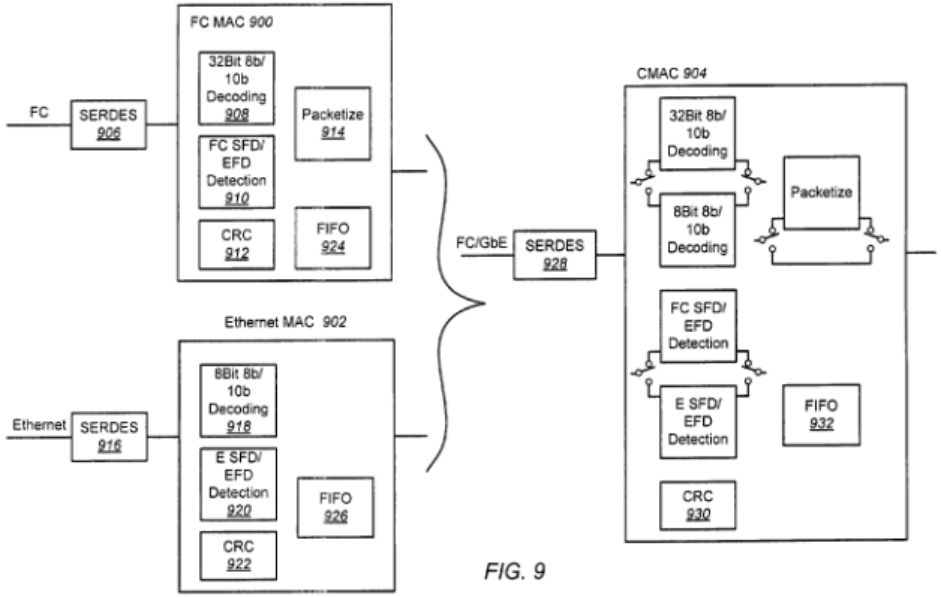
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	a DDR memory interface for connecting to an external DDR memory, for storing and retrieving data messages in queues contained in the DDR memory, and connected to send and receive data messages directly from the BCM; and an application processor module (APEX) coupled to the BCM for controlling the connections between the port interfaces and the DDR memory interface through the BCM, and enabling direct memory access operations between the DDR memory interface and the BCM without data messages passing through the APEX." Chavan '197 at col. 28, l. 60 – col. 29, l. 11. "75. A method for connecting a plurality of port interfaces to a protocol fabric, comprising: providing a plurality of port interfaces for connection to a protocol fabric, the plurality of port interfaces being adjustable to be selectively compatible with FC or GbE; receiving messages from each of the plurality of port interfaces through a bidirectional crossbar module (BCM) having crossbar passageways compatible with both FC and GbE messages; connecting to an external DDR memory for storing and retrieving data messages in queues contained in DDR memory, and for sending and receiving data messages directly from the BCM; and controlling the connections between the port interfaces and the DDR memory interface through the BCM, and enabling direct memory access operations between the DDR memory interface and the BCM without data messages passing through the APEX." Chavan '197 at col. 29, l. 12 – col. 30, l. 14.
[1c] a first processing element having access to said input queue and said output queue;	Chavan '197 teaches "a first processing element having access to said input queue and said output queue," including at least the following exemplary citations. For example, Chavan '197 teaches a first processing element having access to said input queue and said output queue, by disclosing, as non–limiting examples, a processor within the APEX that has access to queues. Any one of these processors within the APEX satisfies the first processing element. The processor has access to both the input queue and output queue. *See, e.g.:*

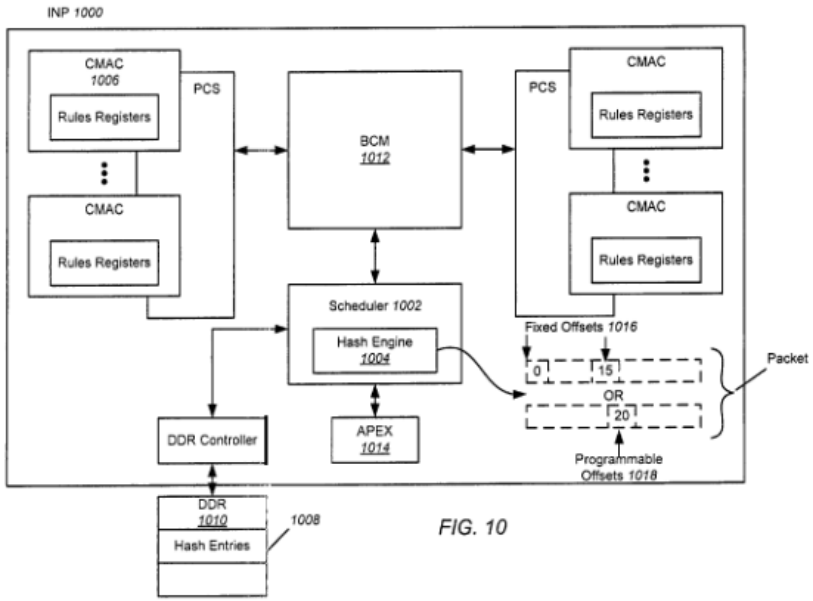
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan '197 at Figure 1.

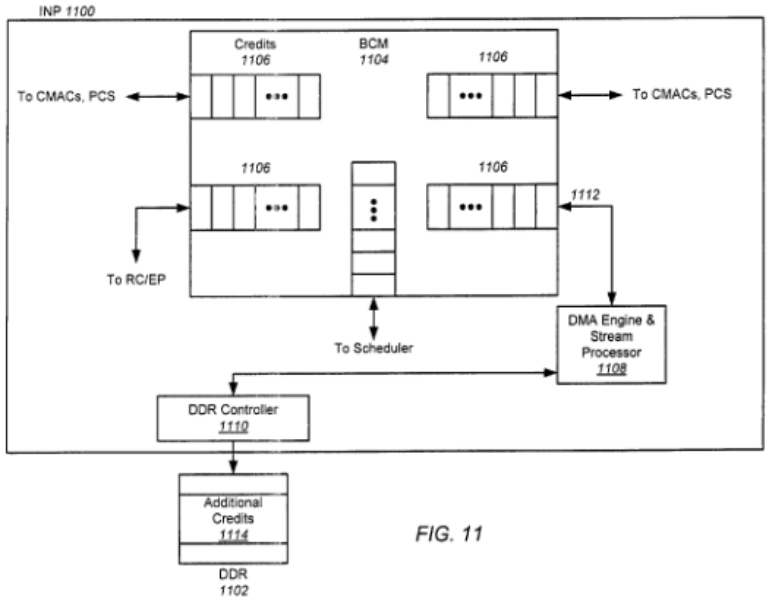
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan ’197 at Figure 3.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan ’197 at Figure 6.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 8 Chavan ’197 at Figure 8.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan ’197 at Figure 9.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan ’197 at Figure 10.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 11 Chavan ’197 at Figure 11. “INPs are used in computer network systems for providing HBA, router, and switch capabilities for particular protocol networks connected to a host through a host interface. An example of an INP 110 is shown in FIG. 1, which may be formed on a single silicon integrated circuit die packaged inside a ceramic, metal, or plastic housing to form a unitary component. INP 110 includes a shared memory crossbar module 112 which is controlled by a flow identification/scheduler module 114 through a bus 128. The flow identification/scheduler module 114 operates under the control of the Application Processor (APEX) module 116 acting through the bus 130, and maps the passages of crossbar locations inside the shared memory crossbar module 112. The APEX 116 in the example of FIG. 1 contains ten core processors for executing firmware, denoted as processors 118-127. The shared memory crossbar module 112 is connected to Protocol Coordination Stage (PCS) modules 132 and 134 through buses 136 and 138, respectively. PCS module 132 is connected to and controls the operation of Media Access Controller (MAC) modules 140, 142, 144, and 146. The MAC modules 140, 142, 144, and 146 are identical hardware interfaces capable of each communicating using the FC protocol at 1 Gigabits per second (Gbps) or 2 Gbps speeds. The speed and protocol settings for the for the MAC modules 140, 142, 144, and 146 are controlled by the PCS 132, which operates in response to control signals from a bus (not shown)

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	connecting the PCS 132 to the APEX 116. A parallel 10 Gbit (X Gbit) Medium-Independent Interface (XGMII) port 147 may be provided from the PCS 132, as determined by control signals originating from the APEX 116. The 10 Gb XGMII interface is not coupled to any of the four MAC modules." Chavan '197 at col. 1, ll. 23–53. "A Double Data Rate (DDR) Synchronous Dynamic Random Access Memory (SDRAM) controller 164 is connected to the scheduler 114 through bus 166. The function of the controller 164 is to control the sending and receiving of data to and from an external DDR memory module (not shown) connected to an external bus 168 of the controller. A support processor 115 handles certain processing tasks not otherwise covered by the APEX 116, and has an external connection 117. A Joint Test Action Group (JTAG) module 168 has an external connection 170, and is used for boundary scan testing of the hardware contained within INP 110. Serial flash memory 172 has an external connection 174, and is a non-volatile reprogrammable memory which may be loaded through the connection to store firmware for operation of APEX 116 and support processor 115." Chavan '197 at col. 2, ll. 14–28. "The present invention is directed to INPs for providing HBA, router, and switch capabilities on FC and GbE protocol networks connected to a single host or multiple hosts (as in the case of multi-root) through a PCIe host interface. The INP includes a Bi-directional Crossbar Module (BCM) which is controlled by a flow identification and scheduler module. The scheduler operates under the control of the APEX, and maps the passages of crossbar locations inside the BCM. The APEX contains multiple core processors for executing firmware. The BCM is connected to PCS modules, which are connected to and control the operation of Combined Media Access Controller (CMAC) modules capable of each communicating using multiple protocols. The speed and protocol settings for the CMAC modules are controlled by the PCS. The PCS also has trunking modes in which it coordinates the operation of the CMAC modules together as a single trunk. A combined SERDES is coupled to each of the CMACs and selectively provides, under control of signals coupled from the APEX, electronic signals in one of multiple protocols (e.g. the FC and GbE protocols), or trunking protocols." Chavan '197 at col. 3, ll. 28–47. "A Root Complex and EndPoint (RC/EP) module is connected to the BCM, and is connected to a SERDES for use with PCIe bus signals. A Direct Memory Access (DMA) engine and stream processor is connected to the BCM, and is also connected to a DDR SDRAM controller. The function of the controller is to control the sending and receiving of data to and from an external DDR memory module. A data pathway from the DMA engine and stream processor to the BCM allows DMA memory operations to occur directly with the

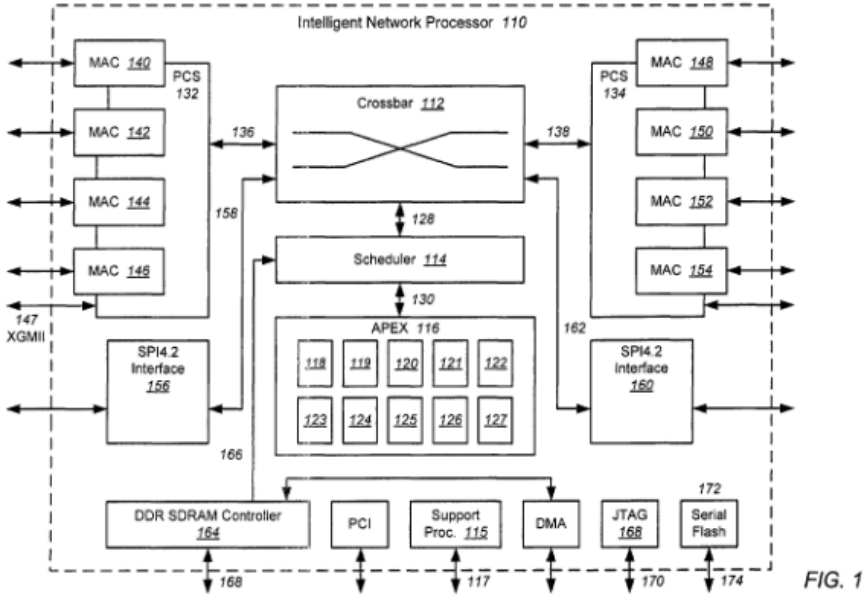
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	controller and its attached DDR memory module, without data flow through the APEX, for data being exchanged with other sources. The controller is also connected to the scheduler, and creates queues in the external DDR memory for storing data messages, and also writes to and reads from such queues." Chavan '197 at col. 3, ll. 48–62. "An INP may also utilize programmable configuration logic that may replace the aperture table described above, or may work together with the aperture table. In this embodiment, the headers of PCIe transactions are evaluated by the configuration logic to determine the classification of the PCIe transaction (i.e. the type of cycle). Depending on the classification, the configuration logic may take certain actions. For example, if the configuration logic determines that a discovery or configuration request (i.e. a configuration cycle) is being presented, the discovery or configuration request is passed through the crossbar to the APEX. A free processor in the APEX then executes firmware to respond to and terminate the configuration cycle by providing information and indications back to the host indicating that virtual PCIe devices have been discovered and configured. The particular processor and firmware performs translations from the configuration space of the requestor to the virtual configuration space of the virtual PCIe device being emulated. The common processing elements of FC and GbE MACs may be integrated into an efficient CMAC. Both protocols require SERDES, Cyclic Redundancy Checking (CRC) and First-In-First-Out (FIFO) buffers for rate matching, and thus the logic for these tasks is shared in the CMAC. However, because the logic (state machine) to perform 8b10b decoding, to detect Start of Frame Delimiters (SFDs) and End of Frame Delimiters (EFDs) for each protocol, and to packetize FC frames is different for the two protocols, this logic is duplicated and switched in or out as needed. To produce a FC/GbE CMAC, multiple bytes must be framed as packets. After the SFDs have been detected, logic specific to each protocol may be employed outside the CMAC and switched in or out depending on the protocol to parse the bytes in the packet. The CMACs are configurable by a processor to decode the incoming packets or frames as either FC "packets" or GbE packets." Chavan '197 at col. 5, l. 50 – col. 6, l. 16. "FIG. 3 is an illustration of an INP 310 implemented according to embodiments of the present invention. INP 310 may be composed of a single silicon integrated circuit die, which may be packaged inside a ceramic, metal, or plastic housing to form a unitary component. NP 310 includes a BCM 312 which is controlled by a flow identification and scheduler module 314 through a bus 328. The BCM 312 may be a multi-ported shared Random Access Memory (RAM) having memory locations mapped as passages, and operating at 224 Gbps. The scheduler 314 operates under the control of the APEX 316 acting through the bus 330, and maps the passages of crossbar locations inside the BCM 312. The scheduler 314 pre-processes received frames,

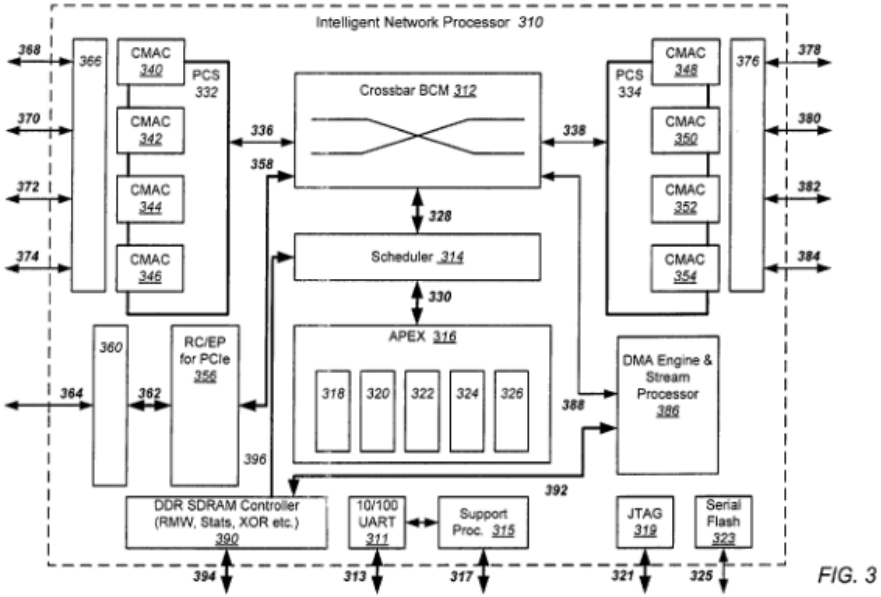
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	packets, and PCIe requests before they are acted upon by APEX 316. The scheduler 314 also parses received FC frames into correct sequence, and parses received GbE packets (in Internet Protocol (IP) format) into a Transport Control Protocol (TCP) flow. The APEX 316 preferably contains five core processors for executing firmware, denoted as processors 318, 320, 322, 324, and 326. The processors 318, 322, 324, and 326 may be core model ARC750 as provided by ARC, Inc., which preferably each have the capability of executing 32 bit Reduced Instruction Set Computer (RISC) firmware with a clock speed of 500 MHz, and may each include 32 KB of dedicated instruction memory (not shown)." Chavan '197 at col. 7, l. 49 – col. 8, l. 6. "The DMA engine and stream processor 386 is connected to the BCM 312 through bus 388, and is connected to the DDR SDRAM controller 390 through bus 392. The function of the controller 390 is to control the sending and receiving of data to and from an external DDR memory module (not shown) connected to the bus 394 of the controller 390. The controller 390 includes capabilities for providing RMW, statistics collection, XOR masking for data being transferred to or from the external DDR memory module. The data pathway through the bus 392 to the DMA engine and stream processor 386 and through the bus 388 to the BCM 312 allows DMA memory operations to occur directly with the controller 390 and its attached DDR memory module, without data flow through the APEX 316, for data being exchanged with other sources including the PCIe capabilities accessed through the bus 358 and RC/EP module 356, and the FC and GbE capabilities accessed through the buses 336 and 338 with the PCS modules 332 and 334. The ability to allow direct access for DMA operations through the BCM 312 with the DDR SDRAM controller 390 saves time that would be otherwise consumed by data being transferred into and out of the APEX 316. The controller 390 is also connected to the scheduler 314 through the bus 396. The controller 390 creates queues in the external DDR memory for storing data messages, and writes to and reads from such queues. The use of such queues, and the direct memory access to the crossbar BCM 312 (through the buses 388 and 392, controller 390, and DMA engine and stream processor 386) effectively increases the size of the crossbar BCM 312, and allows more switching pathways to be created than if the BCM 312 were used without the external DDR memory attached to connection 394." Chavan '197 at col. 8, l. 66 – col. 9, l. 29. "FIG. 6 is an illustration of an exemplary system employing an INP 610 according to embodiments of the present invention. Note that INP 610 is a simplified representation of the INP of FIG. 3, emphasizing the PCIe interface. FIG. 6 illustrates a so-called "single root" system configuration comprised of a single root complex (the host 614) and an INP 610 representing one of perhaps several endpoints. In the example of FIG. 6, the host is logically partitioned into multiple virtual servers, each running a separate "guest" OS 652. The

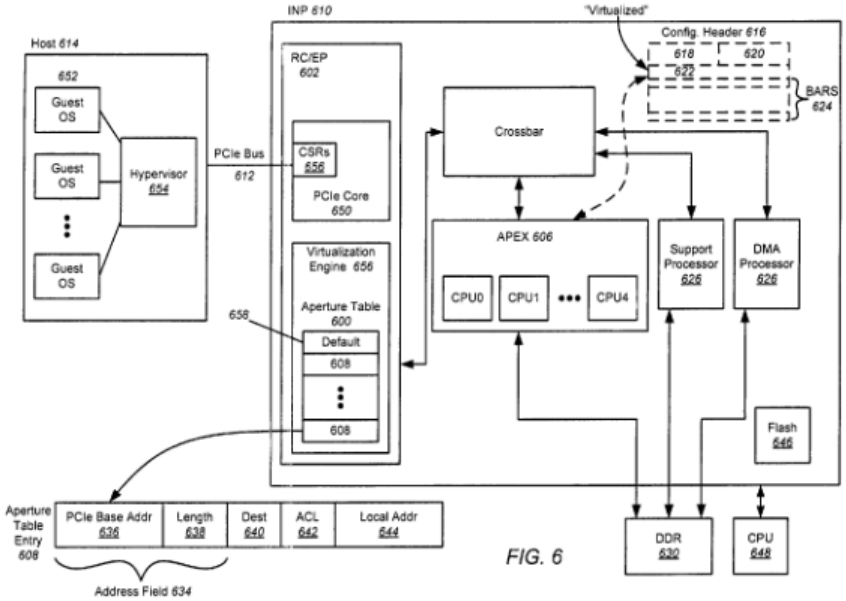
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	logical partitioning of the host 614 is managed by hypervisor software 654." Chavan '197 at col. 11, ll. 29–39. "FIG. 8 is an illustration of an exemplary system employing an INP 800 with virtualization logic 826 comprised of programmable configuration logic 802 according to embodiments of the present invention. In the alternative embodiment of FIG. 8, the configuration logic 802 may replace the aperture table described above, or may work together with the aperture table. Note that in this alternative embodiment there are no hardware configuration registers in the RC/EP 812 that terminate configuration cycles from the host. In the embodiment of FIG. 8, when PCIe transactions 804 are presented to the RC/EP 812, the headers are evaluated by the configuration logic 802 to determine the classification of the PCIe transaction (i.e. the type of cycle). Depending on the classification, the configuration logic 802 may take no further action with regard to the PCIe transaction, may direct the transaction to a processor via crossbar 810, or may perform an address translation. For example, if the configuration logic 802 determines that a discovery or configuration request (i.e. a configuration cycle) is being presented, the discovery or configuration request is passed through the crossbar 810 to the APEX 814. A free processor in the APEX 814 then executes firmware to respond to and terminate the configuration cycle by providing information and indications back to the host indicating that virtual PCIe devices have been discovered and configured. The particular processor and firmware performs translations from the configuration space of the requestor to the virtual configuration space of the virtual PCIe device being emulated." Chavan '197 at col. 16, ll. 6–33. "FIG. 9 is an illustration of how the common processing elements of MACs for FC 900 and MACs for GbE 902 can be integrated into an efficient CMAC 904 according to embodiments of the present invention. Conventionally, FC frames are received and deserialized by SERDES 906, and are then processed by a FC MAC 900. This processing typically includes 32-bit 8b10b decoding 908, detection of a SFD (a.k.a. a "k" character) and an EFD 910, CRC 912, generating packets from the FC frames 914, and buffering in a FIFO buffer 924. Similarly, GbE packets are received and deserialized by SERDES 916, and are then processed by a GbE MAC 902. This processing typically includes 8-bit 8b10b decoding 918, detection of a SFD and an EFD 920, CRC 922, and FIFO buffering 926. Because of the processing similarities, embodiments of the present invention integrate the data paths for both the FC and GbE protocols in a FC/GbE CMAC 904. Both protocols require SERDES 928, CRC 930 and FIFOs 932 for rate matching, and thus the logic for these tasks is shared in the CMAC 904. However, because the logic (state machine) to perform 8b10b decoding, to detect SFDs and EFDs for each protocol, and to packetize FC frames is different for the two protocols, this logic must be duplicated and switched in or out as

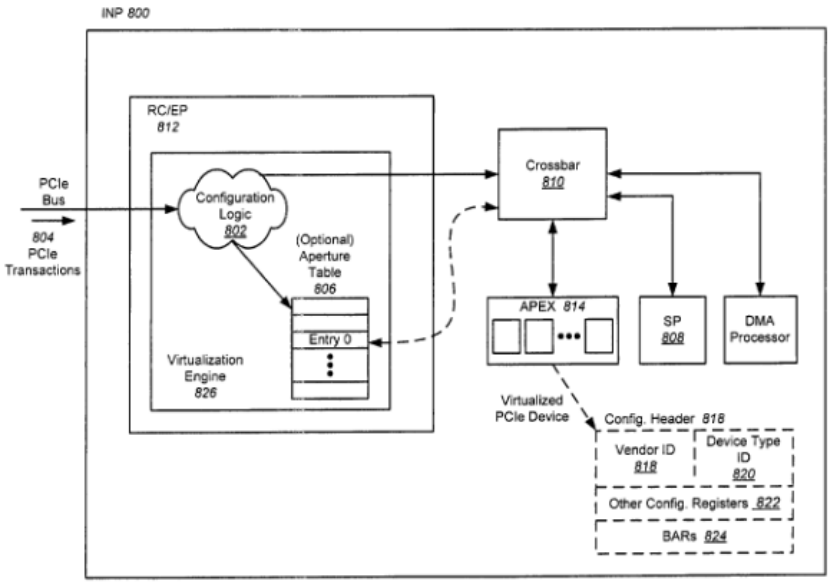
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	needed. To produce a FC/GbE CMAC, multiple bytes must be framed as packets (even though FC uses "frame" terminology). The bytes in a packet are well-defined in the FC and GbE standards. After the SFDs have been detected, logic specific to each protocol may be employed outside the CMAC and switched in or out depending on the protocol to parse the bytes in the packet. The CMACs are configurable by a processor to decode the incoming packets or frames as either FC "packets" or GbE packets." Chavan '197 at col. 17, ll. 22–53. "FIG. 10 illustrates an exemplary INP 1000 employing a hashing scheme for CMAC filtering according to embodiments of the present invention. Note that the representation of INP 1000 of FIG. 10 corresponds to the INP of FIG. 3, but is simplified to emphasize the hashing scheme. In the hashing scheme of FIG. 10, the hashing rules are made scalable without adding hardware, so each CMAC can have many more rules than their hardware registers would allow. In the embodiment of FIG. 10, the scheduler 1002 includes a hash engine 1004. Sophisticated hashing rules can be set for the hash engine 1004. For example, if a CMAC 1006 wants to accept any packet having one of 2 k destination MAC addresses, the hash engine 1004 can be configured such that for every one of the 2 k destination MAC addresses, a hash entry 1008 is created in off-chip DDR 1010. The DDR 1010 can store a large number of hash entries 1008 to achieve scalability. The hash entries 1008 are created at startup, during configuration, when a host configures the INP 1000 to receive the 2 k destination MAC addresses. After the hash entries 1008 are created, if a packet is subsequently received on CMAC 1006 containing one of the 2 k destination MAC addresses, it enters the BCM 1012 and is routed to the scheduler 1002. The scheduler 1002 will do a hash of the destination MAC address in the packet, and attempt to match it with one of the hash entries 1008 in DDR 1010. This is performed by a hardware state machine, so no firmware is required. Although external memory is involved, hardware prefetching is utilized, and thus it may be done at near wire speeds. If a match is found, the packet is then forwarded to one of the APEX processors 1014. On the other hand, if a packet is received that does not have one of the 2 k destination MAC addresses, a bit in the packet is set indicating a "hash miss." The packet is thereafter still forwarded to the APEX block 1014, but the APEX firmware will detect the hash miss bit and know that the packet was not to be received, so the packet is discarded." Chavan '197 at col. 18, ll. 7–42. "74. An intelligent network processor (INP) for connection to a protocol fabric, comprising: a plurality of port interfaces for connection to a protocol fabric, and adjustable to be selectively compatible with FC or GbE; a bidirectional crossbar module (BCM) coupled to receive messages from each of the plurality of port modules, and having crossbar passageways compatible with both FC and GbE messages;

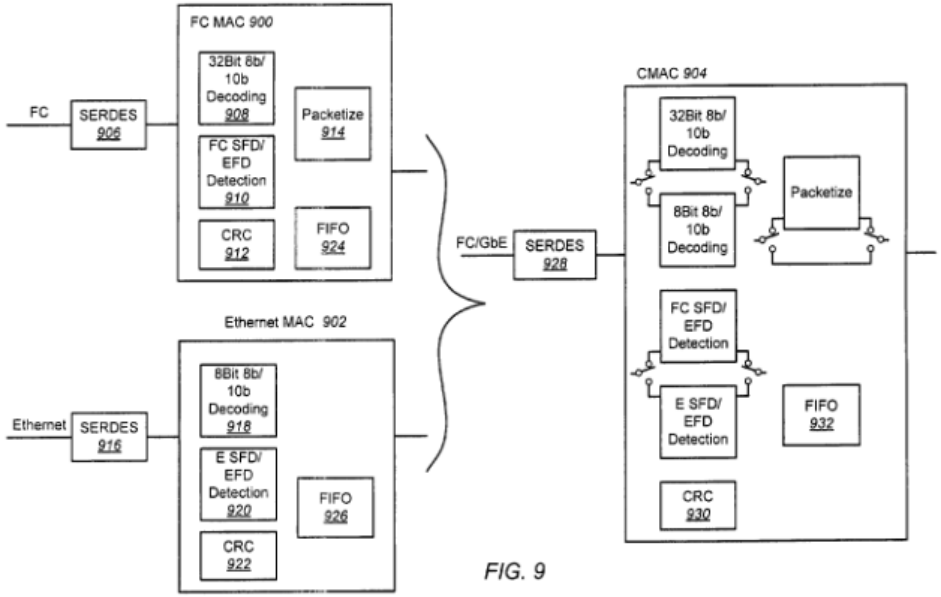
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	a DDR memory interface for connecting to an external DDR memory, for storing and retrieving data messages in queues contained in the DDR memory, and connected to send and receive data messages directly from the BCM; and an application processor module (APEX) coupled to the BCM for controlling the connections between the port interfaces and the DDR memory interface through the BCM, and enabling direct memory access operations between the DDR memory interface and the BCM without data messages passing through the APEX." Chavan '197 at col. 28, l. 60 – col. 29, l. 11. "75. A method for connecting a plurality of port interfaces to a protocol fabric, comprising: providing a plurality of port interfaces for connection to a protocol fabric, the plurality of port interfaces being adjustable to be selectively compatible with FC or GbE; receiving messages from each of the plurality of port interfaces through a bidirectional crossbar module (BCM) having crossbar passageways compatible with both FC and GbE messages; connecting to an external DDR memory for storing and retrieving data messages in queues contained in DDR memory, and for sending and receiving data messages directly from the BCM; and controlling the connections between the port interfaces and the DDR memory interface through the BCM, and enabling direct memory access operations between the DDR memory interface and the BCM without data messages passing through the APEX." Chavan '197 at col. 29, l. 12 – col. 30, l. 14.
[1d] at least one second processing element in communication with said first processing element;	Chavan '197 teaches "at least one second processing element in communication with said first processing element," including at least the following exemplary citations. For example, Chavan '197 teaches at least one second processing element in communication with said first processing element, by disclosing, as non–limiting examples, a scheduler that communicates with the processors within the APEX. The scheduler, the second processing element is in communication with the first processing element. *See, e.g.:*

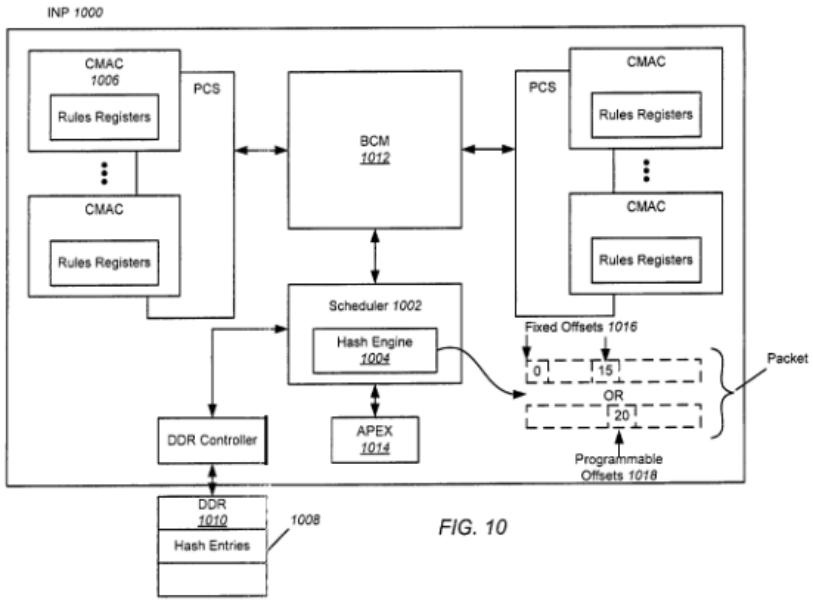
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan '197 at Figure 1.

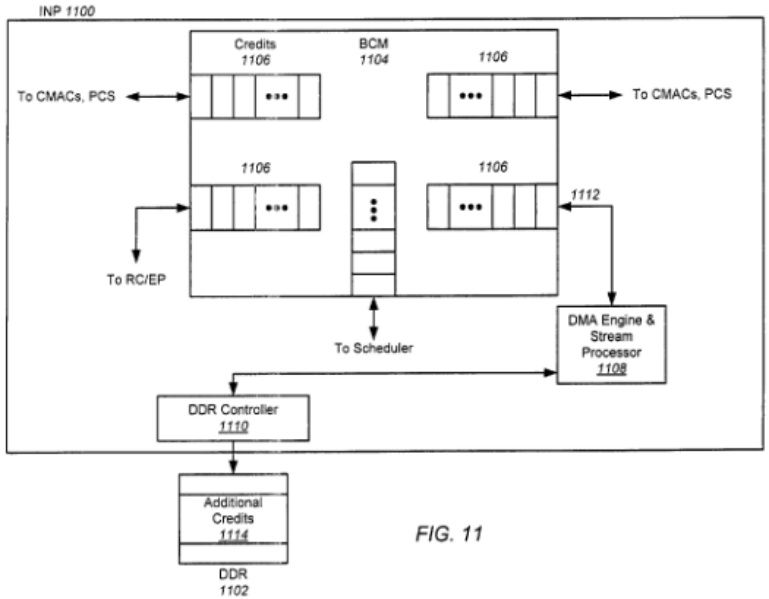
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan ’197 at Figure 3.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan '197 at Figure 6.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 8 Chavan ’197 at Figure 8.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan ’197 at Figure 9.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan ’197 at Figure 10.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 11 Chavan ’197 at Figure 11. “INPs are used in computer network systems for providing HBA, router, and switch capabilities for particular protocol networks connected to a host through a host interface. An example of an INP 110 is shown in FIG. 1, which may be formed on a single silicon integrated circuit die packaged inside a ceramic, metal, or plastic housing to form a unitary component. INP 110 includes a shared memory crossbar module 112 which is controlled by a flow identification/scheduler module 114 through a bus 128. The flow identification/scheduler module 114 operates under the control of the Application Processor (APEX) module 116 acting through the bus 130, and maps the passages of crossbar locations inside the shared memory crossbar module 112. The APEX 116 in the example of FIG. 1 contains ten core processors for executing firmware, denoted as processors 118-127. The shared memory crossbar module 112 is connected to Protocol Coordination Stage (PCS) modules 132 and 134 through buses 136 and 138, respectively. PCS module 132 is connected to and controls the operation of Media Access Controller (MAC) modules 140, 142, 144, and 146. The MAC modules 140, 142, 144, and 146 are identical hardware interfaces capable of each communicating using the FC protocol at 1 Gigabits per second (Gbps) or 2 Gbps speeds. The speed and protocol settings for the for the MAC modules 140, 142, 144, and 146 are controlled by the PCS 132, which operates in response to control signals from a bus (not shown)

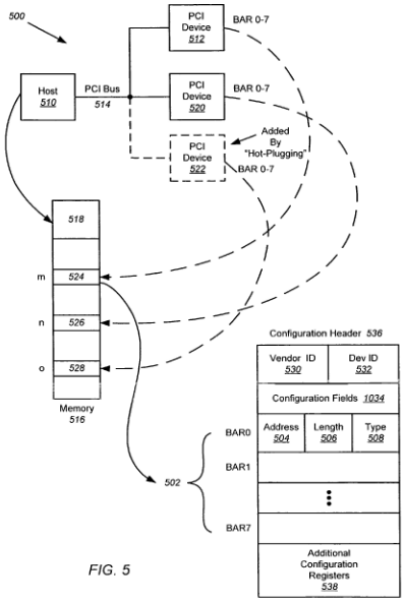
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	connecting the PCS 132 to the APEX 116. A parallel 10 Gbit (X Gbit) Medium-Independent Interface (XGMII) port 147 may be provided from the PCS 132, as determined by control signals originating from the APEX 116. The 10 Gb XGMII interface is not coupled to any of the four MAC modules." Chavan '197 at col. 1, ll. 23–53. "A Double Data Rate (DDR) Synchronous Dynamic Random Access Memory (SDRAM) controller 164 is connected to the scheduler 114 through bus 166. The function of the controller 164 is to control the sending and receiving of data to and from an external DDR memory module (not shown) connected to an external bus 168 of the controller. A support processor 115 handles certain processing tasks not otherwise covered by the APEX 116, and has an external connection 117. A Joint Test Action Group (JTAG) module 168 has an external connection 170, and is used for boundary scan testing of the hardware contained within INP 110. Serial flash memory 172 has an external connection 174, and is a non-volatile reprogrammable memory which may be loaded through the connection to store firmware for operation of APEX 116 and support processor 115." Chavan '197 at col. 2, ll. 14–28. "The present invention is directed to INPs for providing HBA, router, and switch capabilities on FC and GbE protocol networks connected to a single host or multiple hosts (as in the case of multi-root) through a PCIe host interface. The INP includes a Bi-directional Crossbar Module (BCM) which is controlled by a flow identification and scheduler module. The scheduler operates under the control of the APEX, and maps the passages of crossbar locations inside the BCM. The APEX contains multiple core processors for executing firmware. The BCM is connected to PCS modules, which are connected to and control the operation of Combined Media Access Controller (CMAC) modules capable of each communicating using multiple protocols. The speed and protocol settings for the CMAC modules are controlled by the PCS. The PCS also has trunking modes in which it coordinates the operation of the CMAC modules together as a single trunk. A combined SERDES is coupled to each of the CMACs and selectively provides, under control of signals coupled from the APEX, electronic signals in one of multiple protocols (e.g. the FC and GbE protocols), or trunking protocols." Chavan '197 at col. 3, ll. 28–47. "An INP may also utilize programmable configuration logic that may replace the aperture table described above, or may work together with the aperture table. In this embodiment, the headers of PCIe transactions are evaluated by the configuration logic to determine the classification of the PCIe transaction (i.e. the type of cycle). Depending on the classification, the configuration logic may take certain actions. For example, if the configuration logic determines that a discovery or configuration request (i.e. a configuration cycle) is being

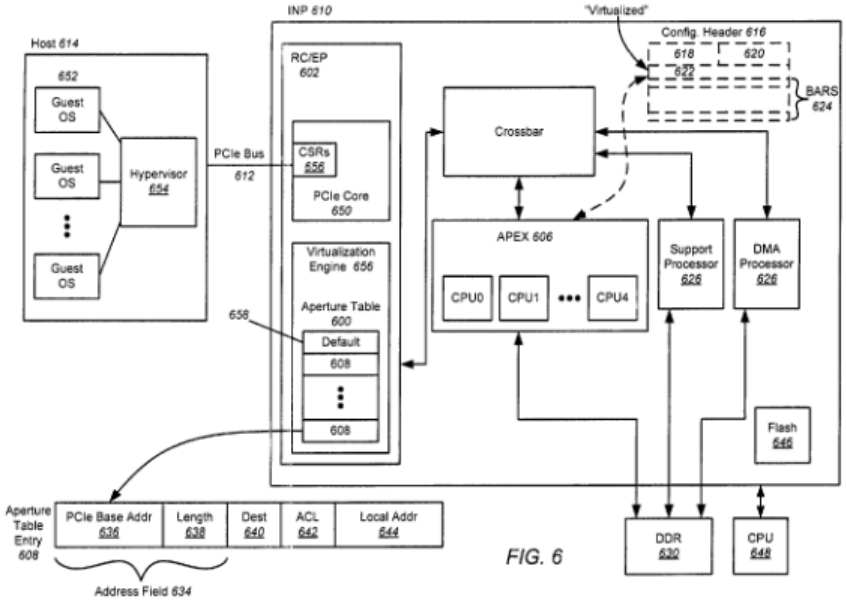
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	presented, the discovery or configuration request is passed through the crossbar to the APEX. A free processor in the APEX then executes firmware to respond to and terminate the configuration cycle by providing information and indications back to the host indicating that virtual PCIe devices have been discovered and configured. The particular processor and firmware performs translations from the configuration space of the requestor to the virtual configuration space of the virtual PCIe device being emulated. The common processing elements of FC and GbE MACs may be integrated into an efficient CMAC. Both protocols require SERDES, Cyclic Redundancy Checking (CRC) and First-In-First-Out (FIFO) buffers for rate matching, and thus the logic for these tasks is shared in the CMAC. However, because the logic (state machine) to perform 8b10b decoding, to detect Start of Frame Delimiters (SFDs) and End of Frame Delimiters (EFDs) for each protocol, and to packetize FC frames is different for the two protocols, this logic is duplicated and switched in or out as needed. To produce a FC/GbE CMAC, multiple bytes must be framed as packets. After the SFDs have been detected, logic specific to each protocol may be employed outside the CMAC and switched in or out depending on the protocol to parse the bytes in the packet. The CMACs are configurable by a processor to decode the incoming packets or frames as either FC "packets" or GbE packets." Chavan '197 at col. 5, l. 50 – col. 6, l. 16. "FIG. 3 is an illustration of an INP 310 implemented according to embodiments of the present invention. INP 310 may be composed of a single silicon integrated circuit die, which may be packaged inside a ceramic, metal, or plastic housing to form a unitary component. NP 310 includes a BCM 312 which is controlled by a flow identification and scheduler module 314 through a bus 328. The BCM 312 may be a multi-ported shared Random Access Memory (RAM) having memory locations mapped as passages, and operating at 224 Gbps. The scheduler 314 operates under the control of the APEX 316 acting through the bus 330, and maps the passages of crossbar locations inside the BCM 312. The scheduler 314 pre-processes received frames, packets, and PCIe requests before they are acted upon by APEX 316. The scheduler 314 also parses received FC frames into correct sequence, and parses received GbE packets (in Internet Protocol (IP) format) into a Transport Control Protocol (TCP) flow. The APEX 316 preferably contains five core processors for executing firmware, denoted as processors 318, 320, 322, 324, and 326. The processors 318, 322, 324, and 326 may be core model ARC750 as provided by ARC, Inc., which preferably each have the capability of executing 32 bit Reduced Instruction Set Computer (RISC) firmware with a clock speed of 500 MHz, and may each include 32 KB of dedicated instruction memory (not shown)." Chavan '197 at col. 7, l. 49 – col. 8, l. 6.

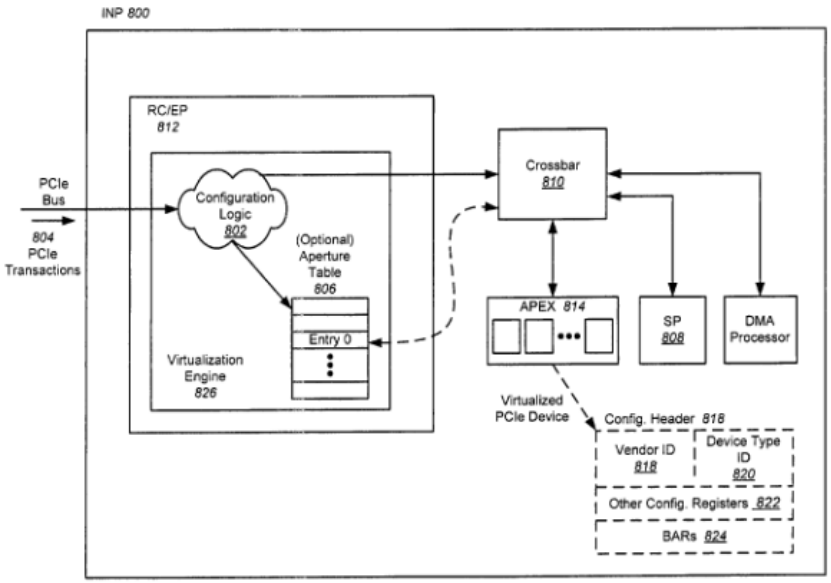
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	"FIG. 6 is an illustration of an exemplary system employing an INP 610 according to embodiments of the present invention. Note that INP 610 is a simplified representation of the INP of FIG. 3, emphasizing the PCIe interface. FIG. 6 illustrates a so-called "single root" system configuration comprised of a single root complex (the host 614) and an INP 610 representing one of perhaps several endpoints. In the example of FIG. 6, the host is logically partitioned into multiple virtual servers, each running a separate "guest" OS 652. The logical partitioning of the host 614 is managed by hypervisor software 654." Chavan '197 at col. 11, ll. 29–39. "FIG. 8 is an illustration of an exemplary system employing an INP 800 with virtualization logic 826 comprised of programmable configuration logic 802 according to embodiments of the present invention. In the alternative embodiment of FIG. 8, the configuration logic 802 may replace the aperture table described above, or may work together with the aperture table. Note that in this alternative embodiment there are no hardware configuration registers in the RC/EP 812 that terminate configuration cycles from the host. In the embodiment of FIG. 8, when PCIe transactions 804 are presented to the RC/EP 812, the headers are evaluated by the configuration logic 802 to determine the classification of the PCIe transaction (i.e. the type of cycle). Depending on the classification, the configuration logic 802 may take no further action with regard to the PCIe transaction, may direct the transaction to a processor via crossbar 810, or may perform an address translation. For example, if the configuration logic 802 determines that a discovery or configuration request (i.e. a configuration cycle) is being presented, the discovery or configuration request is passed through the crossbar 810 to the APEX 814. A free processor in the APEX 814 then executes firmware to respond to and terminate the configuration cycle by providing information and indications back to the host indicating that virtual PCIe devices have been discovered and configured. The particular processor and firmware performs translations from the configuration space of the requestor to the virtual configuration space of the virtual PCIe device being emulated." Chavan '197 at col. 16, ll. 6–33. "FIG. 10 illustrates an exemplary INP 1000 employing a hashing scheme for CMAC filtering according to embodiments of the present invention. Note that the representation of INP 1000 of FIG. 10 corresponds to the INP of FIG. 3, but is simplified to emphasize the hashing scheme. In the hashing scheme of FIG. 10, the hashing rules are made scalable without adding hardware, so each CMAC can have many more rules than their hardware registers would allow. In the embodiment of FIG. 10, the scheduler 1002 includes a hash engine 1004. Sophisticated hashing rules can be set for the hash engine 1004. For example, if a CMAC 1006 wants to accept any packet having one of 2 k destination MAC addresses, the hash engine 1004 can be configured such that for every one of the 2 k destination MAC addresses, a hash entry 1008 is created in off-chip DDR 1010. The DDR 1010 can store a large number of hash entries 1008 to achieve scalability. The

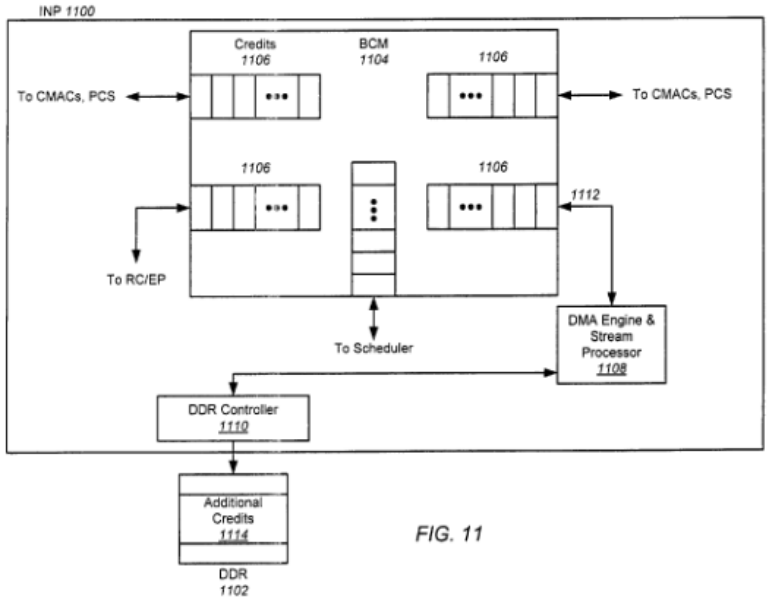
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	hash entries 1008 are created at startup, during configuration, when a host configures the INP 1000 to receive the 2 k destination MAC addresses. After the hash entries 1008 are created, if a packet is subsequently received on CMAC 1006 containing one of the 2 k destination MAC addresses, it enters the BCM 1012 and is routed to the scheduler 1002. The scheduler 1002 will do a hash of the destination MAC address in the packet, and attempt to match it with one of the hash entries 1008 in DDR 1010. This is performed by a hardware state machine, so no firmware is required. Although external memory is involved, hardware prefetching is utilized, and thus it may be done at near wire speeds. If a match is found, the packet is then forwarded to one of the APEX processors 1014. On the other hand, if a packet is received that does not have one of the 2 k destination MAC addresses, a bit in the packet is set indicating a "hash miss." The packet is thereafter still forwarded to the APEX block 1014, but the APEX firmware will detect the hash miss bit and know that the packet was not to be received, so the packet is discarded." Chavan '197 at col. 18, ll. 7–42. "74. An intelligent network processor (INP) for connection to a protocol fabric, comprising: a plurality of port interfaces for connection to a protocol fabric, and adjustable to be selectively compatible with FC or GbE; a bidirectional crossbar module (BCM) coupled to receive messages from each of the plurality of port modules, and having crossbar passageways compatible with both FC and GbE messages; a DDR memory interface for connecting to an external DDR memory, for storing and retrieving data messages in queues contained in the DDR memory, and connected to send and receive data messages directly from the BCM; and an application processor module (APEX) coupled to the BCM for controlling the connections between the port interfaces and the DDR memory interface through the BCM, and enabling direct memory access operations between the DDR memory interface and the BCM without data messages passing through the APEX." Chavan '197 at col. 28, l. 60 – col. 29, l. 11. "75. A method for connecting a plurality of port interfaces to a protocol fabric, comprising: providing a plurality of port interfaces for connection to a protocol fabric, the plurality of port interfaces being adjustable to be selectively compatible with FC or GbE; receiving messages from each of the plurality of port interfaces through a bidirectional crossbar module (BCM) having crossbar passageways compatible with both FC and GbE messages;

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	connecting to an external DDR memory for storing and retrieving data messages in queues contained in DDR memory, and for sending and receiving data messages directly from the BCM; and controlling the connections between the port interfaces and the DDR memory interface through the BCM, and enabling direct memory access operations between the DDR memory interface and the BCM without data messages passing through the APEX." Chavan '197 at col. 29, l. 12 – col. 30, l. 14.
[1e] a reservation register implemented in hardware storing a value indicative of available space in said input queue, said reservation register accessible by both said first processing element and said at least one second processing element;	Chavan '197 teaches "a reservation register implemented in hardware storing a value indicative of available space in said input queue, said reservation register accessible by both said first processing element and said at least one second processing element," including at least the following exemplary citations. For example, Chavan '197 teaches a reservation register implemented in hardware storing a value indicative of available space in said input queue, said reservation register accessible by both said first processing element and said at least one second processing element, by disclosing, as non–limiting examples, credits that represent the number of free buffers. Defendants have proposed a construction that "register" means "set of flip–flops that store a set of related bits." Chavan '197 teaches a reservation register that is a set of flip-flops that store a set of related bits. *See, e.g.*, Chavan '197 at 10:39–42 ("The PCI standard specifies that configuration space shall be allocated for each PCI device in the form of configuration registers that consume hardware resources."). For example, the reservation registers identified above is explained and described as a register and the description of its behavior is consistent with a set of flip-flops that store a set of related bits. *See id.* Chavan '197 is a multiprocessor system that uses registers. A POSITA reading Chavan '197 would have understood the description of the credit counter to be a set of flip-flops that store a set of related bits in view of the relevant disclosures, and further because that is how hardware systems and registers in particular had been designed and implemented for decades. *See, e.g.:*

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan '197 at Figure 5.

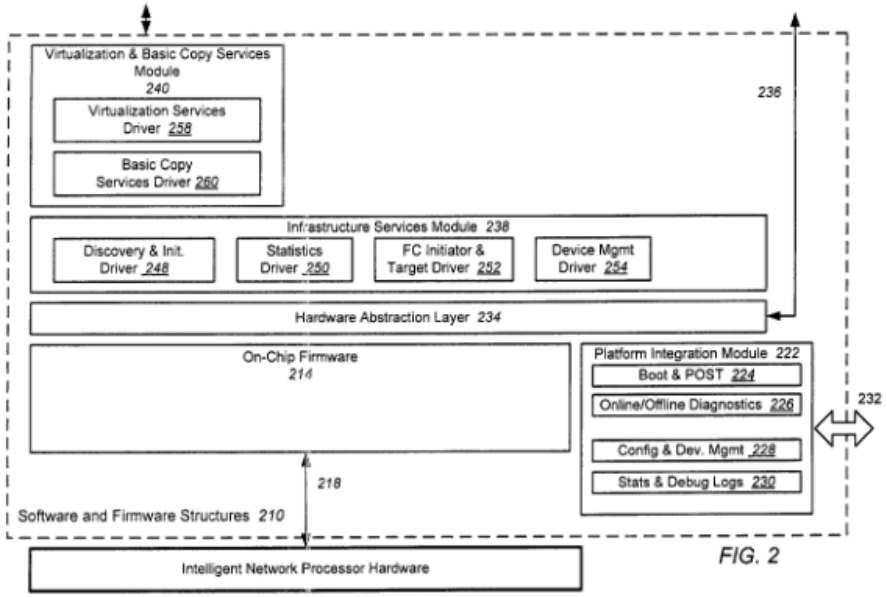
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan '197 at Figure 6.

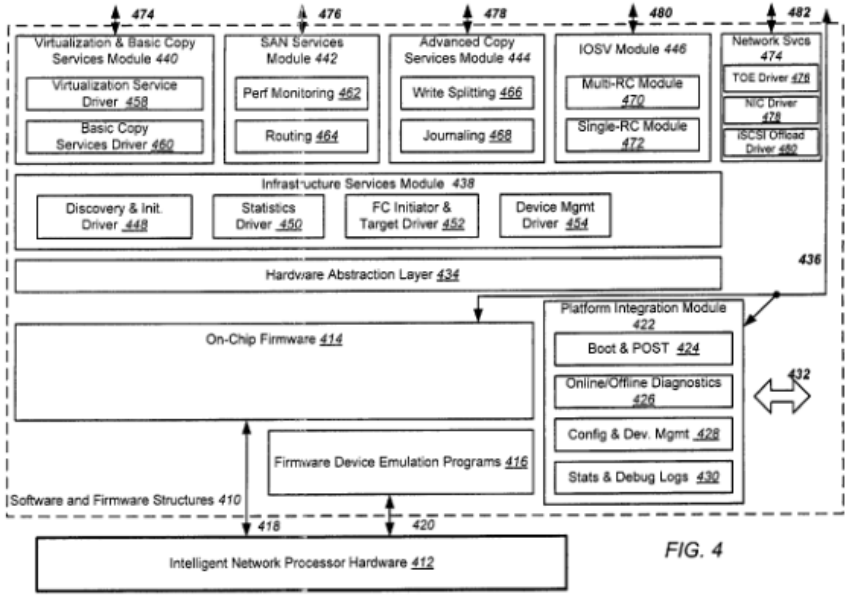
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 8 Chavan ’197 at Figure 8.

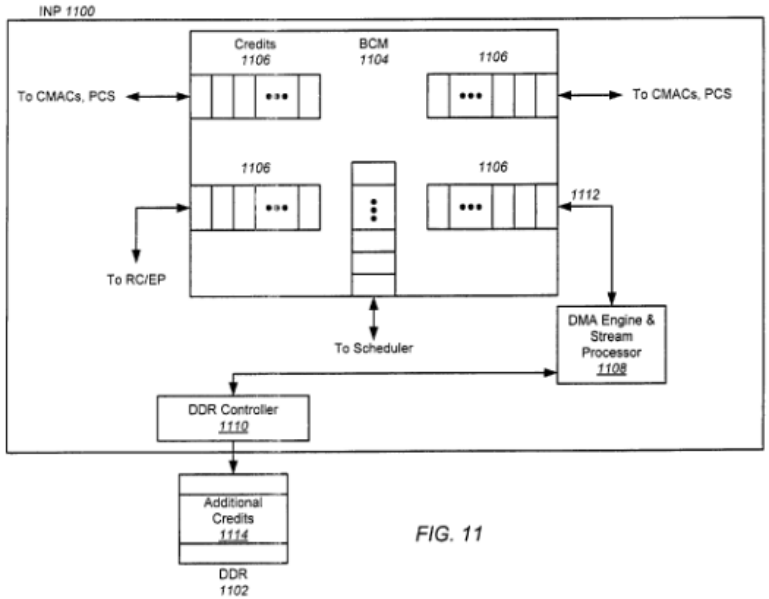
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 11 Chavan ’197 at Figure 11. “Conventional BCMs utilize the concept of credits for their ports. Credits are essentially the number of free buffers at a particular port for storing frames or packets. One credit is consumed when a packet is received, and one credit is released when a packet is sent it out of the crossbar to another port (or to the APEX block). Normally the credits for each of the ports are part of the crossbar, in limited on-chip memory. The INP may also employ crossbar credit extensions wherein multiple ports may be given an extended number of credits using off-chip memory. To accomplish this, the external DDR memory is treated as another port on the crossbar. When crossbar logic in the BCM determines that more credits are needed for a particular port, the crossbar logic will make a request to the DMA block, which has access to the off-chip DDR memory through the DDR controller, to effectively gain access to additional credits stored in the DDR memory.” Chavan ’197 at col. 6, l. 54 – col. 7, l. 2. “As shown in the example of FIG. 5, in a conventional PCI system 500, a host 510 may be connected to multiple PCI devices (although only two PCIe devices 512 and 520 are shown) through a PCI bus 514. The PCI standard specifies that configuration space shall be allocated for each PCI device in the form of configuration registers that consume hardware resources. In particular, each PCI device is assigned one or more mutually-exclusive blocks of memory in the host's memory-mapped memory space 516. In the example

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	of FIG. 5, host 510 has been assigned memory space 518, PCI device 512 has been assigned memory space 524 (designated by the letter "m"), and PCI device 520 has been assigned memory space 526 (designated by the letter "n"). According to the PCI standard, as part of the configuration space allocated to each PCI device, each PCI device may be allocated up to eight BARs 502 in a configuration header 536, referred to as BAR0-BAR7 in FIG. 5. Each BAR contains the start address 504 of a block of memory space in memory 516 assigned to a function of the PCI device, and the length 506 and type of memory space 508 (memory or I/O) required by the function. The start address 504 of the BAR enables a memory or I/O address decoder in the PCI device to detect device accesses whose addresses fall within the block of memory space specified by the BAR. Additionally, other configuration registers 538 (including doorbells and status registers) may be allocated for each PCI device and assigned memory space such that once mapped, a host and the PCI device can communicate using that memory space. During configuration of the PCI system, the host 510 performs a configuration cycle where the host first discovers what PCI devices exist on the PCI bus 514, and then configures the configuration registers for those PCI devices. To implement the discovery process, a host's configuration software sends a discovery request out over the PCI bus 514, which reads a subset of each PCI device's configuration registers (e.g. vendor IDentification (ID) register 530 and device ID register 532) in configuration header 536 to determine the presence of the PCI device and its device type. The host 510 also accesses other configuration registers 534 in the PCI device to determine how many blocks of memory the PCI device requires. To configure a particular PCI device, the host sends a configuration request to assign memory space to the PCI device by assigning an address to the BARs of that PCI device. Note that the length field 506 and the type field 508 of each BAR are generally fixed, and only the address field 504 is programmable. When a particular PCI device has been discovered and configured, the PCI device responds back to the host, resulting in the termination of the configuration cycle for that PCI device." Chavan '197 at col. 10, l. 36 – col. 11, l. 17. "Conventional BCMs utilize the concept of credits for their ports. Credits are essentially the number of free buffers at a particular port. For example, if a port on a BCM has eight credits, eight frames of data could be received into that port and stored in the buffers without having to send an acknowledgement. One credit is consumed when a packet is received, and one credit is released when a packet is sent it out of the crossbar through that port. Normally the credits for each of the ports are part of the crossbar, in on-chip memory. However, on-chip memory is expensive, and thus traditionally the number of credits allocated to the ports on a BCM has been limited. This is sufficient for short-haul communications, but when using the links for long-haul purposes such as distance of 100 km, many packets may be in the pipe, so more credits are needed per

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	port. Increasing the size of the on-chip crossbar is expensive. However, external memory has traditionally not been used to store credits because it would be too slow to control the external memory." Chavan '197 at col. 19, l. 66 – col. 20, l. 16. "FIG. 11 is an illustration of an INP 1100 employing crossbar credit extensions according to embodiments of the present invention. In FIG. 11, a BCM 1104 has multiple ports, each port having a number of credits 1106 for buffering frames as they pass through the BCM 1104. The credit extensions of the present invention extend the number credits available to each port on the BCM 1104 using off-chip memory 1102. As a result, on a port by port basis, more credits can be configured than are available in the crossbar memory 1104. The external DDR memory 1102 is treated as another port on the crossbar 1104, or just an extension of the crossbar. When crossbar logic in the BCM 1104 determines that more credits are needed for a particular port, the crossbar logic will make a request to the DMA block 1108, which has access to the off-chip DDR memory 1102 through the DDR controller 1110, to effectively gain access to additional credits 1114. The DMA engine 1108 treats the off-chip DDR memory 1102 as though it were on-chip memory. When more credits are needed, the off-chip DDR 1102 can be accessed through a port 1112 on the crossbar 1104 and through the DMA engine 1108 to store frames in the DDR 1102 and effectively consume additional credits. Note that this is performed using gates, because the message frames are received so fast that processors would not be able to selectively decide what frames to put in DDR 1102 and which ones to receive from DDR. Instead, the DMA engine 1108 does this work." Chavan '197 at col. 20, ll. 17–42.
[1f] computer storage storing instructions, which when executed by said at least one second processing element:	Chavan '197 teaches computer storage storing instructions executed by said at least one second processing element, including at least the following exemplary citations. For example, Chavan '197 teaches computer storage storing instructions that are executed by said at least one second processing element, by disclosing, as non–limiting examples, software and firmware structures including memory. The intelligent network processor APEX hardware and scheduler execute instructions stored in the software and firmware structures. *See, e.g.:*

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 2 Chavan ’197 at Figure 2.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 4 Chavan ’197 at Figure 4.

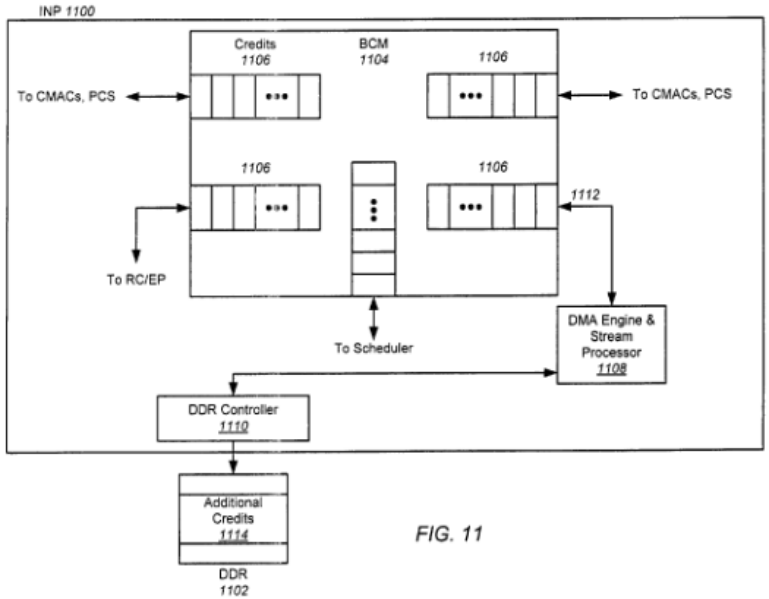
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 11 Chavan ’197 at Figure 11. “FIG. 2 is an illustration of firmware and software structures 210 for use with the INP of FIG. 1 (shown as an INP hardware block in FIG. 2). The structures 210 serve to provide the computer codes for activating the INP hardware block, and for interacting with a host Operating System (OS) software structure (not shown) such as Linux or Microsoft Windows. On-chip firmware 214 may be downloaded into the INP, and are for execution by the APEX module and support processor of FIG. 1. The firmware 214 interacts with the hardware through connection 218.” Chavan ’197 at col. 2, ll. 29–38. “The on-chip firmware 214 and PIM 222 may be accessed by an external software operating system through a hardware abstraction layer 234, or may be directly accessed through a connection 236. The layer 234 is a part of a software driver program which also includes an infrastructure services module 238 and a virtualization and basic copy services module 240. The software driver including the layer 234 and modules 238 and 240 is to be executed as a program on an external host computer processor (not shown) under the control of a host operating system software structure (not shown).” Chavan ’197 at col. 2, ll. 48–57. “The INP may be configured using firmware to emulate multiple virtual PCIe devices. Instead of providing hardware configuration registers for each virtual PCIe device capable of being emulated, the INP presents only a single set of eight hardware Configuration Space Registers (CSRs) (including BARs) to a host within

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	its PCIe core, and responds to and terminates the configuration cycle in a manner similar to conventional PCI devices." Chavan '197 at col. 4, ll. 10–17. "Conventional BCMs utilize the concept of credits for their ports. Credits are essentially the number of free buffers at a particular port for storing frames or packets. One credit is consumed when a packet is received, and one credit is released when a packet is sent it out of the crossbar to another port (or to the APEX block). Normally the credits for each of the ports are part of the crossbar, in limited on-chip memory. The INP may also employ crossbar credit extensions wherein multiple ports may be given an extended number of credits using off-chip memory. To accomplish this, the external DDR memory is treated as another port on the crossbar. When crossbar logic in the BCM determines that more credits are needed for a particular port, the crossbar logic will make a request to the DMA block, which has access to the off-chip DDR memory through the DDR controller, to effectively gain access to additional credits stored in the DDR memory." Chavan '197 at col. 6, l. 54 – col. 7, l. 2. "FIG. 4 is an illustration of the firmware and software structures 410 for use with the INP of FIG. 3, as implemented according to embodiments of the present invention. The structures 410 serve to provide the computer codes for activating the INP hardware 412, and for interacting with a host operating system software structure (not shown) such as Linux or Microsoft Windows. The hardware 412 is preferably the same as the INP of FIG. 3. On-chip firmware 414 and firmware device emulation programs 416 may be downloaded into the hardware 412, and are for execution by the APEX and support processor of FIG. 3. The firmware 414 and programs 416 interact with the hardware 412 through connections 418 and 420, respectively." Chavan '197 at col. 9, ll. 51–63. "The on-chip firmware 414 and PIM 422 may be accessed by an external software operating system through a hardware abstraction layer 434, or may be directly accessed through a connection 436. The layer 434 is a part of a software driver program which also includes an infrastructure services module 438, a virtualization and basic copy services module 440, a Storage Area Network (SAN) services module 442, an advanced copy services module 444, an Input/Output Storage Virtualization (IOSV) module 446, and a network services module 474. The software drivers including the layer 434 and modules 438, 440, 442, 444, 446 and 474 are to be executed as a program on an external host computer processor (not shown) under the control of a host operating system software structure (not shown)." Chavan '197 at col. 10, ll. 4–17. "As shown in the example of FIG. 5, in a conventional PCI system 500, a host 510 may be connected to multiple PCI devices (although only two PCIe devices 512 and 520 are shown) through a PCI bus 514. The PCI standard specifies that configuration space shall be allocated for each PCI device in the form of

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	configuration registers that consume hardware resources. In particular, each PCI device is assigned one or more mutually-exclusive blocks of memory in the host's memory-mapped memory space 516. In the example of FIG. 5, host 510 has been assigned memory space 518, PCI device 512 has been assigned memory space 524 (designated by the letter "m"), and PCI device 520 has been assigned memory space 526 (designated by the letter "n"). According to the PCI standard, as part of the configuration space allocated to each PCI device, each PCI device may be allocated up to eight BARs 502 in a configuration header 536, referred to as BAR0-BAR7 in FIG. 5. Each BAR contains the start address 504 of a block of memory space in memory 516 assigned to a function of the PCI device, and the length 506 and type of memory space 508 (memory or I/O) required by the function. The start address 504 of the BAR enables a memory or I/O address decoder in the PCI device to detect device accesses whose addresses fall within the block of memory space specified by the BAR. Additionally, other configuration registers 538 (including doorbells and status registers) may be allocated for each PCI device and assigned memory space such that once mapped, a host and the PCI device can communicate using that memory space. During configuration of the PCI system, the host 510 performs a configuration cycle where the host first discovers what PCI devices exist on the PCI bus 514, and then configures the configuration registers for those PCI devices. To implement the discovery process, a host's configuration software sends a discovery request out over the PCI bus 514, which reads a subset of each PCI device's configuration registers (e.g. vendor IDentification (ID) register 530 and device ID register 532) in configuration header 536 to determine the presence of the PCI device and its device type. The host 510 also accesses other configuration registers 534 in the PCI device to determine how many blocks of memory the PCI device requires. To configure a particular PCI device, the host sends a configuration request to assign memory space to the PCI device by assigning an address to the BARs of that PCI device. Note that the length field 506 and the type field 508 of each BAR are generally fixed, and only the address field 504 is programmable. When a particular PCI device has been discovered and configured, the PCI device responds back to the host, resulting in the termination of the configuration cycle for that PCI device." Chavan '197 at col. 10, l. 36 – col. 11, l. 17. "After the aperture table 600 has been populated with entries and the configuration phase is complete, the hypervisor 654 is bypassed, and subsequent data transfers occur directly between the drivers in the guest OSs 652 and the INP 610 through read and write commands. Read or write requests from a guest OS 652 are routed to the aperture table 600, and logic within the RC/EP module 602 applies the addresses of the requests to the aperture table and compares them with entries representing the BARs for that guest OS to determine if there is an entry having an address range (i.e. a "bin") encompassing the address of the request. If a match is

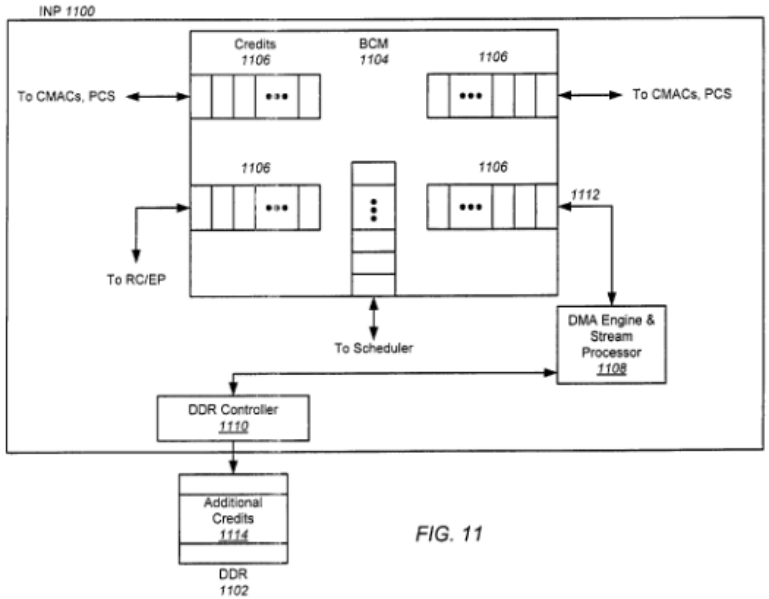
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	found, the logic in the RC/EP module 602 performs the action associated with the enabled rules for the matching entry. For example, the entry may result in no further action being taken with regard to the request, or may result in an address translation. Alternatively, the entry may cause the logic to direct the request to one of the processors in APEX 606 via crossbar 610. A free processor in APEX 606 then executes the firmware necessary to emulate the performance of the virtual PCIe device associated with that particular entry and respond to the host 614 as though it were a physical PCIe device. The firmware may be stored in on-chip flash memory or off-chip memory." Chavan '197 at col. 13, l. 48 – col. 14, l. 3. "The stored firmware routines executed by a processor in the INP 610 are preferably the same as the firmware device emulation programs 416 of FIG. 4. Note that unlike a hardware configuration header in which some of the fields are fixed (e.g. the length and type fields of each BAR), a processor and associated firmware can emulate the virtual configuration header 616 of a virtual PCIe device such that all of the fields can be programmable. In general, the processor and firmware may allocate additional configuration registers 622 and BARs 624 to the virtual PCIe device, up to the limits defined by the PCIe standard." Chavan '197 at col. 14, ll. 4–14. "Conventional BCMs utilize the concept of credits for their ports. Credits are essentially the number of free buffers at a particular port. For example, if a port on a BCM has eight credits, eight frames of data could be received into that port and stored in the buffers without having to send an acknowledgement. One credit is consumed when a packet is received, and one credit is released when a packet is sent it out of the crossbar through that port. Normally the credits for each of the ports are part of the crossbar, in on-chip memory. However, on-chip memory is expensive, and thus traditionally the number of credits allocated to the ports on a BCM has been limited. This is sufficient for short-haul communications, but when using the links for long-haul purposes such as distance of 100 km, many packets may be in the pipe, so more credits are needed per port. Increasing the size of the on-chip crossbar is expensive. However, external memory has traditionally not been used to store credits because it would be too slow to control the external memory." Chavan '197 at col. 19, l. 66 – col. 20, l. 16. "FIG. 11 is an illustration of an INP 1100 employing crossbar credit extensions according to embodiments of the present invention. In FIG. 11, a BCM 1104 has multiple ports, each port having a number of credits 1106 for buffering frames as they pass through the BCM 1104. The credit extensions of the present invention extend the number credits available to each port on the BCM 1104 using off-chip memory 1102. As a result, on a port by port basis, more credits can be configured than are available in the crossbar memory 1104. The external DDR memory 1102 is treated as another port on the crossbar 1104, or just an extension of the crossbar. When crossbar logic in the BCM 1104 determines that more credits are needed for a particular port,

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	the crossbar logic will make a request to the DMA block 1108, which has access to the off-chip DDR memory 1102 through the DDR controller 1110, to effectively gain access to additional credits 1114. The DMA engine 1108 treats the off-chip DDR memory 1102 as though it were on-chip memory. When more credits are needed, the off-chip DDR 1102 can be accessed through a port 1112 on the crossbar 1104 and through the DMA engine 1108 to store frames in the DDR 1102 and effectively consume additional credits. Note that this is performed using gates, because the message frames are received so fast that processors would not be able to selectively decide what frames to put in DDR 1102 and which ones to receive from DDR. Instead, the DMA engine 1108 does this work." Chavan '197 at col. 20, ll. 17–42.
[1g] accesses said reservation register and reads said stored value;	Chavan '197 teaches "accesses said reservation register and reads said stored value," including at least the following exemplary citations. For example, Chavan '197 teaches that the second processing element executes instructions that causes it to access said reservation register and reads said stored value, by disclosing, as non–limiting examples, accessing the credits that represent the number of free buffers. The scheduler accesses and reads the stored value in the reservation register. *See, e.g.:*

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 11 Chavan ’197 at Figure 11. “Conventional BCMs utilize the concept of credits for their ports. Credits are essentially the number of free buffers at a particular port for storing frames or packets. One credit is consumed when a packet is received, and one credit is released when a packet is sent it out of the crossbar to another port (or to the APEX block). Normally the credits for each of the ports are part of the crossbar, in limited on-chip memory. The INP may also employ crossbar credit extensions wherein multiple ports may be given an extended number of credits using off-chip memory. To accomplish this, the external DDR memory is treated as another port on the crossbar. When crossbar logic in the BCM determines that more credits are needed for a particular port, the crossbar logic will make a request to the DMA block, which has access to the off-chip DDR memory through the DDR controller, to effectively gain access to additional credits stored in the DDR memory.” Chavan ’197 at col. 6, l. 54 – col. 7, l. 2. “As shown in the example of FIG. 5, in a conventional PCI system 500, a host 510 may be connected to multiple PCI devices (although only two PCIe devices 512 and 520 are shown) through a PCI bus 514. The PCI standard specifies that configuration space shall be allocated for each PCI device in the form of configuration registers that consume hardware resources. In particular, each PCI device is assigned one or more mutually-exclusive blocks of memory in the host's memory-mapped memory space 516. In the example

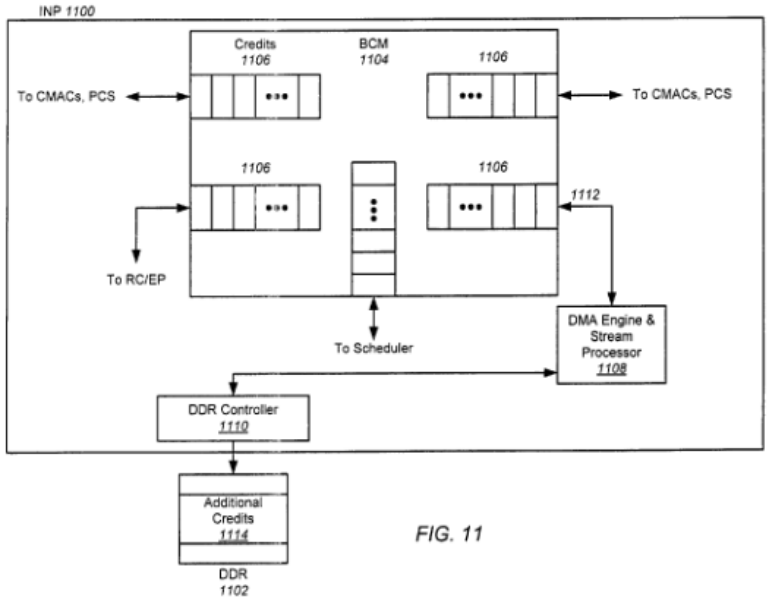
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	of FIG. 5, host 510 has been assigned memory space 518, PCI device 512 has been assigned memory space 524 (designated by the letter "m"), and PCI device 520 has been assigned memory space 526 (designated by the letter "n"). According to the PCI standard, as part of the configuration space allocated to each PCI device, each PCI device may be allocated up to eight BARs 502 in a configuration header 536, referred to as BAR0-BAR7 in FIG. 5. Each BAR contains the start address 504 of a block of memory space in memory 516 assigned to a function of the PCI device, and the length 506 and type of memory space 508 (memory or I/O) required by the function. The start address 504 of the BAR enables a memory or I/O address decoder in the PCI device to detect device accesses whose addresses fall within the block of memory space specified by the BAR. Additionally, other configuration registers 538 (including doorbells and status registers) may be allocated for each PCI device and assigned memory space such that once mapped, a host and the PCI device can communicate using that memory space. During configuration of the PCI system, the host 510 performs a configuration cycle where the host first discovers what PCI devices exist on the PCI bus 514, and then configures the configuration registers for those PCI devices. To implement the discovery process, a host's configuration software sends a discovery request out over the PCI bus 514, which reads a subset of each PCI device's configuration registers (e.g. vendor IDentification (ID) register 530 and device ID register 532) in configuration header 536 to determine the presence of the PCI device and its device type. The host 510 also accesses other configuration registers 534 in the PCI device to determine how many blocks of memory the PCI device requires. To configure a particular PCI device, the host sends a configuration request to assign memory space to the PCI device by assigning an address to the BARs of that PCI device. Note that the length field 506 and the type field 508 of each BAR are generally fixed, and only the address field 504 is programmable. When a particular PCI device has been discovered and configured, the PCI device responds back to the host, resulting in the termination of the configuration cycle for that PCI device." Chavan '197 at col. 10, l. 36 – col. 11, l. 17. "Conventional BCMs utilize the concept of credits for their ports. Credits are essentially the number of free buffers at a particular port. For example, if a port on a BCM has eight credits, eight frames of data could be received into that port and stored in the buffers without having to send an acknowledgement. One credit is consumed when a packet is received, and one credit is released when a packet is sent it out of the crossbar through that port. Normally the credits for each of the ports are part of the crossbar, in on-chip memory. However, on-chip memory is expensive, and thus traditionally the number of credits allocated to the ports on a BCM has been limited. This is sufficient for short-haul communications, but when using the links for long-haul purposes such as distance of 100 km, many packets may be in the pipe, so more credits are needed per

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	port. Increasing the size of the on-chip crossbar is expensive. However, external memory has traditionally not been used to store credits because it would be too slow to control the external memory." Chavan '197 at col. 19, l. 66 – col. 20, l. 16. "FIG. 11 is an illustration of an INP 1100 employing crossbar credit extensions according to embodiments of the present invention. In FIG. 11, a BCM 1104 has multiple ports, each port having a number of credits 1106 for buffering frames as they pass through the BCM 1104. The credit extensions of the present invention extend the number credits available to each port on the BCM 1104 using off-chip memory 1102. As a result, on a port by port basis, more credits can be configured than are available in the crossbar memory 1104. The external DDR memory 1102 is treated as another port on the crossbar 1104, or just an extension of the crossbar. When crossbar logic in the BCM 1104 determines that more credits are needed for a particular port, the crossbar logic will make a request to the DMA block 1108, which has access to the off-chip DDR memory 1102 through the DDR controller 1110, to effectively gain access to additional credits 1114. The DMA engine 1108 treats the off-chip DDR memory 1102 as though it were on-chip memory. When more credits are needed, the off-chip DDR 1102 can be accessed through a port 1112 on the crossbar 1104 and through the DMA engine 1108 to store frames in the DDR 1102 and effectively consume additional credits. Note that this is performed using gates, because the message frames are received so fast that processors would not be able to selectively decide what frames to put in DDR 1102 and which ones to receive from DDR. Instead, the DMA engine 1108 does this work." Chavan '197 at col. 20, ll. 17–42.
[1h] determines when said read value indicates available space in said input queue for said first processing element to issue a command;	Chavan '197 teaches "determines when said read value indicates available space in said input queue for said first processing element to issue a command," including at least the following exemplary citations. For example, Chavan '197 teaches that the second processing element executes instructions that causes it to determine when said read value indicates available space in said input queue for said first processing element to issue a command, by disclosing, as non–limiting examples, determining if there are enough credits that represent the number of free buffers. The scheduler determines if the stored value indicates available space in the input queue for the first processing element to issue a command. *See, e.g.:*

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 11 Chavan ’197 at Figure 11. “Conventional BCMs utilize the concept of credits for their ports. Credits are essentially the number of free buffers at a particular port for storing frames or packets. One credit is consumed when a packet is received, and one credit is released when a packet is sent it out of the crossbar to another port (or to the APEX block). Normally the credits for each of the ports are part of the crossbar, in limited on-chip memory. The INP may also employ crossbar credit extensions wherein multiple ports may be given an extended number of credits using off-chip memory. To accomplish this, the external DDR memory is treated as another port on the crossbar. When crossbar logic in the BCM determines that more credits are needed for a particular port, the crossbar logic will make a request to the DMA block, which has access to the off-chip DDR memory through the DDR controller, to effectively gain access to additional credits stored in the DDR memory.” Chavan ’197 at col. 6, l. 54 – col. 7, l. 2. “As shown in the example of FIG. 5, in a conventional PCI system 500, a host 510 may be connected to multiple PCI devices (although only two PCIe devices 512 and 520 are shown) through a PCI bus 514. The PCI standard specifies that configuration space shall be allocated for each PCI device in the form of configuration registers that consume hardware resources. In particular, each PCI device is assigned one or more mutually-exclusive blocks of memory in the host's memory-mapped memory space 516. In the example

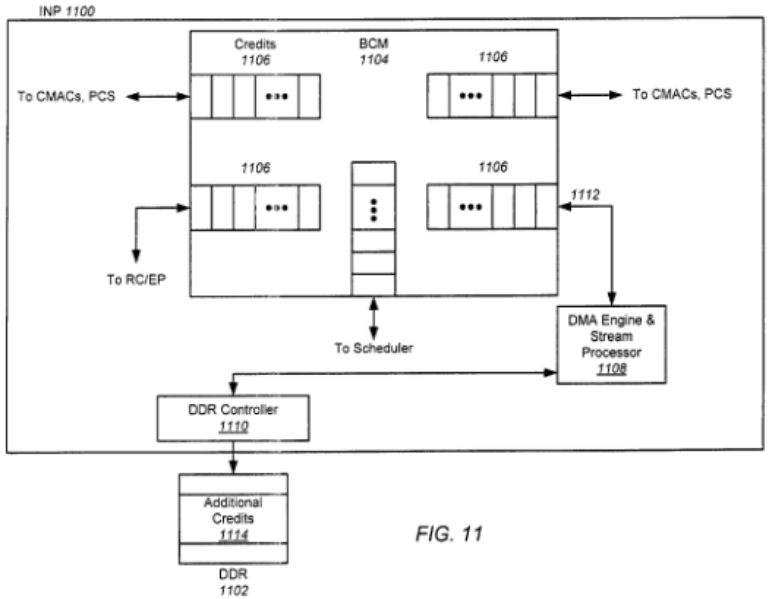
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	of FIG. 5, host 510 has been assigned memory space 518, PCI device 512 has been assigned memory space 524 (designated by the letter "m"), and PCI device 520 has been assigned memory space 526 (designated by the letter "n"). According to the PCI standard, as part of the configuration space allocated to each PCI device, each PCI device may be allocated up to eight BARs 502 in a configuration header 536, referred to as BAR0-BAR7 in FIG. 5. Each BAR contains the start address 504 of a block of memory space in memory 516 assigned to a function of the PCI device, and the length 506 and type of memory space 508 (memory or I/O) required by the function. The start address 504 of the BAR enables a memory or I/O address decoder in the PCI device to detect device accesses whose addresses fall within the block of memory space specified by the BAR. Additionally, other configuration registers 538 (including doorbells and status registers) may be allocated for each PCI device and assigned memory space such that once mapped, a host and the PCI device can communicate using that memory space. During configuration of the PCI system, the host 510 performs a configuration cycle where the host first discovers what PCI devices exist on the PCI bus 514, and then configures the configuration registers for those PCI devices. To implement the discovery process, a host's configuration software sends a discovery request out over the PCI bus 514, which reads a subset of each PCI device's configuration registers (e.g. vendor IDentification (ID) register 530 and device ID register 532) in configuration header 536 to determine the presence of the PCI device and its device type. The host 510 also accesses other configuration registers 534 in the PCI device to determine how many blocks of memory the PCI device requires. To configure a particular PCI device, the host sends a configuration request to assign memory space to the PCI device by assigning an address to the BARs of that PCI device. Note that the length field 506 and the type field 508 of each BAR are generally fixed, and only the address field 504 is programmable. When a particular PCI device has been discovered and configured, the PCI device responds back to the host, resulting in the termination of the configuration cycle for that PCI device." Chavan '197 at col. 10, l. 36 – col. 11, l. 17. "Conventional BCMs utilize the concept of credits for their ports. Credits are essentially the number of free buffers at a particular port. For example, if a port on a BCM has eight credits, eight frames of data could be received into that port and stored in the buffers without having to send an acknowledgement. One credit is consumed when a packet is received, and one credit is released when a packet is sent it out of the crossbar through that port. Normally the credits for each of the ports are part of the crossbar, in on-chip memory. However, on-chip memory is expensive, and thus traditionally the number of credits allocated to the ports on a BCM has been limited. This is sufficient for short-haul communications, but when using the links for long-haul purposes such as distance of 100 km, many packets may be in the pipe, so more credits are needed per

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	port. Increasing the size of the on-chip crossbar is expensive. However, external memory has traditionally not been used to store credits because it would be too slow to control the external memory." Chavan '197 at col. 19, l. 66 – col. 20, l. 16. "FIG. 11 is an illustration of an INP 1100 employing crossbar credit extensions according to embodiments of the present invention. In FIG. 11, a BCM 1104 has multiple ports, each port having a number of credits 1106 for buffering frames as they pass through the BCM 1104. The credit extensions of the present invention extend the number credits available to each port on the BCM 1104 using off-chip memory 1102. As a result, on a port by port basis, more credits can be configured than are available in the crossbar memory 1104. The external DDR memory 1102 is treated as another port on the crossbar 1104, or just an extension of the crossbar. When crossbar logic in the BCM 1104 determines that more credits are needed for a particular port, the crossbar logic will make a request to the DMA block 1108, which has access to the off-chip DDR memory 1102 through the DDR controller 1110, to effectively gain access to additional credits 1114. The DMA engine 1108 treats the off-chip DDR memory 1102 as though it were on-chip memory. When more credits are needed, the off-chip DDR 1102 can be accessed through a port 1112 on the crossbar 1104 and through the DMA engine 1108 to store frames in the DDR 1102 and effectively consume additional credits. Note that this is performed using gates, because the message frames are received so fast that processors would not be able to selectively decide what frames to put in DDR 1102 and which ones to receive from DDR. Instead, the DMA engine 1108 does this work." Chavan '197 at col. 20, ll. 17–42.
[1i] notifies said first processing element to issue said command to said input queue; and	Chavan '197 teaches "notifies said first processing element to issue said command to said input queue," including at least the following exemplary citations. For example, Chavan '197 teaches that the second processing element executes instructions that causes it to notify said first processing element to issue said command to said input queue, by disclosing, as non–limiting examples, notifying that a credit can be used. The scheduler notifies the first processing element to issue a command to the input queue. *See, e.g.:*

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 11 Chavan ’197 at Figure 11. “Conventional BCMs utilize the concept of credits for their ports. Credits are essentially the number of free buffers at a particular port for storing frames or packets. One credit is consumed when a packet is received, and one credit is released when a packet is sent it out of the crossbar to another port (or to the APEX block). Normally the credits for each of the ports are part of the crossbar, in limited on-chip memory. The INP may also employ crossbar credit extensions wherein multiple ports may be given an extended number of credits using off-chip memory. To accomplish this, the external DDR memory is treated as another port on the crossbar. When crossbar logic in the BCM determines that more credits are needed for a particular port, the crossbar logic will make a request to the DMA block, which has access to the off-chip DDR memory through the DDR controller, to effectively gain access to additional credits stored in the DDR memory.” Chavan ’197 at col. 6, l. 54 – col. 7, l. 2. “As shown in the example of FIG. 5, in a conventional PCI system 500, a host 510 may be connected to multiple PCI devices (although only two PCIe devices 512 and 520 are shown) through a PCI bus 514. The PCI standard specifies that configuration space shall be allocated for each PCI device in the form of configuration registers that consume hardware resources. In particular, each PCI device is assigned one or more mutually-exclusive blocks of memory in the host's memory-mapped memory space 516. In the example

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	of FIG. 5, host 510 has been assigned memory space 518, PCI device 512 has been assigned memory space 524 (designated by the letter "m"), and PCI device 520 has been assigned memory space 526 (designated by the letter "n"). According to the PCI standard, as part of the configuration space allocated to each PCI device, each PCI device may be allocated up to eight BARs 502 in a configuration header 536, referred to as BAR0-BAR7 in FIG. 5. Each BAR contains the start address 504 of a block of memory space in memory 516 assigned to a function of the PCI device, and the length 506 and type of memory space 508 (memory or I/O) required by the function. The start address 504 of the BAR enables a memory or I/O address decoder in the PCI device to detect device accesses whose addresses fall within the block of memory space specified by the BAR. Additionally, other configuration registers 538 (including doorbells and status registers) may be allocated for each PCI device and assigned memory space such that once mapped, a host and the PCI device can communicate using that memory space. During configuration of the PCI system, the host 510 performs a configuration cycle where the host first discovers what PCI devices exist on the PCI bus 514, and then configures the configuration registers for those PCI devices. To implement the discovery process, a host's configuration software sends a discovery request out over the PCI bus 514, which reads a subset of each PCI device's configuration registers (e.g. vendor IDentification (ID) register 530 and device ID register 532) in configuration header 536 to determine the presence of the PCI device and its device type. The host 510 also accesses other configuration registers 534 in the PCI device to determine how many blocks of memory the PCI device requires. To configure a particular PCI device, the host sends a configuration request to assign memory space to the PCI device by assigning an address to the BARs of that PCI device. Note that the length field 506 and the type field 508 of each BAR are generally fixed, and only the address field 504 is programmable. When a particular PCI device has been discovered and configured, the PCI device responds back to the host, resulting in the termination of the configuration cycle for that PCI device." Chavan '197 at col. 10, l. 36 – col. 11, l. 17. "Conventional BCMs utilize the concept of credits for their ports. Credits are essentially the number of free buffers at a particular port. For example, if a port on a BCM has eight credits, eight frames of data could be received into that port and stored in the buffers without having to send an acknowledgement. One credit is consumed when a packet is received, and one credit is released when a packet is sent it out of the crossbar through that port. Normally the credits for each of the ports are part of the crossbar, in on-chip memory. However, on-chip memory is expensive, and thus traditionally the number of credits allocated to the ports on a BCM has been limited. This is sufficient for short-haul communications, but when using the links for long-haul purposes such as distance of 100 km, many packets may be in the pipe, so more credits are needed per

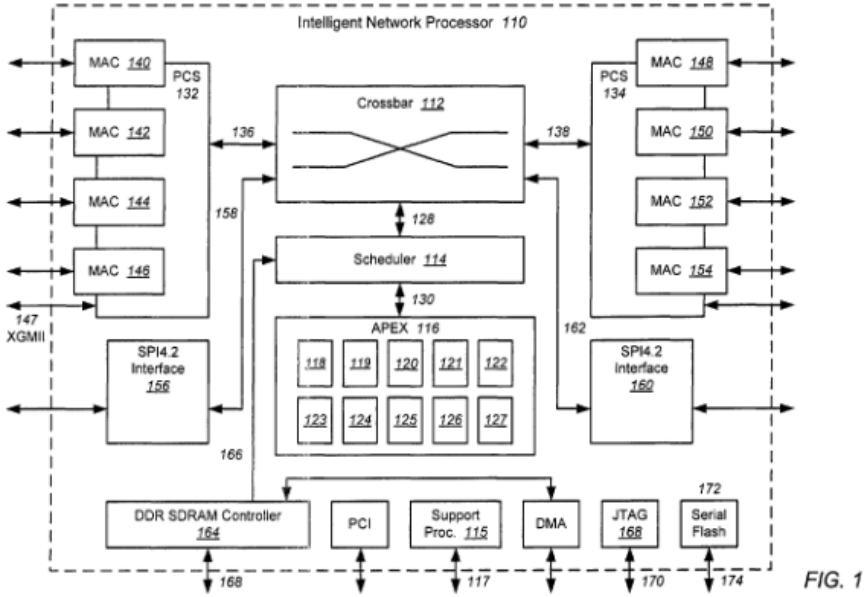
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	port. Increasing the size of the on-chip crossbar is expensive. However, external memory has traditionally not been used to store credits because it would be too slow to control the external memory." Chavan '197 at col. 19, l. 66 – col. 20, l. 16. "FIG. 11 is an illustration of an INP 1100 employing crossbar credit extensions according to embodiments of the present invention. In FIG. 11, a BCM 1104 has multiple ports, each port having a number of credits 1106 for buffering frames as they pass through the BCM 1104. The credit extensions of the present invention extend the number credits available to each port on the BCM 1104 using off-chip memory 1102. As a result, on a port by port basis, more credits can be configured than are available in the crossbar memory 1104. The external DDR memory 1102 is treated as another port on the crossbar 1104, or just an extension of the crossbar. When crossbar logic in the BCM 1104 determines that more credits are needed for a particular port, the crossbar logic will make a request to the DMA block 1108, which has access to the off-chip DDR memory 1102 through the DDR controller 1110, to effectively gain access to additional credits 1114. The DMA engine 1108 treats the off-chip DDR memory 1102 as though it were on-chip memory. When more credits are needed, the off-chip DDR 1102 can be accessed through a port 1112 on the crossbar 1104 and through the DMA engine 1108 to store frames in the DDR 1102 and effectively consume additional credits. Note that this is performed using gates, because the message frames are received so fast that processors would not be able to selectively decide what frames to put in DDR 1102 and which ones to receive from DDR. Instead, the DMA engine 1108 does this work." Chavan '197 at col. 20, ll. 17–42.
[1j] wherein said first processing element receives notification from said at least one second processing element regarding issuing the command, issues said command to said input queue, and receives a response corresponding to said command from said output queue.	Chavan '197 teaches "wherein said first processing element receives notification from said at least one second processing element regarding issuing the command, issues said command to said input queue, and receives a response corresponding to said command from said output queue," including at least the following exemplary citations. For example, Chavan '197 teaches wherein said first processing element receives notification from said at least one second processing element regarding issuing the command, issues said command to said input queue, and receives a response corresponding to said command from said output queue, by disclosing, as non–limiting examples, notifying that a credit can be used, writing into the buffer, and releasing the credit when the frame is processed. The intelligent network processor receives the notification, issues the command, and receives a corresponding response from the output queue. *See, e.g.:*

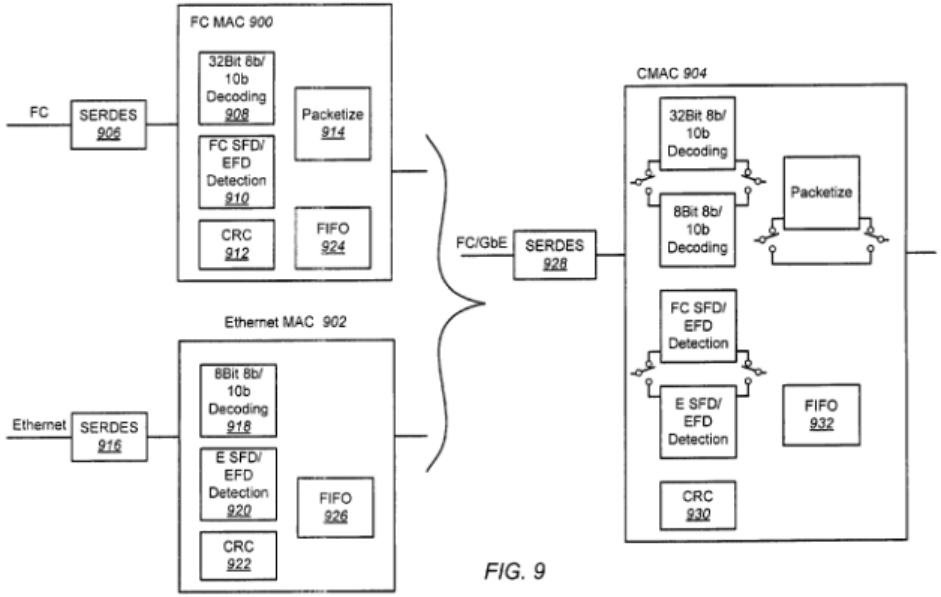
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 11 Chavan ’197 at Figure 11. “INPs are used in computer network systems for providing HBA, router, and switch capabilities for particular protocol networks connected to a host through a host interface. An example of an INP 110 is shown in FIG. 1, which may be formed on a single silicon integrated circuit die packaged inside a ceramic, metal, or plastic housing to form a unitary component. INP 110 includes a shared memory crossbar module 112 which is controlled by a flow identification/scheduler module 114 through a bus 128. The flow identification/scheduler module 114 operates under the control of the Application Processor (APEX) module 116 acting through the bus 130, and maps the passages of crossbar locations inside the shared memory crossbar module 112. The APEX 116 in the example of FIG. 1 contains ten core processors for executing firmware, denoted as processors 118-127. The shared memory crossbar module 112 is connected to Protocol Coordination Stage (PCS) modules 132 and 134 through buses 136 and 138, respectively. PCS module 132 is connected to and controls the operation of Media Access Controller (MAC) modules 140, 142, 144, and 146. The MAC modules 140, 142, 144, and 146 are identical hardware interfaces capable of each communicating using the FC protocol at 1 Gigabits per second (Gbps) or 2 Gbps speeds. The speed and protocol settings for the for the MAC modules 140, 142, 144, and 146 are controlled by the PCS 132, which operates in response to control signals from a bus (not shown)

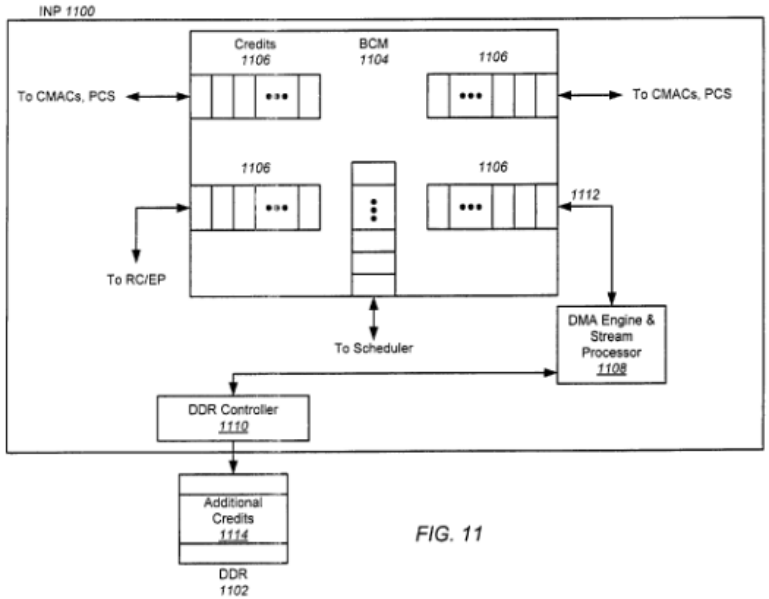
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	connecting the PCS 132 to the APEX 116. A parallel 10 Gbit (X Gbit) Medium-Independent Interface (XGMII) port 147 may be provided from the PCS 132, as determined by control signals originating from the APEX 116. The 10 Gb XGMII interface is not coupled to any of the four MAC modules." Chavan '197 at col. 1, ll. 23–53. "A Root Complex and EndPoint (RC/EP) module is connected to the BCM, and is connected to a SERDES for use with PCIe bus signals. A Direct Memory Access (DMA) engine and stream processor is connected to the BCM, and is also connected to a DDR SDRAM controller. The function of the controller is to control the sending and receiving of data to and from an external DDR memory module. A data pathway from the DMA engine and stream processor to the BCM allows DMA memory operations to occur directly with the controller and its attached DDR memory module, without data flow through the APEX, for data being exchanged with other sources. The controller is also connected to the scheduler, and creates queues in the external DDR memory for storing data messages, and also writes to and reads from such queues." Chavan '197 at col. 3, ll. 48–62. "The DMA engine and stream processor 386 is connected to the BCM 312 through bus 388, and is connected to the DDR SDRAM controller 390 through bus 392. The function of the controller 390 is to control the sending and receiving of data to and from an external DDR memory module (not shown) connected to the bus 394 of the controller 390. The controller 390 includes capabilities for providing RMW, statistics collection, XOR masking for data being transferred to or from the external DDR memory module. The data pathway through the bus 392 to the DMA engine and stream processor 386 and through the bus 388 to the BCM 312 allows DMA memory operations to occur directly with the controller 390 and its attached DDR memory module, without data flow through the APEX 316, for data being exchanged with other sources including the PCIe capabilities accessed through the bus 358 and RC/EP module 356, and the FC and GbE capabilities accessed through the buses 336 and 338 with the PCS modules 332 and 334. The ability to allow direct access for DMA operations through the BCM 312 with the DDR SDRAM controller 390 saves time that would be otherwise consumed by data being transferred into and out of the APEX 316. The controller 390 is also connected to the scheduler 314 through the bus 396. The controller 390 creates queues in the external DDR memory for storing data messages, and writes to and reads from such queues. The use of such queues, and the direct memory access to the crossbar BCM 312 (through the buses 388 and 392, controller 390, and DMA engine and stream processor 386) effectively increases the size of the crossbar BCM 312, and allows more switching pathways to be created than if the BCM 312 were used without the external DDR memory attached to connection 394." Chavan '197 at col. 8, l. 66 – col. 9, l. 29.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	"As shown in the example of FIG. 5, in a conventional PCI system 500, a host 510 may be connected to multiple PCI devices (although only two PCIe devices 512 and 520 are shown) through a PCI bus 514. The PCI standard specifies that configuration space shall be allocated for each PCI device in the form of configuration registers that consume hardware resources. In particular, each PCI device is assigned one or more mutually-exclusive blocks of memory in the host's memory-mapped memory space 516. In the example of FIG. 5, host 510 has been assigned memory space 518, PCI device 512 has been assigned memory space 524 (designated by the letter "m"), and PCI device 520 has been assigned memory space 526 (designated by the letter "n"). According to the PCI standard, as part of the configuration space allocated to each PCI device, each PCI device may be allocated up to eight BARs 502 in a configuration header 536, referred to as BAR0-BAR7 in FIG. 5. Each BAR contains the start address 504 of a block of memory space in memory 516 assigned to a function of the PCI device, and the length 506 and type of memory space 508 (memory or I/O) required by the function. The start address 504 of the BAR enables a memory or I/O address decoder in the PCI device to detect device accesses whose addresses fall within the block of memory space specified by the BAR. Additionally, other configuration registers 538 (including doorbells and status registers) may be allocated for each PCI device and assigned memory space such that once mapped, a host and the PCI device can communicate using that memory space. During configuration of the PCI system, the host 510 performs a configuration cycle where the host first discovers what PCI devices exist on the PCI bus 514, and then configures the configuration registers for those PCI devices. To implement the discovery process, a host's configuration software sends a discovery request out over the PCI bus 514, which reads a subset of each PCI device's configuration registers (e.g. vendor IDentification (ID) register 530 and device ID register 532) in configuration header 536 to determine the presence of the PCI device and its device type. The host 510 also accesses other configuration registers 534 in the PCI device to determine how many blocks of memory the PCI device requires. To configure a particular PCI device, the host sends a configuration request to assign memory space to the PCI device by assigning an address to the BARs of that PCI device. Note that the length field 506 and the type field 508 of each BAR are generally fixed, and only the address field 504 is programmable. When a particular PCI device has been discovered and configured, the PCI device responds back to the host, resulting in the termination of the configuration cycle for that PCI device." Chavan '197 at col. 10, l. 36 – col. 11, l. 17. "Conventional BCMs utilize the concept of credits for their ports. Credits are essentially the number of free buffers at a particular port. For example, if a port on a BCM has eight credits, eight frames of data could be received into that port and stored in the buffers without having to send an acknowledgement. One credit is

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	consumed when a packet is received, and one credit is released when a packet is sent it out of the crossbar through that port. Normally the credits for each of the ports are part of the crossbar, in on-chip memory. However, on-chip memory is expensive, and thus traditionally the number of credits allocated to the ports on a BCM has been limited. This is sufficient for short-haul communications, but when using the links for long-haul purposes such as distance of 100 km, many packets may be in the pipe, so more credits are needed per port. Increasing the size of the on-chip crossbar is expensive. However, external memory has traditionally not been used to store credits because it would be too slow to control the external memory." Chavan '197 at col. 19, l. 66 – col. 20, l. 16. "FIG. 11 is an illustration of an INP 1100 employing crossbar credit extensions according to embodiments of the present invention. In FIG. 11, a BCM 1104 has multiple ports, each port having a number of credits 1106 for buffering frames as they pass through the BCM 1104. The credit extensions of the present invention extend the number credits available to each port on the BCM 1104 using off-chip memory 1102. As a result, on a port by port basis, more credits can be configured than are available in the crossbar memory 1104. The external DDR memory 1102 is treated as another port on the crossbar 1104, or just an extension of the crossbar. When crossbar logic in the BCM 1104 determines that more credits are needed for a particular port, the crossbar logic will make a request to the DMA block 1108, which has access to the off-chip DDR memory 1102 through the DDR controller 1110, to effectively gain access to additional credits 1114. The DMA engine 1108 treats the off-chip DDR memory 1102 as though it were on-chip memory. When more credits are needed, the off-chip DDR 1102 can be accessed through a port 1112 on the crossbar 1104 and through the DMA engine 1108 to store frames in the DDR 1102 and effectively consume additional credits. Note that this is performed using gates, because the message frames are received so fast that processors would not be able to selectively decide what frames to put in DDR 1102 and which ones to receive from DDR. Instead, the DMA engine 1108 does this work." Chavan '197 at col. 20, ll. 17–42.
2. The system of claim 1, wherein said input queue is a first-in-first-out (FIFO) buffer.	Chavan '197 teaches "[t]he system of claim 1, wherein said input queue is a first-in-first-out (FIFO) buffer," including at least the following exemplary citations. For example, Chavan '197 teaches wherein said input queue is a first-in-first-out (FIFO) buffer, by disclosing, as non–limiting examples, FIFO buffers. *See, e.g.:*

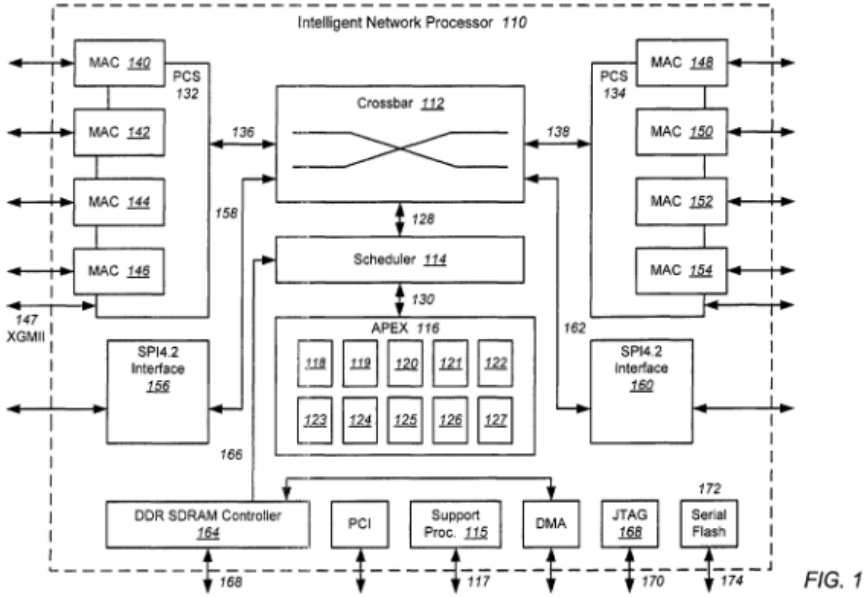
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan '197 at Figure 1.

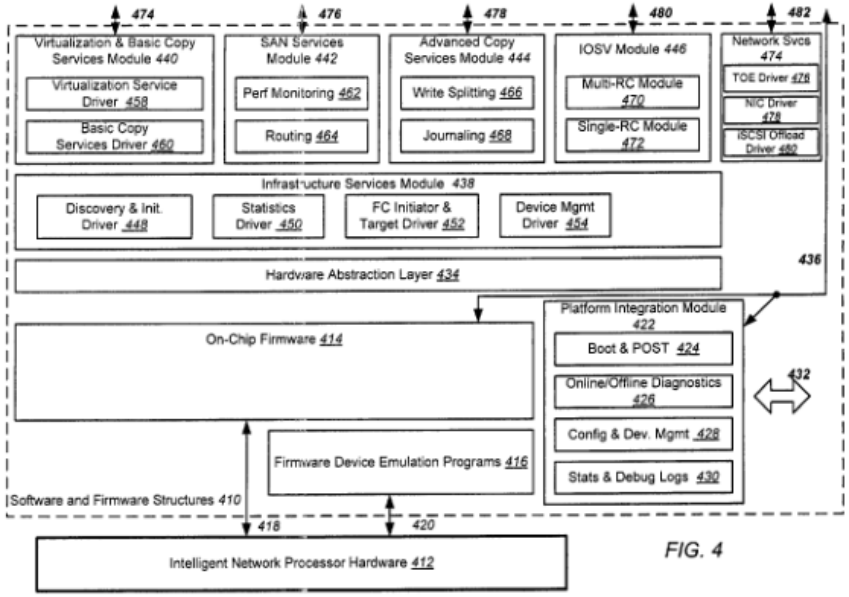
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan ’197 at Figure 9.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 11 Chavan ’197 at Figure 11. “An INP may also utilize programmable configuration logic that may replace the aperture table described above, or may work together with the aperture table. In this embodiment, the headers of PCIe transactions are evaluated by the configuration logic to determine the classification of the PCIe transaction (i.e. the type of cycle). Depending on the classification, the configuration logic may take certain actions. For example, if the configuration logic determines that a discovery or configuration request (i.e. a configuration cycle) is being presented, the discovery or configuration request is passed through the crossbar to the APEX. A free processor in the APEX then executes firmware to respond to and terminate the configuration cycle by providing information and indications back to the host indicating that virtual PCIe devices have been discovered and configured. The particular processor and firmware performs translations from the configuration space of the requestor to the virtual configuration space of the virtual PCIe device being emulated. The common processing elements of FC and GbE MACs may be integrated into an efficient CMAC. Both protocols require SERDES, Cyclic Redundancy Checking (CRC) and First-In-First-Out (FIFO) buffers for rate matching, and thus the logic for these tasks is shared in the CMAC. However, because the logic (state machine) to perform 8b10b decoding, to detect Start of Frame Delimiters (SFDs) and End of Frame

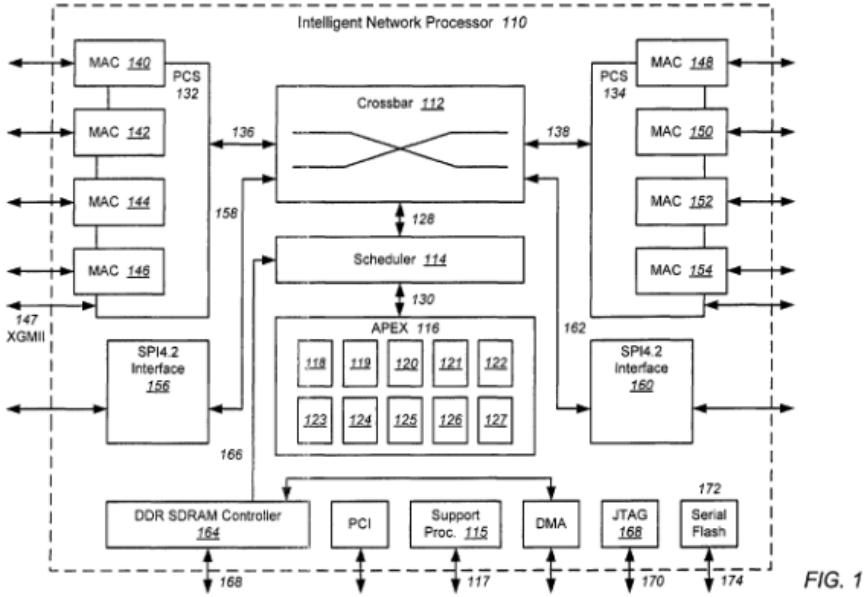
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	Delimiters (EFDs) for each protocol, and to packetize FC frames is different for the two protocols, this logic is duplicated and switched in or out as needed. To produce a FC/GbE CMAC, multiple bytes must be framed as packets. After the SFDs have been detected, logic specific to each protocol may be employed outside the CMAC and switched in or out depending on the protocol to parse the bytes in the packet. The CMACs are configurable by a processor to decode the incoming packets or frames as either FC "packets" or GbE packets." Chavan '197 at col. 5, l. 50 – col. 6, l. 16. "FIG. 9 is an illustration of how the common processing elements of MACs for FC 900 and MACs for GbE 902 can be integrated into an efficient CMAC 904 according to embodiments of the present invention. Conventionally, FC frames are received and deserialized by SERDES 906, and are then processed by a FC MAC 900. This processing typically includes 32-bit 8b10b decoding 908, detection of a SFD (a.k.a. a "k" character) and an EFD 910, CRC 912, generating packets from the FC frames 914, and buffering in a FIFO buffer 924. Similarly, GbE packets are received and deserialized by SERDES 916, and are then processed by a GbE MAC 902. This processing typically includes 8-bit 8b10b decoding 918, detection of a SFD and an EFD 920, CRC 922, and FIFO buffering 926. Because of the processing similarities, embodiments of the present invention integrate the data paths for both the FC and GbE protocols in a FC/GbE CMAC 904. Both protocols require SERDES 928, CRC 930 and FIFOs 932 for rate matching, and thus the logic for these tasks is shared in the CMAC 904. However, because the logic (state machine) to perform 8b10b decoding, to detect SFDs and EFDs for each protocol, and to packetize FC frames is different for the two protocols, this logic must be duplicated and switched in or out as needed. To produce a FC/GbE CMAC, multiple bytes must be framed as packets (even though FC uses "frame" terminology). The bytes in a packet are well-defined in the FC and GbE standards. After the SFDs have been detected, logic specific to each protocol may be employed outside the CMAC and switched in or out depending on the protocol to parse the bytes in the packet. The CMACs are configurable by a processor to decode the incoming packets or frames as either FC "packets" or GbE packets." Chavan '197 at col. 17, ll. 22–53. "52. The INP as recited in claim 48, wherein each CMAC comprises: switchable 32-bit FC 8B10b decoding logic and 8-bit GbE 8b10b decoding logic; switchable FC and GbE Start of Frame Delimiter (SFD) and End of Frame Delimiter (EFD) detection logic; common Cyclic Redundancy Check (CRC) logic; common packetizing logic; and

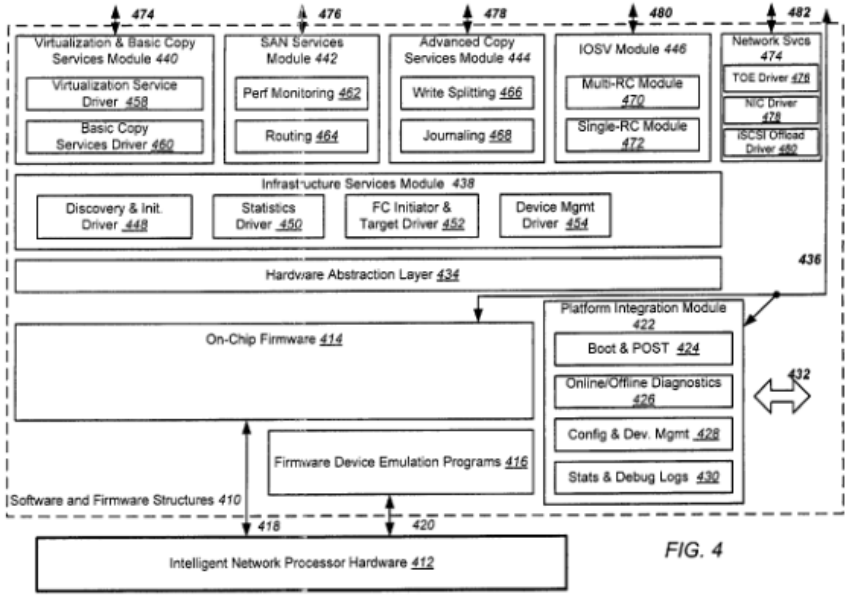
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	common First In First Out (FIFO) buffering." Chavan '197 at col. 25, ll. 18–26. "57. The method as recited in claim 53, further comprising forming each CMAC by: providing switchable 32-bit FC 8B10b decoding logic and 8-bit GbE 8b10b decoding logic; providing switchable FC and GbE Start of Frame Delimiter (SFD) and End of Frame Delimiter (EFD) detection logic; providing common Cyclic Redundancy Check (CRC) logic; providing common packetizing logic; and providing common First In First Out (FIFO) buffering." Chavan '197 at col. 26, ll. 1–10.
3. The system of claim 1, wherein said system is part of a storage area network (SAN) controller.	Chavan '197 teaches "[t]he system of claim 1, wherein said system is part of a storage area network (SAN) controller," including at least the following exemplary citations. For example, Chavan '197 teaches wherein said system is part of a storage area network (SAN) controller, by disclosing, as non–limiting examples, an intelligent network processor that supports Fibre Channel and Gigabit Ethernet. Fibre Channel is used in a SAN controller. *See, e.g.:* "An intelligent network processor is disclosed that provides a PCI express (PCIe) host bus adapter with firmware selectable hardware capabilities and firmware enabled emulation of capabilities not supported by hardware. Support for Fibre Channel (FC) and Gigabit Ethernet (GbE) protocols are provided through the same fabric ports, including multiple port trunking for both protocols. On chip protocol conversion is provided for switching and routing between FC and GbE ports. Switching using the same crossbar module is provided for both FC and GbE protocols. The crossbar module is coupled to directly access external DDR memory so that messages from FC, GbE, and PCIe interfaces may be switched directly to the DDR memory." Chavan '197 at Abstract.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan '197 at Figure 1.

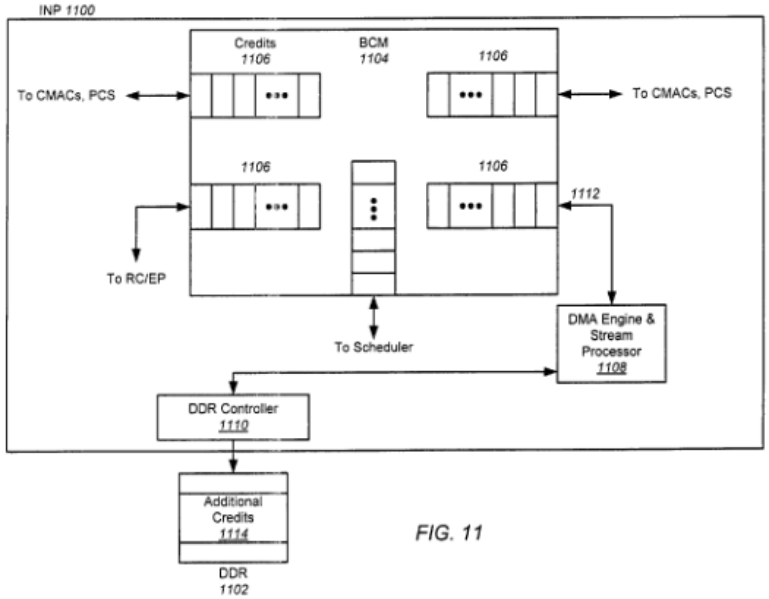
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 4 Chavan ’197 at Figure 4. “This invention relates to Intelligent Network Processors (INPs) for use in computer network systems, and more particularly to such INPs for providing Host Bus Adapter (HBA), router, and switch capabilities on Fibre Channel (FC) and Gigabit Ethernet (GbE) protocol networks connected to a host through a Peripheral Component Interconnect express (PCIe) host interface.” Chavan ’197 at col. 1, ll. 13–19. “FIG. 9 is an illustration of how the common processing elements of MACs for FC 900 and MACs for GbE 902 can be integrated into an efficient CMAC 904 according to embodiments of the present invention. Conventionally, FC frames are received and deserialized by SERDES 906, and are then processed by a FC MAC 900. This processing typically includes 32-bit 8b10b decoding 908, detection of a SFD (a.k.a. a “k” character) and an EFD 910, CRC 912, generating packets from the FC frames 914, and buffering in a FIFO buffer 924. Similarly, GbE packets are received and deserialized by SERDES 916, and are then processed by a GbE MAC 902. This processing typically includes 8-bit 8b10b decoding 918, detection of a SFD and an EFD 920, CRC 922, and FIFO buffering 926. Because of the processing similarities, embodiments of the present invention integrate the data paths for both the FC and GbE protocols in a FC/GbE CMAC 904. Both protocols require SERDES 928, CRC 930 and FIFOs 932 for rate matching, and thus the logic for these tasks is shared in the CMAC 904. However, because

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	the logic (state machine) to perform 8b10b decoding, to detect SFDs and EFDs for each protocol, and to packetize FC frames is different for the two protocols, this logic must be duplicated and switched in or out as needed. To produce a FC/GbE CMAC, multiple bytes must be framed as packets (even though FC uses "frame" terminology). The bytes in a packet are well-defined in the FC and GbE standards. After the SFDs have been detected, logic specific to each protocol may be employed outside the CMAC and switched in or out depending on the protocol to parse the bytes in the packet. The CMACs are configurable by a processor to decode the incoming packets or frames as either FC "packets" or GbE packets." Chavan '197 at col. 17, ll. 22–53.
4. The system of claim 1, wherein said system is part of a network attached storage (NAS) controller.	Chavan '197 teaches "[t]he system of claim 1, wherein said system is part of a network attached storage (NAS) controller," including at least the following exemplary citations. For example, Chavan '197 teaches wherein said system is part of a network attached storage (NAS) controller, by disclosing, as non–limiting examples, an intelligent network processor that supports Fibre Channel and Gigabit Ethernet. Gigabet Ethernet is used in a NAS controller. *See, e.g.:* "An intelligent network processor is disclosed that provides a PCI express (PCIe) host bus adapter with firmware selectable hardware capabilities and firmware enabled emulation of capabilities not supported by hardware. Support for Fibre Channel (FC) and Gigabit Ethernet (GbE) protocols are provided through the same fabric ports, including multiple port trunking for both protocols. On chip protocol conversion is provided for switching and routing between FC and GbE ports. Switching using the same crossbar module is provided for both FC and GbE protocols. The crossbar module is coupled to directly access external DDR memory so that messages from FC, GbE, and PCIe interfaces may be switched directly to the DDR memory." Chavan '197 at Abstract.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan '197 at Figure 1.

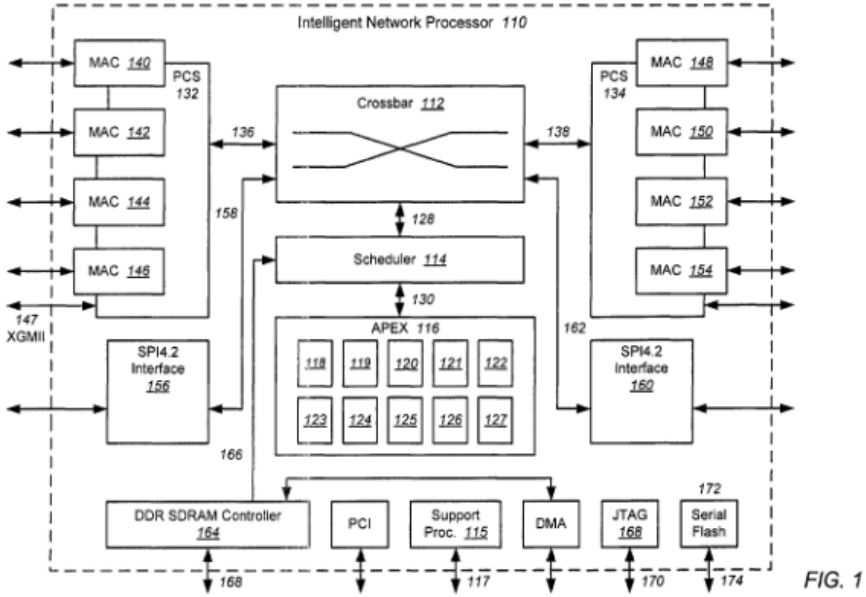
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 4 Chavan ’197 at Figure 4. “This invention relates to Intelligent Network Processors (INPs) for use in computer network systems, and more particularly to such INPs for providing Host Bus Adapter (HBA), router, and switch capabilities on Fibre Channel (FC) and Gigabit Ethernet (GbE) protocol networks connected to a host through a Peripheral Component Interconnect express (PCIe) host interface.” Chavan ’197 at col. 1, ll. 13–19. “FIG. 9 is an illustration of how the common processing elements of MACs for FC 900 and MACs for GbE 902 can be integrated into an efficient CMAC 904 according to embodiments of the present invention. Conventionally, FC frames are received and deserialized by SERDES 906, and are then processed by a FC MAC 900. This processing typically includes 32-bit 8b10b decoding 908, detection of a SFD (a.k.a. a “k” character) and an EFD 910, CRC 912, generating packets from the FC frames 914, and buffering in a FIFO buffer 924. Similarly, GbE packets are received and deserialized by SERDES 916, and are then processed by a GbE MAC 902. This processing typically includes 8-bit 8b10b decoding 918, detection of a SFD and an EFD 920, CRC 922, and FIFO buffering 926. Because of the processing similarities, embodiments of the present invention integrate the data paths for both the FC and GbE protocols in a FC/GbE CMAC 904. Both protocols require SERDES 928, CRC 930 and FIFOs 932 for rate matching, and thus the logic for these tasks is shared in the CMAC 904. However, because

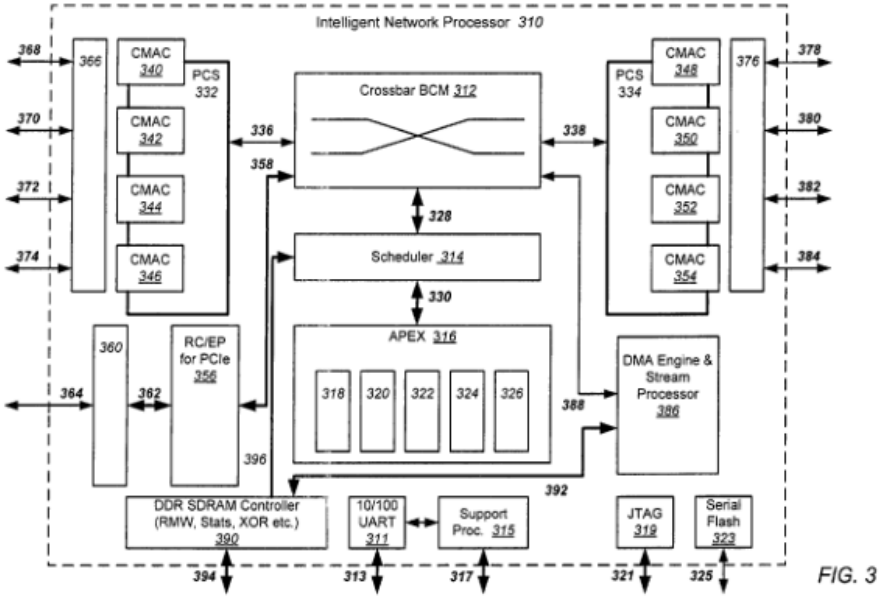
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	the logic (state machine) to perform 8b10b decoding, to detect SFDs and EFDs for each protocol, and to packetize FC frames is different for the two protocols, this logic must be duplicated and switched in or out as needed. To produce a FC/GbE CMAC, multiple bytes must be framed as packets (even though FC uses "frame" terminology). The bytes in a packet are well-defined in the FC and GbE standards. After the SFDs have been detected, logic specific to each protocol may be employed outside the CMAC and switched in or out depending on the protocol to parse the bytes in the packet. The CMACs are configurable by a processor to decode the incoming packets or frames as either FC "packets" or GbE packets." Chavan '197 at col. 17, ll. 22–53.
5. The system of claim 1, wherein said stored value is decremented by a predefined value prior to returning said stored value to said at least one second processing element.	Chavan '197 teaches "[t]he system of claim 1, wherein said stored value is decremented by a predefined value prior to returning said stored value to said at least one second processing element," including at least the following exemplary citations. For example, Chavan '197 teaches wherein said stored value is decremented by a predefined value prior to returning said stored value to said at least one second processing element by disclosing, as non–limiting examples, decrementing the credits. The credit counter decrements the stored value by a predefined value. *See, e.g.:* "Conventional BCMs utilize the concept of credits for their ports. Credits are essentially the number of free buffers at a particular port for storing frames or packets. One credit is consumed when a packet is received, and one credit is released when a packet is sent it out of the crossbar to another port (or to the APEX block). Normally the credits for each of the ports are part of the crossbar, in limited on-chip memory. The INP may also employ crossbar credit extensions wherein multiple ports may be given an extended number of credits using off-chip memory. To accomplish this, the external DDR memory is treated as another port on the crossbar. When crossbar logic in the BCM determines that more credits are needed for a particular port, the crossbar logic will make a request to the DMA block, which has access to the off-chip DDR memory through the DDR controller, to effectively gain access to additional credits stored in the DDR memory." Chavan '197 at col. 6, l. 54 – col. 7, l. 2. "Conventional BCMs utilize the concept of credits for their ports. Credits are essentially the number of free buffers at a particular port. For example, if a port on a BCM has eight credits, eight frames of data could be received into that port and stored in the buffers without having to send an acknowledgement. One credit is consumed when a packet is received, and one credit is released when a packet is sent it out of the crossbar through that port. Normally the credits for each of the ports are part of the crossbar, in on-chip memory. However, on-chip memory is expensive, and thus traditionally the number of credits allocated to the ports on

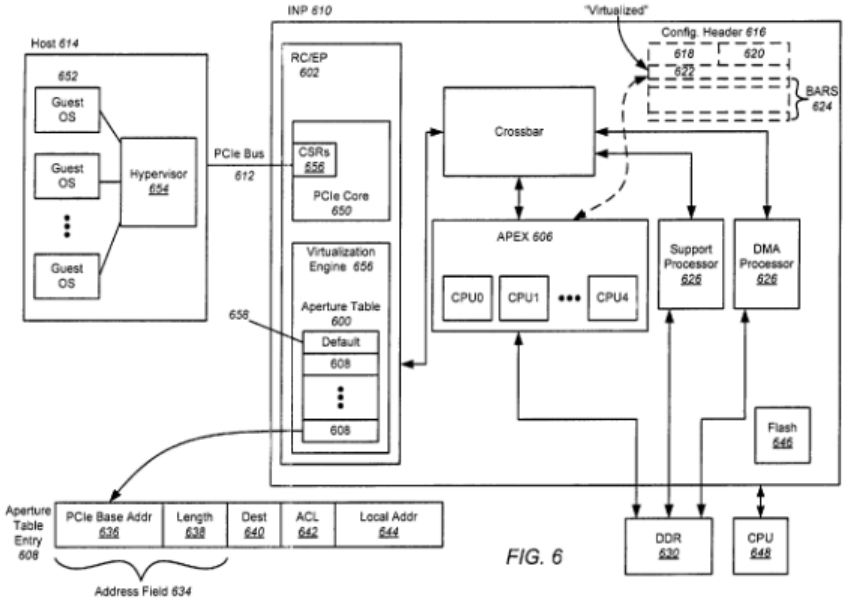
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	a BCM has been limited. This is sufficient for short-haul communications, but when using the links for long-haul purposes such as distance of 100 km, many packets may be in the pipe, so more credits are needed per port. Increasing the size of the on-chip crossbar is expensive. However, external memory has traditionally not been used to store credits because it would be too slow to control the external memory." Chavan '197 at col. 19, l. 66 – col. 20, l. 16.
6. The system of claim 1, wherein at least one third processing element executes said command and notifies said output queue to issue said response.	Chavan '197 teaches "[t]he system of claim 1, wherein at least one third processing element executes said command and notifies said output queue to issue said response," including at least the following exemplary citations. For example, Chavan '197 teaches wherein at least one third processing element executes said command and notifies said output queue to issue said response, by disclosing, as non–limiting examples, DMA Engine and Stream Processor. The DMA Engine and Stream Processor executes the command and notifies the output queue to issue a response. *See, e.g.:*  *FIG. 11* Chavan '197 at Figure 11.

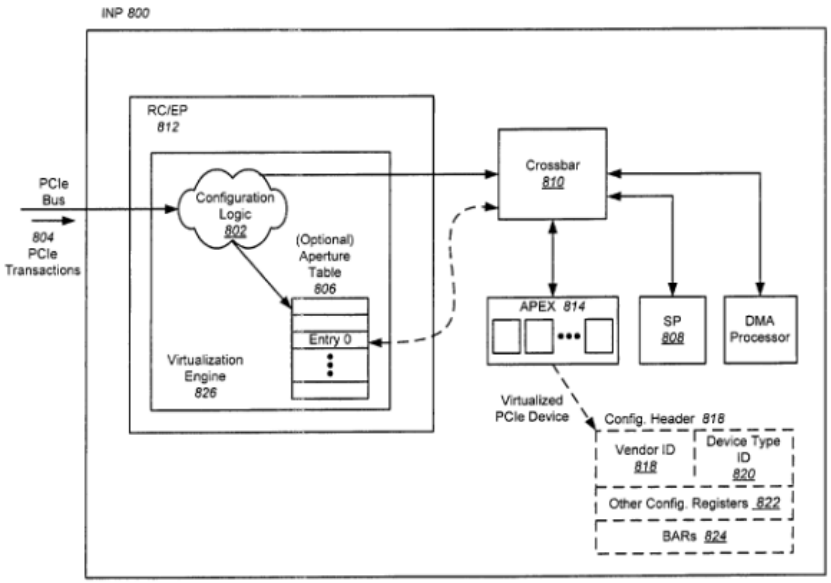
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	"INPs are used in computer network systems for providing HBA, router, and switch capabilities for particular protocol networks connected to a host through a host interface. An example of an INP 110 is shown in FIG. 1, which may be formed on a single silicon integrated circuit die packaged inside a ceramic, metal, or plastic housing to form a unitary component. INP 110 includes a shared memory crossbar module 112 which is controlled by a flow identification/scheduler module 114 through a bus 128. The flow identification/scheduler module 114 operates under the control of the Application Processor (APEX) module 116 acting through the bus 130, and maps the passages of crossbar locations inside the shared memory crossbar module 112. The APEX 116 in the example of FIG. 1 contains ten core processors for executing firmware, denoted as processors 118-127. The shared memory crossbar module 112 is connected to Protocol Coordination Stage (PCS) modules 132 and 134 through buses 136 and 138, respectively. PCS module 132 is connected to and controls the operation of Media Access Controller (MAC) modules 140, 142, 144, and 146. The MAC modules 140, 142, 144, and 146 are identical hardware interfaces capable of each communicating using the FC protocol at 1 Gigabits per second (Gbps) or 2 Gbps speeds. The speed and protocol settings for the for the MAC modules 140, 142, 144, and 146 are controlled by the PCS 132, which operates in response to control signals from a bus (not shown) connecting the PCS 132 to the APEX 116. A parallel 10 Gbit (X Gbit) Medium-Independent Interface (XGMII) port 147 may be provided from the PCS 132, as determined by control signals originating from the APEX 116. The 10 Gb XGMII interface is not coupled to any of the four MAC modules." Chavan '197 at col. 1, ll. 23–53. "A Root Complex and EndPoint (RC/EP) module is connected to the BCM, and is connected to a SERDES for use with PCIe bus signals. A Direct Memory Access (DMA) engine and stream processor is connected to the BCM, and is also connected to a DDR SDRAM controller. The function of the controller is to control the sending and receiving of data to and from an external DDR memory module. A data pathway from the DMA engine and stream processor to the BCM allows DMA memory operations to occur directly with the controller and its attached DDR memory module, without data flow through the APEX, for data being exchanged with other sources. The controller is also connected to the scheduler, and creates queues in the external DDR memory for storing data messages, and also writes to and reads from such queues." Chavan '197 at col. 3, ll. 48–62. "The DMA engine and stream processor 386 is connected to the BCM 312 through bus 388, and is connected to the DDR SDRAM controller 390 through bus 392. The function of the controller 390 is to control the sending and receiving of data to and from an external DDR memory module (not shown) connected to the bus 394 of the controller 390. The controller 390 includes capabilities for providing RMW, statistics collection,

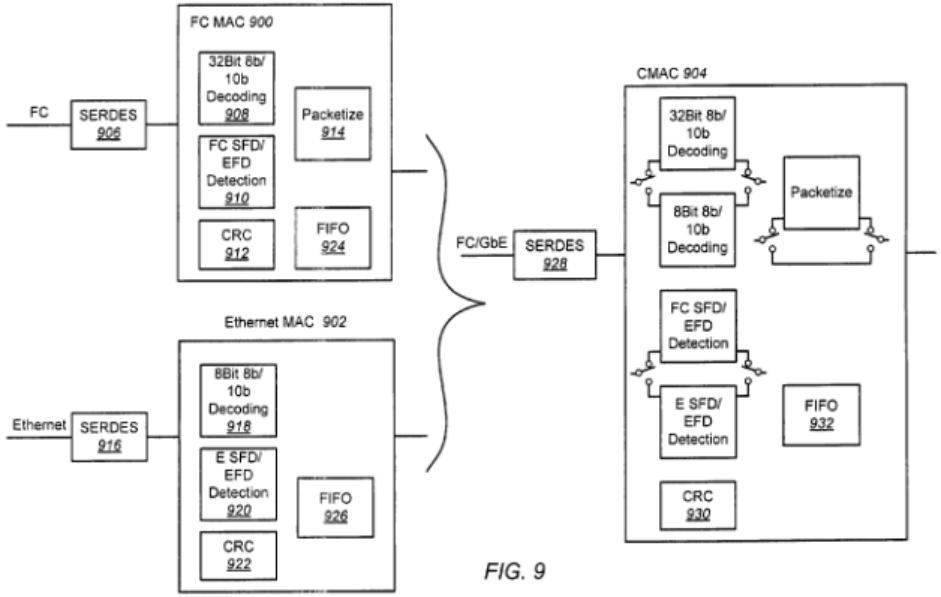
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	XOR masking for data being transferred to or from the external DDR memory module. The data pathway through the bus 392 to the DMA engine and stream processor 386 and through the bus 388 to the BCM 312 allows DMA memory operations to occur directly with the controller 390 and its attached DDR memory module, without data flow through the APEX 316, for data being exchanged with other sources including the PCIe capabilities accessed through the bus 358 and RC/EP module 356, and the FC and GbE capabilities accessed through the buses 336 and 338 with the PCS modules 332 and 334. The ability to allow direct access for DMA operations through the BCM 312 with the DDR SDRAM controller 390 saves time that would be otherwise consumed by data being transferred into and out of the APEX 316. The controller 390 is also connected to the scheduler 314 through the bus 396. The controller 390 creates queues in the external DDR memory for storing data messages, and writes to and reads from such queues. The use of such queues, and the direct memory access to the crossbar BCM 312 (through the buses 388 and 392, controller 390, and DMA engine and stream processor 386) effectively increases the size of the crossbar BCM 312, and allows more switching pathways to be created than if the BCM 312 were used without the external DDR memory attached to connection 394." Chavan '197 at col. 8, l. 66 – col. 9, l. 29.
[7pre] A system for dividing and synchronizing a processing task across a plurality of processing elements comprising:	Chavan '197 teaches "[a] system for dividing and synchronizing a processing task across a plurality of processing elements," including at least the following exemplary citations. *See* above at row [1pre].
[7a] an input queue implemented in hardware;	Chavan '197 teaches "an input queue implemented in hardware," including at least the following exemplary citations. *See* above at row [1a].
[7b] an output queue implemented in hardware;	Chavan '197 teaches "an output queue implemented in hardware," including at least the following exemplary citations. *See* above at row [1b].
[7c] a first processing element having access to said input queue and said output queue;	Chavan '197 teaches "a first processing element having access to said input queue and said output queue," including at least the following exemplary citations. *See* above at row [1c].

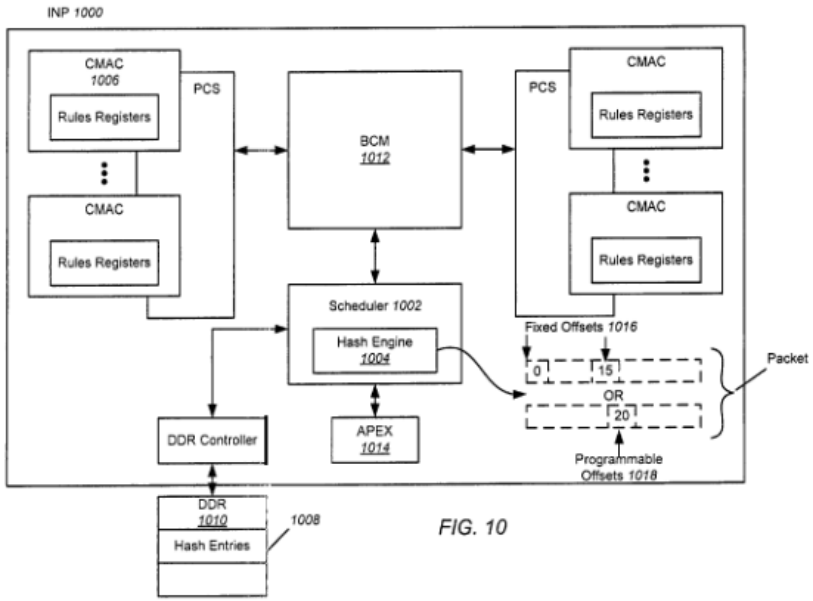
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
[7d] at least one second processing element in communication with said input queue;	Chavan '197 teaches "at least one second processing element in communication with said input queue," including at least the following exemplary citations. For example, Chavan '197 teaches at least one second processing element in communication with said input queue, by disclosing, as non–limiting examples, a scheduler that communicates with the processors within the APEX and the crossbar. The scheduler, the second processing element is in communication with the input queue. *See, e.g.:*  Chavan '197 at Figure 1.

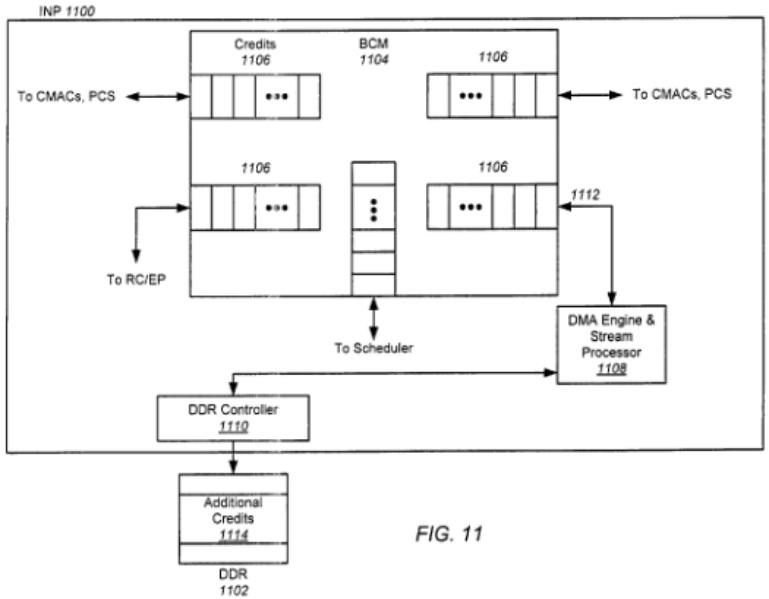
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan ’197 at Figure 3.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan ’197 at Figure 6.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 8 Chavan ’197 at Figure 8.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan ’197 at Figure 9.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan ’197 at Figure 10.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 11 Chavan ’197 at Figure 11. “INPs are used in computer network systems for providing HBA, router, and switch capabilities for particular protocol networks connected to a host through a host interface. An example of an INP 110 is shown in FIG. 1, which may be formed on a single silicon integrated circuit die packaged inside a ceramic, metal, or plastic housing to form a unitary component. INP 110 includes a shared memory crossbar module 112 which is controlled by a flow identification/scheduler module 114 through a bus 128. The flow identification/scheduler module 114 operates under the control of the Application Processor (APEX) module 116 acting through the bus 130, and maps the passages of crossbar locations inside the shared memory crossbar module 112. The APEX 116 in the example of FIG. 1 contains ten core processors for executing firmware, denoted as processors 118-127. The shared memory crossbar module 112 is connected to Protocol Coordination Stage (PCS) modules 132 and 134 through buses 136 and 138, respectively. PCS module 132 is connected to and controls the operation of Media Access Controller (MAC) modules 140, 142, 144, and 146. The MAC modules 140, 142, 144, and 146 are identical hardware interfaces capable of each communicating using the FC protocol at 1 Gigabits per second (Gbps) or 2 Gbps speeds. The speed and protocol settings for the for the MAC modules 140, 142, 144, and 146 are controlled by the PCS 132, which operates in response to control signals from a bus (not shown)

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	connecting the PCS 132 to the APEX 116. A parallel 10 Gbit (X Gbit) Medium-Independent Interface (XGMII) port 147 may be provided from the PCS 132, as determined by control signals originating from the APEX 116. The 10 Gb XGMII interface is not coupled to any of the four MAC modules." Chavan '197 at col. 1, ll. 23–53. "A Double Data Rate (DDR) Synchronous Dynamic Random Access Memory (SDRAM) controller 164 is connected to the scheduler 114 through bus 166. The function of the controller 164 is to control the sending and receiving of data to and from an external DDR memory module (not shown) connected to an external bus 168 of the controller. A support processor 115 handles certain processing tasks not otherwise covered by the APEX 116, and has an external connection 117. A Joint Test Action Group (JTAG) module 168 has an external connection 170, and is used for boundary scan testing of the hardware contained within INP 110. Serial flash memory 172 has an external connection 174, and is a non-volatile reprogrammable memory which may be loaded through the connection to store firmware for operation of APEX 116 and support processor 115." Chavan '197 at col. 2, ll. 14–28. "The present invention is directed to INPs for providing HBA, router, and switch capabilities on FC and GbE protocol networks connected to a single host or multiple hosts (as in the case of multi-root) through a PCIe host interface. The INP includes a Bi-directional Crossbar Module (BCM) which is controlled by a flow identification and scheduler module. The scheduler operates under the control of the APEX, and maps the passages of crossbar locations inside the BCM. The APEX contains multiple core processors for executing firmware. The BCM is connected to PCS modules, which are connected to and control the operation of Combined Media Access Controller (CMAC) modules capable of each communicating using multiple protocols. The speed and protocol settings for the CMAC modules are controlled by the PCS. The PCS also has trunking modes in which it coordinates the operation of the CMAC modules together as a single trunk. A combined SERDES is coupled to each of the CMACs and selectively provides, under control of signals coupled from the APEX, electronic signals in one of multiple protocols (e.g. the FC and GbE protocols), or trunking protocols." Chavan '197 at col. 3, ll. 28–47. "A Root Complex and EndPoint (RC/EP) module is connected to the BCM, and is connected to a SERDES for use with PCIe bus signals. A Direct Memory Access (DMA) engine and stream processor is connected to the BCM, and is also connected to a DDR SDRAM controller. The function of the controller is to control the sending and receiving of data to and from an external DDR memory module. A data pathway from the DMA engine and stream processor to the BCM allows DMA memory operations to occur directly with the

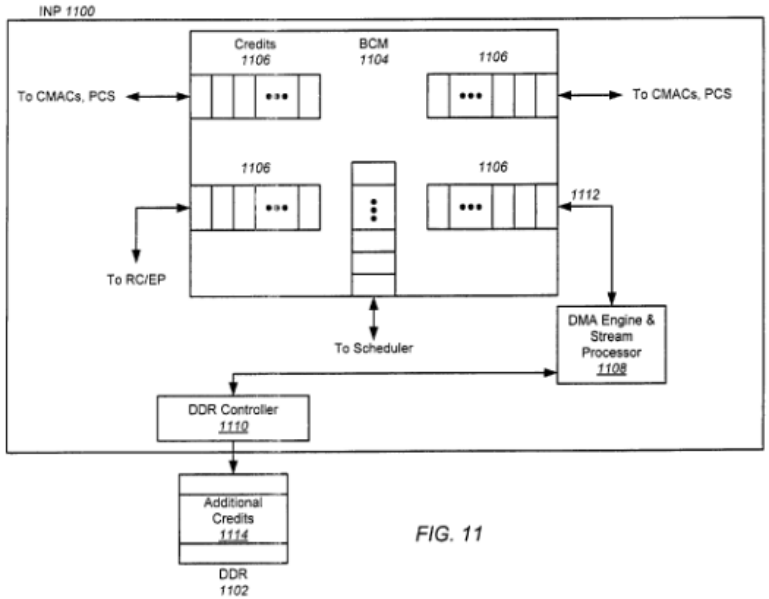
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	controller and its attached DDR memory module, without data flow through the APEX, for data being exchanged with other sources. The controller is also connected to the scheduler, and creates queues in the external DDR memory for storing data messages, and also writes to and reads from such queues." Chavan '197 at col. 3, ll. 48–62. "An INP may also utilize programmable configuration logic that may replace the aperture table described above, or may work together with the aperture table. In this embodiment, the headers of PCIe transactions are evaluated by the configuration logic to determine the classification of the PCIe transaction (i.e. the type of cycle). Depending on the classification, the configuration logic may take certain actions. For example, if the configuration logic determines that a discovery or configuration request (i.e. a configuration cycle) is being presented, the discovery or configuration request is passed through the crossbar to the APEX. A free processor in the APEX then executes firmware to respond to and terminate the configuration cycle by providing information and indications back to the host indicating that virtual PCIe devices have been discovered and configured. The particular processor and firmware performs translations from the configuration space of the requestor to the virtual configuration space of the virtual PCIe device being emulated. The common processing elements of FC and GbE MACs may be integrated into an efficient CMAC. Both protocols require SERDES, Cyclic Redundancy Checking (CRC) and First-In-First-Out (FIFO) buffers for rate matching, and thus the logic for these tasks is shared in the CMAC. However, because the logic (state machine) to perform 8b10b decoding, to detect Start of Frame Delimiters (SFDs) and End of Frame Delimiters (EFDs) for each protocol, and to packetize FC frames is different for the two protocols, this logic is duplicated and switched in or out as needed. To produce a FC/GbE CMAC, multiple bytes must be framed as packets. After the SFDs have been detected, logic specific to each protocol may be employed outside the CMAC and switched in or out depending on the protocol to parse the bytes in the packet. The CMACs are configurable by a processor to decode the incoming packets or frames as either FC "packets" or GbE packets." Chavan '197 at col. 5, l. 50 – col. 6, l. 16. "FIG. 3 is an illustration of an INP 310 implemented according to embodiments of the present invention. INP 310 may be composed of a single silicon integrated circuit die, which may be packaged inside a ceramic, metal, or plastic housing to form a unitary component. NP 310 includes a BCM 312 which is controlled by a flow identification and scheduler module 314 through a bus 328. The BCM 312 may be a multi-ported shared Random Access Memory (RAM) having memory locations mapped as passages, and operating at 224 Gbps. The scheduler 314 operates under the control of the APEX 316 acting through the bus 330, and maps the passages of crossbar locations inside the BCM 312. The scheduler 314 pre-processes received frames,

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	packets, and PCIe requests before they are acted upon by APEX 316. The scheduler 314 also parses received FC frames into correct sequence, and parses received GbE packets (in Internet Protocol (IP) format) into a Transport Control Protocol (TCP) flow. The APEX 316 preferably contains five core processors for executing firmware, denoted as processors 318, 320, 322, 324, and 326. The processors 318, 322, 324, and 326 may be core model ARC750 as provided by ARC, Inc., which preferably each have the capability of executing 32 bit Reduced Instruction Set Computer (RISC) firmware with a clock speed of 500 MHz, and may each include 32 KB of dedicated instruction memory (not shown)." Chavan '197 at col. 7, l. 49 – col. 8, l. 6. "The DMA engine and stream processor 386 is connected to the BCM 312 through bus 388, and is connected to the DDR SDRAM controller 390 through bus 392. The function of the controller 390 is to control the sending and receiving of data to and from an external DDR memory module (not shown) connected to the bus 394 of the controller 390. The controller 390 includes capabilities for providing RMW, statistics collection, XOR masking for data being transferred to or from the external DDR memory module. The data pathway through the bus 392 to the DMA engine and stream processor 386 and through the bus 388 to the BCM 312 allows DMA memory operations to occur directly with the controller 390 and its attached DDR memory module, without data flow through the APEX 316, for data being exchanged with other sources including the PCIe capabilities accessed through the bus 358 and RC/EP module 356, and the FC and GbE capabilities accessed through the buses 336 and 338 with the PCS modules 332 and 334. The ability to allow direct access for DMA operations through the BCM 312 with the DDR SDRAM controller 390 saves time that would be otherwise consumed by data being transferred into and out of the APEX 316. The controller 390 is also connected to the scheduler 314 through the bus 396. The controller 390 creates queues in the external DDR memory for storing data messages, and writes to and reads from such queues. The use of such queues, and the direct memory access to the crossbar BCM 312 (through the buses 388 and 392, controller 390, and DMA engine and stream processor 386) effectively increases the size of the crossbar BCM 312, and allows more switching pathways to be created than if the BCM 312 were used without the external DDR memory attached to connection 394." Chavan '197 at col. 8, l. 66 – col. 9, l. 29. "FIG. 6 is an illustration of an exemplary system employing an INP 610 according to embodiments of the present invention. Note that INP 610 is a simplified representation of the INP of FIG. 3, emphasizing the PCIe interface. FIG. 6 illustrates a so-called "single root" system configuration comprised of a single root complex (the host 614) and an INP 610 representing one of perhaps several endpoints. In the example of FIG. 6, the host is logically partitioned into multiple virtual servers, each running a separate "guest" OS 652. The

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	logical partitioning of the host 614 is managed by hypervisor software 654." Chavan '197 at col. 11, ll. 29–39. "FIG. 8 is an illustration of an exemplary system employing an INP 800 with virtualization logic 826 comprised of programmable configuration logic 802 according to embodiments of the present invention. In the alternative embodiment of FIG. 8, the configuration logic 802 may replace the aperture table described above, or may work together with the aperture table. Note that in this alternative embodiment there are no hardware configuration registers in the RC/EP 812 that terminate configuration cycles from the host. In the embodiment of FIG. 8, when PCIe transactions 804 are presented to the RC/EP 812, the headers are evaluated by the configuration logic 802 to determine the classification of the PCIe transaction (i.e. the type of cycle). Depending on the classification, the configuration logic 802 may take no further action with regard to the PCIe transaction, may direct the transaction to a processor via crossbar 810, or may perform an address translation. For example, if the configuration logic 802 determines that a discovery or configuration request (i.e. a configuration cycle) is being presented, the discovery or configuration request is passed through the crossbar 810 to the APEX 814. A free processor in the APEX 814 then executes firmware to respond to and terminate the configuration cycle by providing information and indications back to the host indicating that virtual PCIe devices have been discovered and configured. The particular processor and firmware performs translations from the configuration space of the requestor to the virtual configuration space of the virtual PCIe device being emulated." Chavan '197 at col. 16, ll. 6–33. "FIG. 9 is an illustration of how the common processing elements of MACs for FC 900 and MACs for GbE 902 can be integrated into an efficient CMAC 904 according to embodiments of the present invention. Conventionally, FC frames are received and deserialized by SERDES 906, and are then processed by a FC MAC 900. This processing typically includes 32-bit 8b10b decoding 908, detection of a SFD (a.k.a. a "k" character) and an EFD 910, CRC 912, generating packets from the FC frames 914, and buffering in a FIFO buffer 924. Similarly, GbE packets are received and deserialized by SERDES 916, and are then processed by a GbE MAC 902. This processing typically includes 8-bit 8b10b decoding 918, detection of a SFD and an EFD 920, CRC 922, and FIFO buffering 926. Because of the processing similarities, embodiments of the present invention integrate the data paths for both the FC and GbE protocols in a FC/GbE CMAC 904. Both protocols require SERDES 928, CRC 930 and FIFOs 932 for rate matching, and thus the logic for these tasks is shared in the CMAC 904. However, because the logic (state machine) to perform 8b10b decoding, to detect SFDs and EFDs for each protocol, and to packetize FC frames is different for the two protocols, this logic must be duplicated and switched in or out as

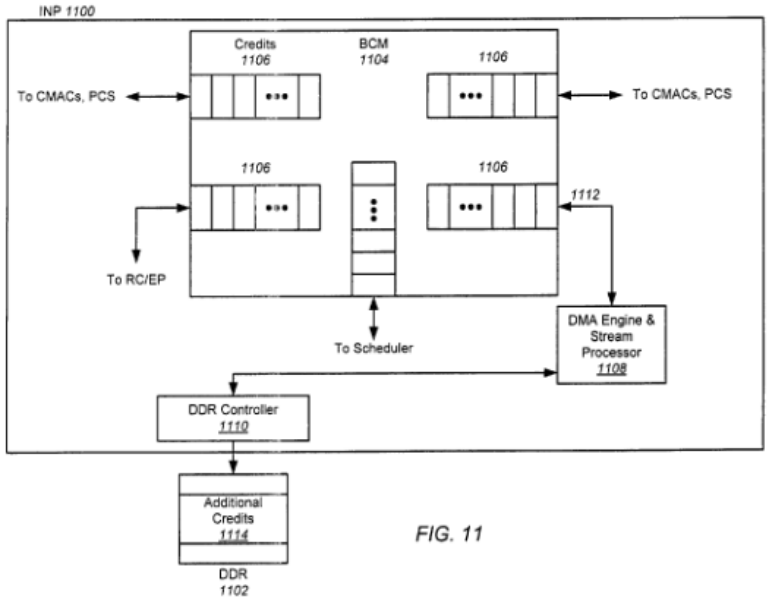
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	needed. To produce a FC/GbE CMAC, multiple bytes must be framed as packets (even though FC uses "frame" terminology). The bytes in a packet are well-defined in the FC and GbE standards. After the SFDs have been detected, logic specific to each protocol may be employed outside the CMAC and switched in or out depending on the protocol to parse the bytes in the packet. The CMACs are configurable by a processor to decode the incoming packets or frames as either FC "packets" or GbE packets." Chavan '197 at col. 17, ll. 22–53. "FIG. 10 illustrates an exemplary INP 1000 employing a hashing scheme for CMAC filtering according to embodiments of the present invention. Note that the representation of INP 1000 of FIG. 10 corresponds to the INP of FIG. 3, but is simplified to emphasize the hashing scheme. In the hashing scheme of FIG. 10, the hashing rules are made scalable without adding hardware, so each CMAC can have many more rules than their hardware registers would allow. In the embodiment of FIG. 10, the scheduler 1002 includes a hash engine 1004. Sophisticated hashing rules can be set for the hash engine 1004. For example, if a CMAC 1006 wants to accept any packet having one of 2 k destination MAC addresses, the hash engine 1004 can be configured such that for every one of the 2 k destination MAC addresses, a hash entry 1008 is created in off-chip DDR 1010. The DDR 1010 can store a large number of hash entries 1008 to achieve scalability. The hash entries 1008 are created at startup, during configuration, when a host configures the INP 1000 to receive the 2 k destination MAC addresses. After the hash entries 1008 are created, if a packet is subsequently received on CMAC 1006 containing one of the 2 k destination MAC addresses, it enters the BCM 1012 and is routed to the scheduler 1002. The scheduler 1002 will do a hash of the destination MAC address in the packet, and attempt to match it with one of the hash entries 1008 in DDR 1010. This is performed by a hardware state machine, so no firmware is required. Although external memory is involved, hardware prefetching is utilized, and thus it may be done at near wire speeds. If a match is found, the packet is then forwarded to one of the APEX processors 1014. On the other hand, if a packet is received that does not have one of the 2 k destination MAC addresses, a bit in the packet is set indicating a "hash miss." The packet is thereafter still forwarded to the APEX block 1014, but the APEX firmware will detect the hash miss bit and know that the packet was not to be received, so the packet is discarded." Chavan '197 at col. 18, ll. 7–42. "74. An intelligent network processor (INP) for connection to a protocol fabric, comprising: a plurality of port interfaces for connection to a protocol fabric, and adjustable to be selectively compatible with FC or GbE; a bidirectional crossbar module (BCM) coupled to receive messages from each of the plurality of port modules, and having crossbar passageways compatible with both FC and GbE messages;

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	a DDR memory interface for connecting to an external DDR memory, for storing and retrieving data messages in queues contained in the DDR memory, and connected to send and receive data messages directly from the BCM; and an application processor module (APEX) coupled to the BCM for controlling the connections between the port interfaces and the DDR memory interface through the BCM, and enabling direct memory access operations between the DDR memory interface and the BCM without data messages passing through the APEX." Chavan '197 at col. 28, l. 60 – col. 29, l. 11. "75. A method for connecting a plurality of port interfaces to a protocol fabric, comprising: providing a plurality of port interfaces for connection to a protocol fabric, the plurality of port interfaces being adjustable to be selectively compatible with FC or GbE; receiving messages from each of the plurality of port interfaces through a bidirectional crossbar module (BCM) having crossbar passageways compatible with both FC and GbE messages; connecting to an external DDR memory for storing and retrieving data messages in queues contained in DDR memory, and for sending and receiving data messages directly from the BCM; and controlling the connections between the port interfaces and the DDR memory interface through the BCM, and enabling direct memory access operations between the DDR memory interface and the BCM without data messages passing through the APEX." Chavan '197 at col. 29, l. 12 – col. 30, l. 14.
[7e] a reservation register implemented in hardware storing a value indicative of available space in said input queue, said reservation register accessible by both said first processing element and said at least one second processing element;	Chavan '197 teaches "a reservation register implemented in hardware storing a value indicative of available space in said input queue, said reservation register accessible by both said first processing element and said at least one second processing element," including at least the following exemplary citations. *See* above at row [1e].
[7f] computer storage storing instructions, which when executed by said at least one second processing element:	Chavan '197 teaches computer storage storing instructions executed by said at least one second processing element, including at least the following exemplary citations. *See* above at row [1f].

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
[7g] receives a command from said first processing element;	Chavan ’197 teaches “receives a command from said first processing element,” including at least the following exemplary citations. For example, Chavan ’197 teaches that the second processing element executes instructions that causes it to receive a command from said first processing element, by disclosing, as non–limiting examples, receiving a credit. The scheduler, the second processing element receives a command from the first processing element. *See, e.g.:*  *FIG. 11* Chavan ’197 at Figure 11. “Conventional BCMs utilize the concept of credits for their ports. Credits are essentially the number of free buffers at a particular port for storing frames or packets. One credit is consumed when a packet is received, and one credit is released when a packet is sent it out of the crossbar to another port (or to the APEX block). Normally the credits for each of the ports are part of the crossbar, in limited on-chip memory. The INP may also employ crossbar credit extensions wherein multiple ports may be given an extended number of credits using off-chip memory. To accomplish this, the external DDR memory is treated as another port on the crossbar. When crossbar logic in the BCM determines that more credits are needed for a particular port, the crossbar logic will make a request to the DMA block, which has access to the off-chip DDR memory through

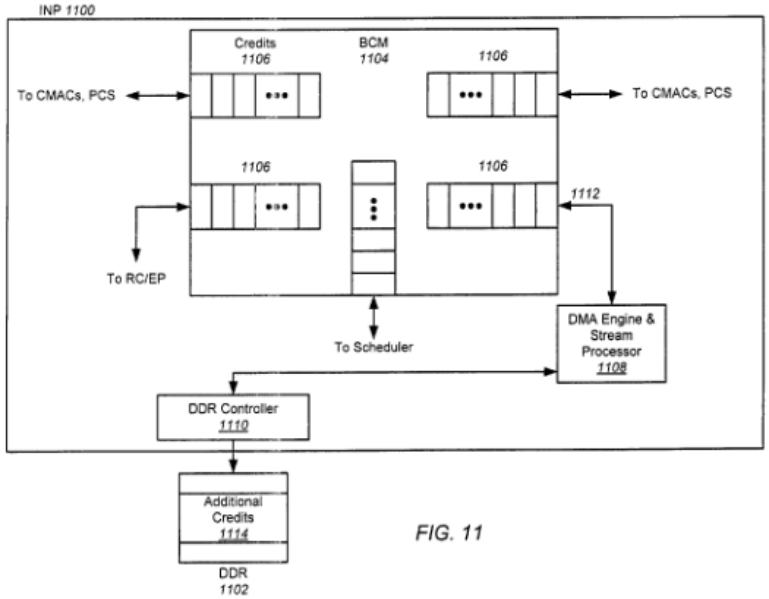
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	the DDR controller, to effectively gain access to additional credits stored in the DDR memory." Chavan '197 at col. 6, l. 54 – col. 7, l. 2. "As shown in the example of FIG. 5, in a conventional PCI system 500, a host 510 may be connected to multiple PCI devices (although only two PCIe devices 512 and 520 are shown) through a PCI bus 514. The PCI standard specifies that configuration space shall be allocated for each PCI device in the form of configuration registers that consume hardware resources. In particular, each PCI device is assigned one or more mutually-exclusive blocks of memory in the host's memory-mapped memory space 516. In the example of FIG. 5, host 510 has been assigned memory space 518, PCI device 512 has been assigned memory space 524 (designated by the letter "m"), and PCI device 520 has been assigned memory space 526 (designated by the letter "n"). According to the PCI standard, as part of the configuration space allocated to each PCI device, each PCI device may be allocated up to eight BARs 502 in a configuration header 536, referred to as BAR0-BAR7 in FIG. 5. Each BAR contains the start address 504 of a block of memory space in memory 516 assigned to a function of the PCI device, and the length 506 and type of memory space 508 (memory or I/O) required by the function. The start address 504 of the BAR enables a memory or I/O address decoder in the PCI device to detect device accesses whose addresses fall within the block of memory space specified by the BAR. Additionally, other configuration registers 538 (including doorbells and status registers) may be allocated for each PCI device and assigned memory space such that once mapped, a host and the PCI device can communicate using that memory space. During configuration of the PCI system, the host 510 performs a configuration cycle where the host first discovers what PCI devices exist on the PCI bus 514, and then configures the configuration registers for those PCI devices. To implement the discovery process, a host's configuration software sends a discovery request out over the PCI bus 514, which reads a subset of each PCI device's configuration registers (e.g. vendor IDentification (ID) register 530 and device ID register 532) in configuration header 536 to determine the presence of the PCI device and its device type. The host 510 also accesses other configuration registers 534 in the PCI device to determine how many blocks of memory the PCI device requires. To configure a particular PCI device, the host sends a configuration request to assign memory space to the PCI device by assigning an address to the BARs of that PCI device. Note that the length field 506 and the type field 508 of each BAR are generally fixed, and only the address field 504 is programmable. When a particular PCI device has been discovered and configured, the PCI device responds back to the host, resulting in the termination of the configuration cycle for that PCI device." Chavan '197 at col. 10, l. 36 – col. 11, l. 17.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	"Conventional BCMs utilize the concept of credits for their ports. Credits are essentially the number of free buffers at a particular port. For example, if a port on a BCM has eight credits, eight frames of data could be received into that port and stored in the buffers without having to send an acknowledgement. One credit is consumed when a packet is received, and one credit is released when a packet is sent it out of the crossbar through that port. Normally the credits for each of the ports are part of the crossbar, in on-chip memory. However, on-chip memory is expensive, and thus traditionally the number of credits allocated to the ports on a BCM has been limited. This is sufficient for short-haul communications, but when using the links for long-haul purposes such as distance of 100 km, many packets may be in the pipe, so more credits are needed per port. Increasing the size of the on-chip crossbar is expensive. However, external memory has traditionally not been used to store credits because it would be too slow to control the external memory." Chavan '197 at col. 19, l. 66 – col. 20, l. 16. "FIG. 11 is an illustration of an INP 1100 employing crossbar credit extensions according to embodiments of the present invention. In FIG. 11, a BCM 1104 has multiple ports, each port having a number of credits 1106 for buffering frames as they pass through the BCM 1104. The credit extensions of the present invention extend the number credits available to each port on the BCM 1104 using off-chip memory 1102. As a result, on a port by port basis, more credits can be configured than are available in the crossbar memory 1104. The external DDR memory 1102 is treated as another port on the crossbar 1104, or just an extension of the crossbar. When crossbar logic in the BCM 1104 determines that more credits are needed for a particular port, the crossbar logic will make a request to the DMA block 1108, which has access to the off-chip DDR memory 1102 through the DDR controller 1110, to effectively gain access to additional credits 1114. The DMA engine 1108 treats the off-chip DDR memory 1102 as though it were on-chip memory. When more credits are needed, the off-chip DDR 1102 can be accessed through a port 1112 on the crossbar 1104 and through the DMA engine 1108 to store frames in the DDR 1102 and effectively consume additional credits. Note that this is performed using gates, because the message frames are received so fast that processors would not be able to selectively decide what frames to put in DDR 1102 and which ones to receive from DDR. Instead, the DMA engine 1108 does this work." Chavan '197 at col. 20, ll. 17–42.
[7h] accesses said reservation register and reads said stored value;	Chavan '197 teaches "accesses said reservation register and reads said stored value," including at least the following exemplary citations. *See* above at row [1g].

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
[7i] determines when said read value indicates available space in said input queue to issue said command;	Chavan '197 teaches "determines when said read value indicates available space in said input queue to issue said command," including at least the following exemplary citations. *See* above at row [1h].
[7j] issues said command to said input queue on behalf of said first processing element; and	Chavan '197 teaches "issues said command to said input queue on behalf of said first processing element," including at least the following exemplary citations. For example, Chavan '197 teaches that the second processing element executes instructions that causes it to issue said command to said input queue on behalf of said first processing element, by disclosing, as non–limiting examples, notifying that a credit can be used and writing into the buffer. The scheduler, the second processing element, issues the command to the input queue on behalf of the first processing element. *See, e.g.:*  *FIG. 11* Chavan '197 at Figure 11. "The DMA engine and stream processor 386 is connected to the BCM 312 through bus 388, and is connected to the DDR SDRAM controller 390 through bus 392. The function of the controller 390 is to control the sending and receiving of data to and from an external DDR memory module (not shown) connected to the bus

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	394 of the controller 390. The controller 390 includes capabilities for providing RMW, statistics collection, XOR masking for data being transferred to or from the external DDR memory module. The data pathway through the bus 392 to the DMA engine and stream processor 386 and through the bus 388 to the BCM 312 allows DMA memory operations to occur directly with the controller 390 and its attached DDR memory module, without data flow through the APEX 316, for data being exchanged with other sources including the PCIe capabilities accessed through the bus 358 and RC/EP module 356, and the FC and GbE capabilities accessed through the buses 336 and 338 with the PCS modules 332 and 334. The ability to allow direct access for DMA operations through the BCM 312 with the DDR SDRAM controller 390 saves time that would be otherwise consumed by data being transferred into and out of the APEX 316. The controller 390 is also connected to the scheduler 314 through the bus 396. The controller 390 creates queues in the external DDR memory for storing data messages, and writes to and reads from such queues. The use of such queues, and the direct memory access to the crossbar BCM 312 (through the buses 388 and 392, controller 390, and DMA engine and stream processor 386) effectively increases the size of the crossbar BCM 312, and allows more switching pathways to be created than if the BCM 312 were used without the external DDR memory attached to connection 394." Chavan '197 at col. 8, l. 66 – col. 9, l. 29. "As shown in the example of FIG. 5, in a conventional PCI system 500, a host 510 may be connected to multiple PCI devices (although only two PCIe devices 512 and 520 are shown) through a PCI bus 514. The PCI standard specifies that configuration space shall be allocated for each PCI device in the form of configuration registers that consume hardware resources. In particular, each PCI device is assigned one or more mutually-exclusive blocks of memory in the host's memory-mapped memory space 516. In the example of FIG. 5, host 510 has been assigned memory space 518, PCI device 512 has been assigned memory space 524 (designated by the letter "m"), and PCI device 520 has been assigned memory space 526 (designated by the letter "n"). According to the PCI standard, as part of the configuration space allocated to each PCI device, each PCI device may be allocated up to eight BARs 502 in a configuration header 536, referred to as BAR0-BAR7 in FIG. 5. Each BAR contains the start address 504 of a block of memory space in memory 516 assigned to a function of the PCI device, and the length 506 and type of memory space 508 (memory or I/O) required by the function. The start address 504 of the BAR enables a memory or I/O address decoder in the PCI device to detect device accesses whose addresses fall within the block of memory space specified by the BAR. Additionally, other configuration registers 538 (including doorbells and status registers) may be allocated for each PCI device and assigned memory space such that once mapped, a host and the PCI device can communicate using that memory space.

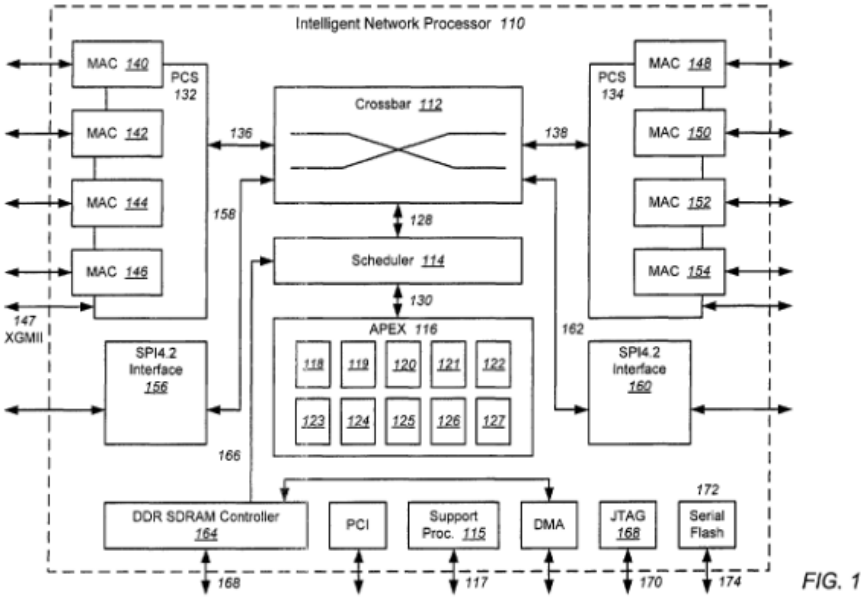
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	During configuration of the PCI system, the host 510 performs a configuration cycle where the host first discovers what PCI devices exist on the PCI bus 514, and then configures the configuration registers for those PCI devices. To implement the discovery process, a host's configuration software sends a discovery request out over the PCI bus 514, which reads a subset of each PCI device's configuration registers (e.g. vendor IDentification (ID) register 530 and device ID register 532) in configuration header 536 to determine the presence of the PCI device and its device type. The host 510 also accesses other configuration registers 534 in the PCI device to determine how many blocks of memory the PCI device requires. To configure a particular PCI device, the host sends a configuration request to assign memory space to the PCI device by assigning an address to the BARs of that PCI device. Note that the length field 506 and the type field 508 of each BAR are generally fixed, and only the address field 504 is programmable. When a particular PCI device has been discovered and configured, the PCI device responds back to the host, resulting in the termination of the configuration cycle for that PCI device." Chavan '197 at col. 10, l. 36 – col. 11, l. 17. "Conventional BCMs utilize the concept of credits for their ports. Credits are essentially the number of free buffers at a particular port. For example, if a port on a BCM has eight credits, eight frames of data could be received into that port and stored in the buffers without having to send an acknowledgement. One credit is consumed when a packet is received, and one credit is released when a packet is sent it out of the crossbar through that port. Normally the credits for each of the ports are part of the crossbar, in on-chip memory. However, on-chip memory is expensive, and thus traditionally the number of credits allocated to the ports on a BCM has been limited. This is sufficient for short-haul communications, but when using the links for long-haul purposes such as distance of 100 km, many packets may be in the pipe, so more credits are needed per port. Increasing the size of the on-chip crossbar is expensive. However, external memory has traditionally not been used to store credits because it would be too slow to control the external memory." Chavan '197 at col. 19, l. 66 – col. 20, l. 16. "FIG. 11 is an illustration of an INP 1100 employing crossbar credit extensions according to embodiments of the present invention. In FIG. 11, a BCM 1104 has multiple ports, each port having a number of credits 1106 for buffering frames as they pass through the BCM 1104. The credit extensions of the present invention extend the number credits available to each port on the BCM 1104 using off-chip memory 1102. As a result, on a port by port basis, more credits can be configured than are available in the crossbar memory 1104. The external DDR memory 1102 is treated as another port on the crossbar 1104, or just an extension of the crossbar. When crossbar logic in the BCM 1104 determines that more credits are needed for a particular port, the crossbar logic will make a request to the DMA block 1108, which has access to the off-chip DDR memory 1102 through the DDR controller 1110, to effectively gain access to additional credits 1114.

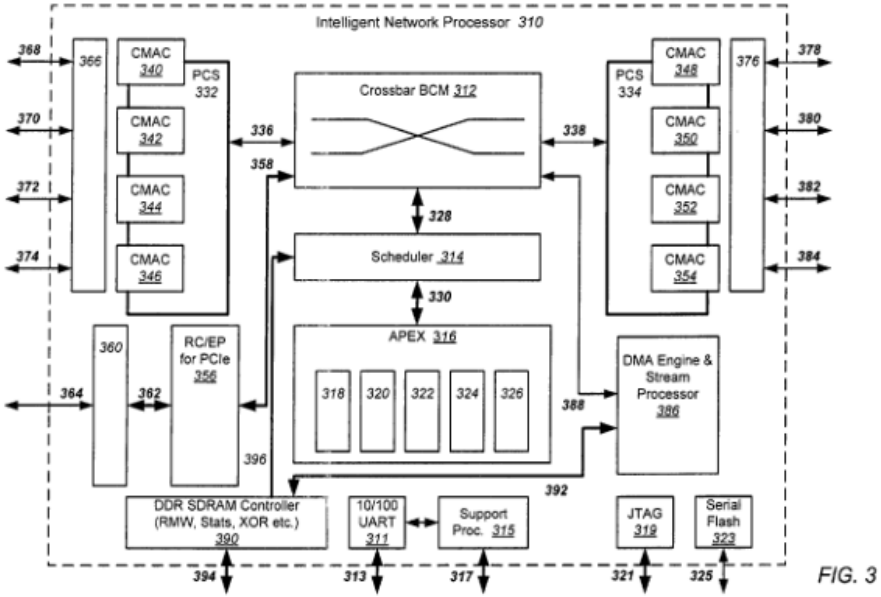
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	The DMA engine 1108 treats the off-chip DDR memory 1102 as though it were on-chip memory. When more credits are needed, the off-chip DDR 1102 can be accessed through a port 1112 on the crossbar 1104 and through the DMA engine 1108 to store frames in the DDR 1102 and effectively consume additional credits. Note that this is performed using gates, because the message frames are received so fast that processors would not be able to selectively decide what frames to put in DDR 1102 and which ones to receive from DDR. Instead, the DMA engine 1108 does this work." Chavan '197 at col. 20, ll. 17–42.
[7k] wherein said first processing element receives a response corresponding to said command from said output queue.	Chavan '197 teaches "wherein said first processing element receives a response corresponding to said command from said output queue," including at least the following exemplary citations. For example, Chavan '197 teaches wherein said first processing element receives a response corresponding to said command from said output queue, by disclosing, as non–limiting examples, releasing the credit when the frame is processed. The processor receives a response from the output queue corresponding to the command. *See, e.g.:*  FIG. 11 Chavan '197 at Figure 11.

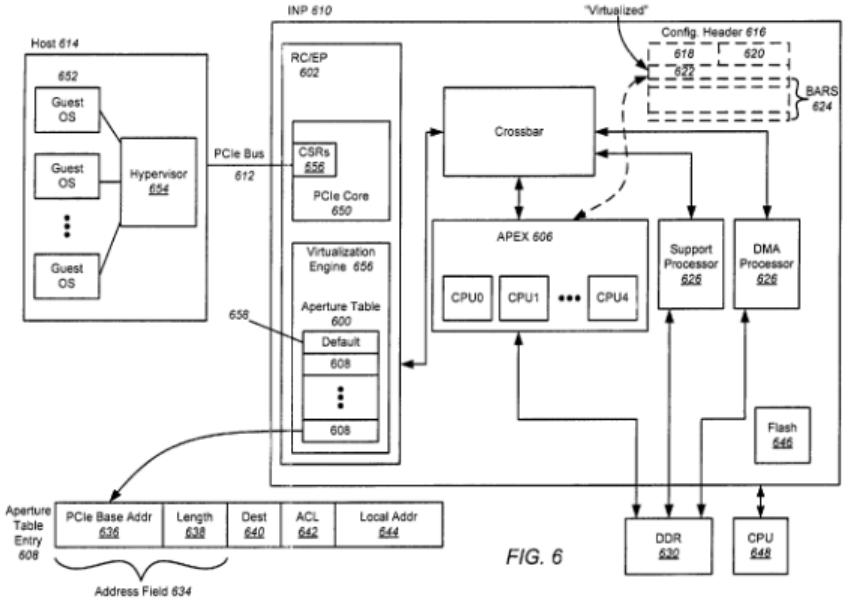
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	"INPs are used in computer network systems for providing HBA, router, and switch capabilities for particular protocol networks connected to a host through a host interface. An example of an INP 110 is shown in FIG. 1, which may be formed on a single silicon integrated circuit die packaged inside a ceramic, metal, or plastic housing to form a unitary component. INP 110 includes a shared memory crossbar module 112 which is controlled by a flow identification/scheduler module 114 through a bus 128. The flow identification/scheduler module 114 operates under the control of the Application Processor (APEX) module 116 acting through the bus 130, and maps the passages of crossbar locations inside the shared memory crossbar module 112. The APEX 116 in the example of FIG. 1 contains ten core processors for executing firmware, denoted as processors 118-127. The shared memory crossbar module 112 is connected to Protocol Coordination Stage (PCS) modules 132 and 134 through buses 136 and 138, respectively. PCS module 132 is connected to and controls the operation of Media Access Controller (MAC) modules 140, 142, 144, and 146. The MAC modules 140, 142, 144, and 146 are identical hardware interfaces capable of each communicating using the FC protocol at 1 Gigabits per second (Gbps) or 2 Gbps speeds. The speed and protocol settings for the for the MAC modules 140, 142, 144, and 146 are controlled by the PCS 132, which operates in response to control signals from a bus (not shown) connecting the PCS 132 to the APEX 116. A parallel 10 Gbit (X Gbit) Medium-Independent Interface (XGMII) port 147 may be provided from the PCS 132, as determined by control signals originating from the APEX 116. The 10 Gb XGMII interface is not coupled to any of the four MAC modules." Chavan '197 at col. 1, ll. 23–53. "A Root Complex and EndPoint (RC/EP) module is connected to the BCM, and is connected to a SERDES for use with PCIe bus signals. A Direct Memory Access (DMA) engine and stream processor is connected to the BCM, and is also connected to a DDR SDRAM controller. The function of the controller is to control the sending and receiving of data to and from an external DDR memory module. A data pathway from the DMA engine and stream processor to the BCM allows DMA memory operations to occur directly with the controller and its attached DDR memory module, without data flow through the APEX, for data being exchanged with other sources. The controller is also connected to the scheduler, and creates queues in the external DDR memory for storing data messages, and also writes to and reads from such queues." Chavan '197 at col. 3, ll. 48–62. "The DMA engine and stream processor 386 is connected to the BCM 312 through bus 388, and is connected to the DDR SDRAM controller 390 through bus 392. The function of the controller 390 is to control the sending and receiving of data to and from an external DDR memory module (not shown) connected to the bus 394 of the controller 390. The controller 390 includes capabilities for providing RMW, statistics collection,

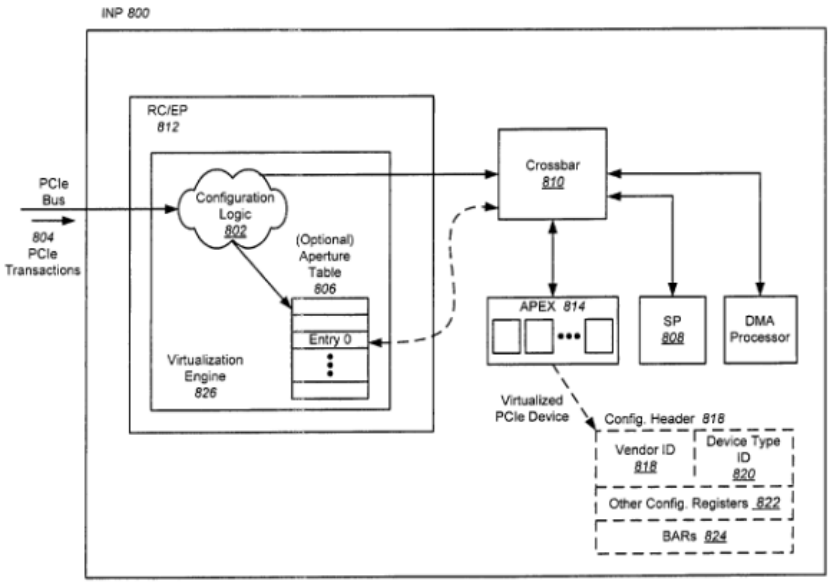
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	XOR masking for data being transferred to or from the external DDR memory module. The data pathway through the bus 392 to the DMA engine and stream processor 386 and through the bus 388 to the BCM 312 allows DMA memory operations to occur directly with the controller 390 and its attached DDR memory module, without data flow through the APEX 316, for data being exchanged with other sources including the PCIe capabilities accessed through the bus 358 and RC/EP module 356, and the FC and GbE capabilities accessed through the buses 336 and 338 with the PCS modules 332 and 334. The ability to allow direct access for DMA operations through the BCM 312 with the DDR SDRAM controller 390 saves time that would be otherwise consumed by data being transferred into and out of the APEX 316. The controller 390 is also connected to the scheduler 314 through the bus 396. The controller 390 creates queues in the external DDR memory for storing data messages, and writes to and reads from such queues. The use of such queues, and the direct memory access to the crossbar BCM 312 (through the buses 388 and 392, controller 390, and DMA engine and stream processor 386) effectively increases the size of the crossbar BCM 312, and allows more switching pathways to be created than if the BCM 312 were used without the external DDR memory attached to connection 394." Chavan '197 at col. 8, l. 66 – col. 9, l. 29.
8. The system of claim 7, wherein said input queue is a first-in-first-out (FIFO) buffer.	Chavan '197 teaches "[t]he system of claim 7, wherein said input queue is a first-in-first-out (FIFO) buffer," including at least the following exemplary citations. *See* above at row 2.
9. The system of claim 7, wherein said system is part of a storage area network (SAN) controller.	Chavan '197 teaches "[t]he system of claim 7, wherein said system is part of a storage area network (SAN) controller," including at least the following exemplary citations. *See* above at row 3.
10. The system of claim 7, wherein said system is part of a network attached storage (NAS) controller.	Chavan '197 teaches "[t]he system of claim 7, wherein said system is part of a network attached storage (NAS) controller," including at least the following exemplary citations. *See* above at row 4.
11. The system of claim 7, wherein said stored value is decremented by a predefined value prior to returning said stored value to said	Chavan '197 teaches "[t]he system of claim 7, wherein said stored value is decremented by a predefined value prior to returning said stored value to said at least one second processing element," including at least the following exemplary citations.

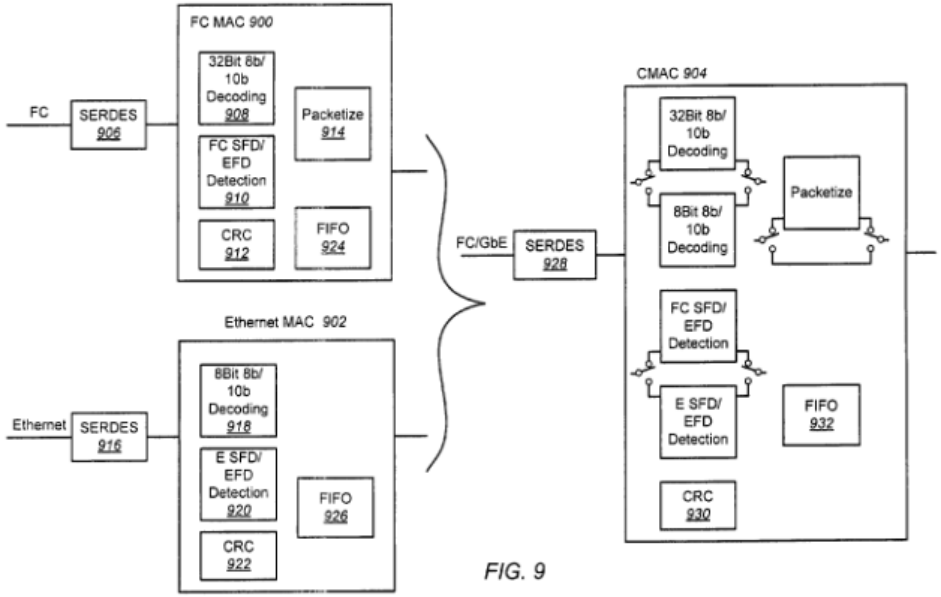
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
at least one second processing element.	*See* above at row 5.
12. The system of claim 7, wherein at least one third processing element executes said command and notifies said output queue to issue said response.	Chavan '197 teaches "[t]he system of claim 7, wherein at least one third processing element executes said command and notifies said output queue to issue said response," including at least the following exemplary citations. *See* above at row 6.
[13pre] A system for dividing and synchronizing a processing task across a plurality of processing elements comprising:	Chavan '197 teaches "[a] system for dividing and synchronizing a processing task across a plurality of processing elements," including at least the following exemplary citations. *See* above at row [1pre].
[13a] an input queue implemented in hardware;	Chavan '197 teaches "an input queue implemented in hardware," including at least the following exemplary citations. *See* above at row [1a].
[13b] an output queue implemented in hardware;	Chavan '197 teaches "an output queue implemented in hardware," including at least the following exemplary citations. *See* above at row [1b].
[13c] a first processing element having access to said input queue and said output queue;	Chavan '197 teaches "a first processing element having access to said input queue and said output queue," including at least the following exemplary citations. *See* above at row [1c].
[13d] at least one second processing element in communication with said input queue and said output queue;	Chavan '197 teaches "at least one second processing element in communication with said input queue and said output queue," including at least the following exemplary citations. For example, Chavan '197 teaches at least one second processing element in communication with said input queue and said output queue, by disclosing, as non–limiting examples, a scheduler that communicates with

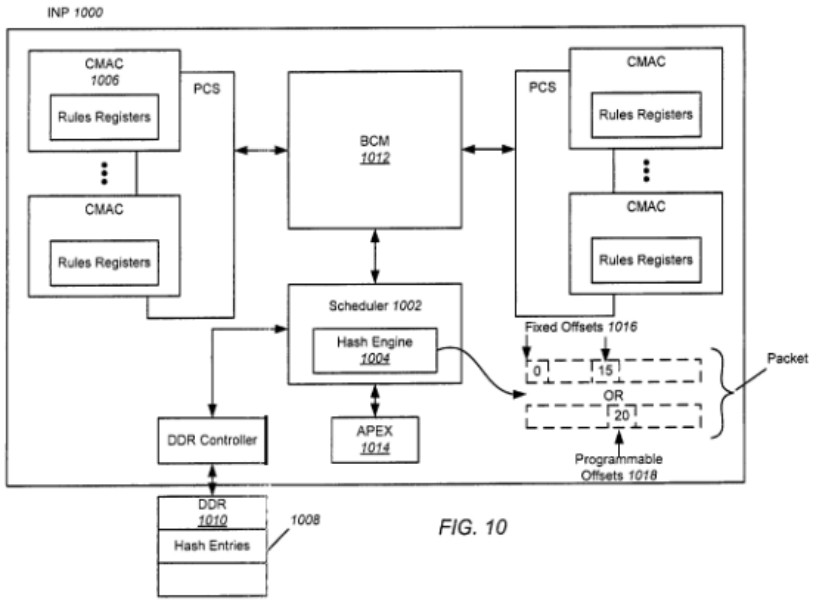
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	the processors within the APEX and the crossbar. The scheduler, the second processing element is in communication with the input and output queue. *See, e.g.:*  Chavan ’197 at Figure 1.

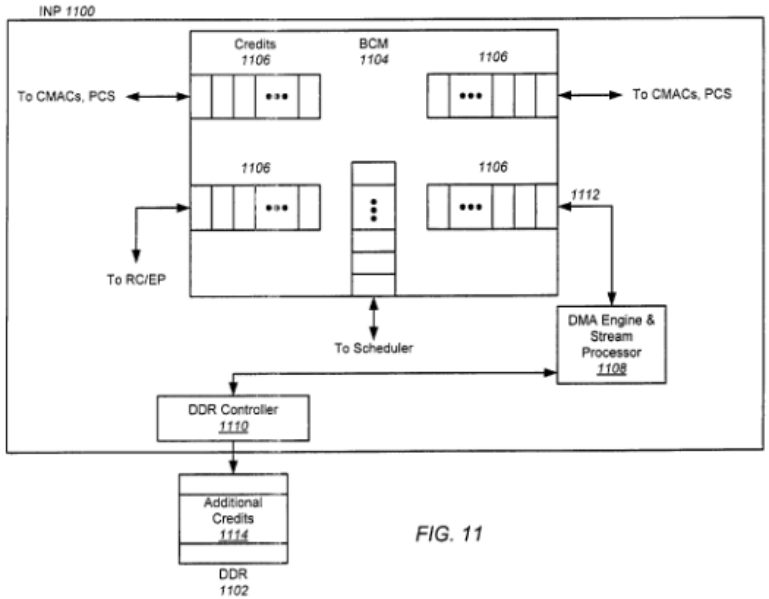
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan ’197 at Figure 3.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan ’197 at Figure 6.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 8 Chavan ’197 at Figure 8.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan ’197 at Figure 9.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Chavan ’197 at Figure 10.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 11 Chavan ’197 at Figure 11. “INPs are used in computer network systems for providing HBA, router, and switch capabilities for particular protocol networks connected to a host through a host interface. An example of an INP 110 is shown in FIG. 1, which may be formed on a single silicon integrated circuit die packaged inside a ceramic, metal, or plastic housing to form a unitary component. INP 110 includes a shared memory crossbar module 112 which is controlled by a flow identification/scheduler module 114 through a bus 128. The flow identification/scheduler module 114 operates under the control of the Application Processor (APEX) module 116 acting through the bus 130, and maps the passages of crossbar locations inside the shared memory crossbar module 112. The APEX 116 in the example of FIG. 1 contains ten core processors for executing firmware, denoted as processors 118-127. The shared memory crossbar module 112 is connected to Protocol Coordination Stage (PCS) modules 132 and 134 through buses 136 and 138, respectively. PCS module 132 is connected to and controls the operation of Media Access Controller (MAC) modules 140, 142, 144, and 146. The MAC modules 140, 142, 144, and 146 are identical hardware interfaces capable of each communicating using the FC protocol at 1 Gigabits per second (Gbps) or 2 Gbps speeds. The speed and protocol settings for the for the MAC modules 140, 142, 144, and 146 are controlled by the PCS 132, which operates in response to control signals from a bus (not shown)

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	connecting the PCS 132 to the APEX 116. A parallel 10 Gbit (X Gbit) Medium-Independent Interface (XGMII) port 147 may be provided from the PCS 132, as determined by control signals originating from the APEX 116. The 10 Gb XGMII interface is not coupled to any of the four MAC modules." Chavan '197 at col. 1, ll. 23–53. "A Double Data Rate (DDR) Synchronous Dynamic Random Access Memory (SDRAM) controller 164 is connected to the scheduler 114 through bus 166. The function of the controller 164 is to control the sending and receiving of data to and from an external DDR memory module (not shown) connected to an external bus 168 of the controller. A support processor 115 handles certain processing tasks not otherwise covered by the APEX 116, and has an external connection 117. A Joint Test Action Group (JTAG) module 168 has an external connection 170, and is used for boundary scan testing of the hardware contained within INP 110. Serial flash memory 172 has an external connection 174, and is a non-volatile reprogrammable memory which may be loaded through the connection to store firmware for operation of APEX 116 and support processor 115." Chavan '197 at col. 2, ll. 14–28. "The present invention is directed to INPs for providing HBA, router, and switch capabilities on FC and GbE protocol networks connected to a single host or multiple hosts (as in the case of multi-root) through a PCIe host interface. The INP includes a Bi-directional Crossbar Module (BCM) which is controlled by a flow identification and scheduler module. The scheduler operates under the control of the APEX, and maps the passages of crossbar locations inside the BCM. The APEX contains multiple core processors for executing firmware. The BCM is connected to PCS modules, which are connected to and control the operation of Combined Media Access Controller (CMAC) modules capable of each communicating using multiple protocols. The speed and protocol settings for the CMAC modules are controlled by the PCS. The PCS also has trunking modes in which it coordinates the operation of the CMAC modules together as a single trunk. A combined SERDES is coupled to each of the CMACs and selectively provides, under control of signals coupled from the APEX, electronic signals in one of multiple protocols (e.g. the FC and GbE protocols), or trunking protocols." Chavan '197 at col. 3, ll. 28–47. "A Root Complex and EndPoint (RC/EP) module is connected to the BCM, and is connected to a SERDES for use with PCIe bus signals. A Direct Memory Access (DMA) engine and stream processor is connected to the BCM, and is also connected to a DDR SDRAM controller. The function of the controller is to control the sending and receiving of data to and from an external DDR memory module. A data pathway from the DMA engine and stream processor to the BCM allows DMA memory operations to occur directly with the

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	controller and its attached DDR memory module, without data flow through the APEX, for data being exchanged with other sources. The controller is also connected to the scheduler, and creates queues in the external DDR memory for storing data messages, and also writes to and reads from such queues." Chavan '197 at col. 3, ll. 48–62. "An INP may also utilize programmable configuration logic that may replace the aperture table described above, or may work together with the aperture table. In this embodiment, the headers of PCIe transactions are evaluated by the configuration logic to determine the classification of the PCIe transaction (i.e. the type of cycle). Depending on the classification, the configuration logic may take certain actions. For example, if the configuration logic determines that a discovery or configuration request (i.e. a configuration cycle) is being presented, the discovery or configuration request is passed through the crossbar to the APEX. A free processor in the APEX then executes firmware to respond to and terminate the configuration cycle by providing information and indications back to the host indicating that virtual PCIe devices have been discovered and configured. The particular processor and firmware performs translations from the configuration space of the requestor to the virtual configuration space of the virtual PCIe device being emulated. The common processing elements of FC and GbE MACs may be integrated into an efficient CMAC. Both protocols require SERDES, Cyclic Redundancy Checking (CRC) and First-In-First-Out (FIFO) buffers for rate matching, and thus the logic for these tasks is shared in the CMAC. However, because the logic (state machine) to perform 8b10b decoding, to detect Start of Frame Delimiters (SFDs) and End of Frame Delimiters (EFDs) for each protocol, and to packetize FC frames is different for the two protocols, this logic is duplicated and switched in or out as needed. To produce a FC/GbE CMAC, multiple bytes must be framed as packets. After the SFDs have been detected, logic specific to each protocol may be employed outside the CMAC and switched in or out depending on the protocol to parse the bytes in the packet. The CMACs are configurable by a processor to decode the incoming packets or frames as either FC "packets" or GbE packets." Chavan '197 at col. 5, l. 50 – col. 6, l. 16. "FIG. 3 is an illustration of an INP 310 implemented according to embodiments of the present invention. INP 310 may be composed of a single silicon integrated circuit die, which may be packaged inside a ceramic, metal, or plastic housing to form a unitary component. NP 310 includes a BCM 312 which is controlled by a flow identification and scheduler module 314 through a bus 328. The BCM 312 may be a multi-ported shared Random Access Memory (RAM) having memory locations mapped as passages, and operating at 224 Gbps. The scheduler 314 operates under the control of the APEX 316 acting through the bus 330, and maps the passages of crossbar locations inside the BCM 312. The scheduler 314 pre-processes received frames,

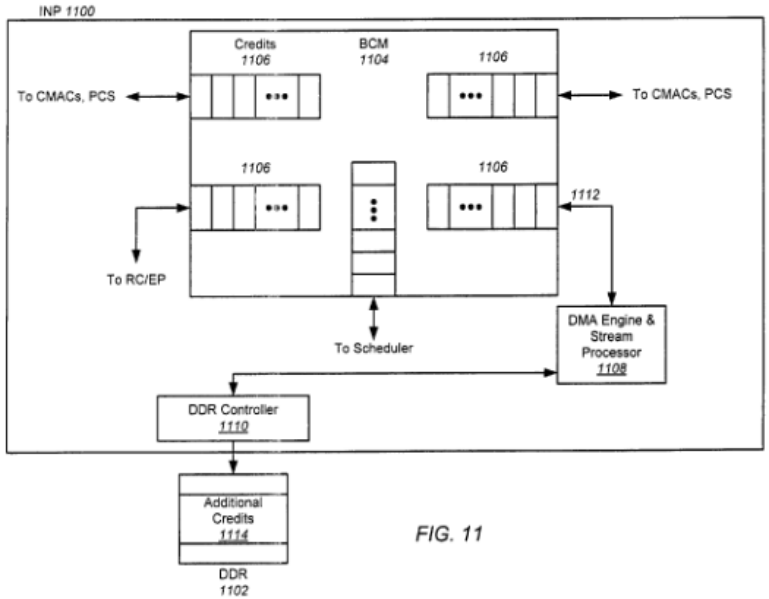
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	packets, and PCIe requests before they are acted upon by APEX 316. The scheduler 314 also parses received FC frames into correct sequence, and parses received GbE packets (in Internet Protocol (IP) format) into a Transport Control Protocol (TCP) flow. The APEX 316 preferably contains five core processors for executing firmware, denoted as processors 318, 320, 322, 324, and 326. The processors 318, 322, 324, and 326 may be core model ARC750 as provided by ARC, Inc., which preferably each have the capability of executing 32 bit Reduced Instruction Set Computer (RISC) firmware with a clock speed of 500 MHz, and may each include 32 KB of dedicated instruction memory (not shown).” Chavan ’197 at col. 7, l. 49 – col. 8, l. 6. “The DMA engine and stream processor 386 is connected to the BCM 312 through bus 388, and is connected to the DDR SDRAM controller 390 through bus 392. The function of the controller 390 is to control the sending and receiving of data to and from an external DDR memory module (not shown) connected to the bus 394 of the controller 390. The controller 390 includes capabilities for providing RMW, statistics collection, XOR masking for data being transferred to or from the external DDR memory module. The data pathway through the bus 392 to the DMA engine and stream processor 386 and through the bus 388 to the BCM 312 allows DMA memory operations to occur directly with the controller 390 and its attached DDR memory module, without data flow through the APEX 316, for data being exchanged with other sources including the PCIe capabilities accessed through the bus 358 and RC/EP module 356, and the FC and GbE capabilities accessed through the buses 336 and 338 with the PCS modules 332 and 334. The ability to allow direct access for DMA operations through the BCM 312 with the DDR SDRAM controller 390 saves time that would be otherwise consumed by data being transferred into and out of the APEX 316. The controller 390 is also connected to the scheduler 314 through the bus 396. The controller 390 creates queues in the external DDR memory for storing data messages, and writes to and reads from such queues. The use of such queues, and the direct memory access to the crossbar BCM 312 (through the buses 388 and 392, controller 390, and DMA engine and stream processor 386) effectively increases the size of the crossbar BCM 312, and allows more switching pathways to be created than if the BCM 312 were used without the external DDR memory attached to connection 394.” Chavan ’197 at col. 8, l. 66 – col. 9, l. 29. “FIG. 6 is an illustration of an exemplary system employing an INP 610 according to embodiments of the present invention. Note that INP 610 is a simplified representation of the INP of FIG. 3, emphasizing the PCIe interface. FIG. 6 illustrates a so-called “single root” system configuration comprised of a single root complex (the host 614) and an INP 610 representing one of perhaps several endpoints. In the example of FIG. 6, the host is logically partitioned into multiple virtual servers, each running a separate “guest” OS 652. The

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	logical partitioning of the host 614 is managed by hypervisor software 654." Chavan '197 at col. 11, ll. 29–39. "FIG. 8 is an illustration of an exemplary system employing an INP 800 with virtualization logic 826 comprised of programmable configuration logic 802 according to embodiments of the present invention. In the alternative embodiment of FIG. 8, the configuration logic 802 may replace the aperture table described above, or may work together with the aperture table. Note that in this alternative embodiment there are no hardware configuration registers in the RC/EP 812 that terminate configuration cycles from the host. In the embodiment of FIG. 8, when PCIe transactions 804 are presented to the RC/EP 812, the headers are evaluated by the configuration logic 802 to determine the classification of the PCIe transaction (i.e. the type of cycle). Depending on the classification, the configuration logic 802 may take no further action with regard to the PCIe transaction, may direct the transaction to a processor via crossbar 810, or may perform an address translation. For example, if the configuration logic 802 determines that a discovery or configuration request (i.e. a configuration cycle) is being presented, the discovery or configuration request is passed through the crossbar 810 to the APEX 814. A free processor in the APEX 814 then executes firmware to respond to and terminate the configuration cycle by providing information and indications back to the host indicating that virtual PCIe devices have been discovered and configured. The particular processor and firmware performs translations from the configuration space of the requestor to the virtual configuration space of the virtual PCIe device being emulated." Chavan '197 at col. 16, ll. 6–33. "FIG. 9 is an illustration of how the common processing elements of MACs for FC 900 and MACs for GbE 902 can be integrated into an efficient CMAC 904 according to embodiments of the present invention. Conventionally, FC frames are received and deserialized by SERDES 906, and are then processed by a FC MAC 900. This processing typically includes 32-bit 8b10b decoding 908, detection of a SFD (a.k.a. a "k" character) and an EFD 910, CRC 912, generating packets from the FC frames 914, and buffering in a FIFO buffer 924. Similarly, GbE packets are received and deserialized by SERDES 916, and are then processed by a GbE MAC 902. This processing typically includes 8-bit 8b10b decoding 918, detection of a SFD and an EFD 920, CRC 922, and FIFO buffering 926. Because of the processing similarities, embodiments of the present invention integrate the data paths for both the FC and GbE protocols in a FC/GbE CMAC 904. Both protocols require SERDES 928, CRC 930 and FIFOs 932 for rate matching, and thus the logic for these tasks is shared in the CMAC 904. However, because the logic (state machine) to perform 8b10b decoding, to detect SFDs and EFDs for each protocol, and to packetize FC frames is different for the two protocols, this logic must be duplicated and switched in or out as

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	needed. To produce a FC/GbE CMAC, multiple bytes must be framed as packets (even though FC uses "frame" terminology). The bytes in a packet are well-defined in the FC and GbE standards. After the SFDs have been detected, logic specific to each protocol may be employed outside the CMAC and switched in or out depending on the protocol to parse the bytes in the packet. The CMACs are configurable by a processor to decode the incoming packets or frames as either FC "packets" or GbE packets." Chavan '197 at col. 17, ll. 22–53. "FIG. 10 illustrates an exemplary INP 1000 employing a hashing scheme for CMAC filtering according to embodiments of the present invention. Note that the representation of INP 1000 of FIG. 10 corresponds to the INP of FIG. 3, but is simplified to emphasize the hashing scheme. In the hashing scheme of FIG. 10, the hashing rules are made scalable without adding hardware, so each CMAC can have many more rules than their hardware registers would allow. In the embodiment of FIG. 10, the scheduler 1002 includes a hash engine 1004. Sophisticated hashing rules can be set for the hash engine 1004. For example, if a CMAC 1006 wants to accept any packet having one of 2 k destination MAC addresses, the hash engine 1004 can be configured such that for every one of the 2 k destination MAC addresses, a hash entry 1008 is created in off-chip DDR 1010. The DDR 1010 can store a large number of hash entries 1008 to achieve scalability. The hash entries 1008 are created at startup, during configuration, when a host configures the INP 1000 to receive the 2 k destination MAC addresses. After the hash entries 1008 are created, if a packet is subsequently received on CMAC 1006 containing one of the 2 k destination MAC addresses, it enters the BCM 1012 and is routed to the scheduler 1002. The scheduler 1002 will do a hash of the destination MAC address in the packet, and attempt to match it with one of the hash entries 1008 in DDR 1010. This is performed by a hardware state machine, so no firmware is required. Although external memory is involved, hardware prefetching is utilized, and thus it may be done at near wire speeds. If a match is found, the packet is then forwarded to one of the APEX processors 1014. On the other hand, if a packet is received that does not have one of the 2 k destination MAC addresses, a bit in the packet is set indicating a "hash miss." The packet is thereafter still forwarded to the APEX block 1014, but the APEX firmware will detect the hash miss bit and know that the packet was not to be received, so the packet is discarded." Chavan '197 at col. 18, ll. 7–42. "74. An intelligent network processor (INP) for connection to a protocol fabric, comprising: a plurality of port interfaces for connection to a protocol fabric, and adjustable to be selectively compatible with FC or GbE; a bidirectional crossbar module (BCM) coupled to receive messages from each of the plurality of port modules, and having crossbar passageways compatible with both FC and GbE messages;

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	a DDR memory interface for connecting to an external DDR memory, for storing and retrieving data messages in queues contained in the DDR memory, and connected to send and receive data messages directly from the BCM; and an application processor module (APEX) coupled to the BCM for controlling the connections between the port interfaces and the DDR memory interface through the BCM, and enabling direct memory access operations between the DDR memory interface and the BCM without data messages passing through the APEX." Chavan '197 at col. 28, l. 60 – col. 29, l. 11. "75. A method for connecting a plurality of port interfaces to a protocol fabric, comprising: providing a plurality of port interfaces for connection to a protocol fabric, the plurality of port interfaces being adjustable to be selectively compatible with FC or GbE; receiving messages from each of the plurality of port interfaces through a bidirectional crossbar module (BCM) having crossbar passageways compatible with both FC and GbE messages; connecting to an external DDR memory for storing and retrieving data messages in queues contained in DDR memory, and for sending and receiving data messages directly from the BCM; and controlling the connections between the port interfaces and the DDR memory interface through the BCM, and enabling direct memory access operations between the DDR memory interface and the BCM without data messages passing through the APEX." Chavan '197 at col. 29, l. 12 – col. 30, l. 14.
[13e] a reservation register implemented in hardware storing a value indicative of available space in said input queue, said reservation register accessible by both said first processing element and said at least one second processing element;	Chavan '197 teaches "a reservation register implemented in hardware storing a value indicative of available space in said input queue, said reservation register accessible by both said first processing element and said at least one second processing element," including at least the following exemplary citations. *See* above at row [1e].
[13f] computer storage storing instructions, which when executed by said at least one second processing element:	Chavan '197 teaches computer storage storing instructions that are executed by said at least one second processing element including at least the following exemplary citations. *See* above at row [1f].

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
[13g] receives a command from said first processing element;	Chavan ’197 teaches “receives a command from said first processing element,” including at least the following exemplary citations. *See* above at row [7g].
[13h] accesses said reservation register and reads said stored value;	Chavan ’197 teaches “accesses said reservation register and reads said stored value,” including at least the following exemplary citations. *See* above at row [1g].
[13i] determines when said read value indicates available space in said input queue to issue said command;	Chavan ’197 teaches “determines when said read value indicates available space in said input queue to issue said command,” including at least the following exemplary citations. *See* above at row [1h].
[13j] issues said command to said input queue on behalf of said first processing element; and	Chavan ’197 teaches “issues said command to said input queue on behalf of said first processing element,” including at least the following exemplary citations. *See* above at row [7j].
[13k] receives a response corresponding to said command from said output queue.	Chavan ’197 teaches “receives a response corresponding to said command from said output queue,” including at least the following exemplary citations. For example, Chavan ’197 teaches that the second processing element executes instructions that causes it to receive a response corresponding to said command from said output queue, by disclosing, as non–limiting examples, releasing the credit when the frame is processed. The scheduler receives a corresponding response from the output queue.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 11 Chavan ’197 at Figure 11. “INPs are used in computer network systems for providing HBA, router, and switch capabilities for particular protocol networks connected to a host through a host interface. An example of an INP 110 is shown in FIG. 1, which may be formed on a single silicon integrated circuit die packaged inside a ceramic, metal, or plastic housing to form a unitary component. INP 110 includes a shared memory crossbar module 112 which is controlled by a flow identification/scheduler module 114 through a bus 128. The flow identification/scheduler module 114 operates under the control of the Application Processor (APEX) module 116 acting through the bus 130, and maps the passages of crossbar locations inside the shared memory crossbar module 112. The APEX 116 in the example of FIG. 1 contains ten core processors for executing firmware, denoted as processors 118-127. The shared memory crossbar module 112 is connected to Protocol Coordination Stage (PCS) modules 132 and 134 through buses 136 and 138, respectively. PCS module 132 is connected to and controls the operation of Media Access Controller (MAC) modules 140, 142, 144, and 146. The MAC modules 140, 142, 144, and 146 are identical hardware interfaces capable of each communicating using the FC protocol at 1 Gigabits per second (Gbps) or 2 Gbps speeds. The speed and protocol settings for the for the MAC modules 140, 142, 144, and 146 are controlled by the PCS 132, which operates in response to control signals from a bus (not shown)

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	connecting the PCS 132 to the APEX 116. A parallel 10 Gbit (X Gbit) Medium-Independent Interface (XGMII) port 147 may be provided from the PCS 132, as determined by control signals originating from the APEX 116. The 10 Gb XGMII interface is not coupled to any of the four MAC modules." Chavan '197 at col. 1, ll. 23–53. "A Root Complex and EndPoint (RC/EP) module is connected to the BCM, and is connected to a SERDES for use with PCIe bus signals. A Direct Memory Access (DMA) engine and stream processor is connected to the BCM, and is also connected to a DDR SDRAM controller. The function of the controller is to control the sending and receiving of data to and from an external DDR memory module. A data pathway from the DMA engine and stream processor to the BCM allows DMA memory operations to occur directly with the controller and its attached DDR memory module, without data flow through the APEX, for data being exchanged with other sources. The controller is also connected to the scheduler, and creates queues in the external DDR memory for storing data messages, and also writes to and reads from such queues." Chavan '197 at col. 3, ll. 48–62. "The DMA engine and stream processor 386 is connected to the BCM 312 through bus 388, and is connected to the DDR SDRAM controller 390 through bus 392. The function of the controller 390 is to control the sending and receiving of data to and from an external DDR memory module (not shown) connected to the bus 394 of the controller 390. The controller 390 includes capabilities for providing RMW, statistics collection, XOR masking for data being transferred to or from the external DDR memory module. The data pathway through the bus 392 to the DMA engine and stream processor 386 and through the bus 388 to the BCM 312 allows DMA memory operations to occur directly with the controller 390 and its attached DDR memory module, without data flow through the APEX 316, for data being exchanged with other sources including the PCIe capabilities accessed through the bus 358 and RC/EP module 356, and the FC and GbE capabilities accessed through the buses 336 and 338 with the PCS modules 332 and 334. The ability to allow direct access for DMA operations through the BCM 312 with the DDR SDRAM controller 390 saves time that would be otherwise consumed by data being transferred into and out of the APEX 316. The controller 390 is also connected to the scheduler 314 through the bus 396. The controller 390 creates queues in the external DDR memory for storing data messages, and writes to and reads from such queues. The use of such queues, and the direct memory access to the crossbar BCM 312 (through the buses 388 and 392, controller 390, and DMA engine and stream processor 386) effectively increases the size of the crossbar BCM 312, and allows more switching pathways to be created than if the BCM 312 were used without the external DDR memory attached to connection 394." Chavan '197 at col. 8, l. 66 – col. 9, l. 29.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
14. The system of claim 13, wherein said input queue is a first-in-first-out (FIFO) buffer.	Chavan '197 teaches "[t]he system of claim 13, wherein said input queue is a first-in-first-out (FIFO) buffer," including at least the following exemplary citations. *See* above at row 2.
15. The system of claim 13, wherein said system is either part of a storage area network (SAN) controller.	Chavan '197 teaches "[t]he system of claim 13, wherein said system is either part of a storage area network (SAN) controller," including at least the following exemplary citations. *See* above at row 3.
16. The system of claim 13, wherein said system is part of a network attached storage (NAS) controller.	Chavan '197 teaches "[t]he system of claim 13, wherein said system is part of a network attached storage (NAS) controller," including at least the following exemplary citations. *See* above at row 4.
17. The system of claim 13, wherein said stored value is decremented by a predefined value prior to returning said stored value to said at least one second processing element.	Chavan '197 teaches "[t]he system of claim 13, wherein said stored value is decremented by a predefined value prior to returning said stored value to said at least one second processing element," including at least the following exemplary citations. *See* above at row 5.
18. The system of claim 13, wherein at least one third processing element executes said command and notifies said output queue to issue said response.	Chavan '197 teaches "[t]he system of claim 13, wherein at least one third processing element executes said command and notifies said output queue to issue said response," including at least the following exemplary citations. *See* above at row 6.