Exhibit 753-7
Invalidity of the ’753 Patent by Da Silva

The asserted claims of U.S. Patent No. 10,873,753 are anticipated and/or rendered obvious in view of U.S. Patent No. 9,148,389 (“Da Silva”) (avail. at PA035748-PA035778) alone or in combination with the knowledge of one of skill in the art or other prior art.

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

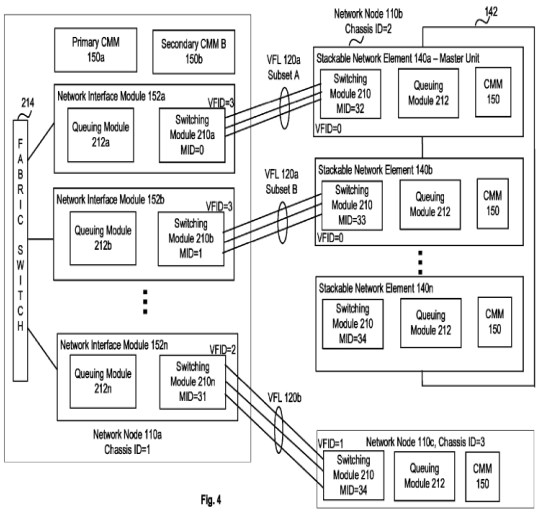
The claim chart below identifies where each limitation of each asserted claim of the ’753 patent is anticipated and/or rendered obvious in view of Da Silva. The citations provided below are exemplary, rather than exhaustive, and Defendants reserve the right to rely upon any other portion of the cited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

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
[1.pre] A platform for data flow processing comprising:	
[1.a] one or more swappable pods or cards in one or more chassis, coupled through a messaging interface network;	Da Silva discloses or renders obvious “one or more swappable pods or cards in one or more chassis, coupled through a messaging interface network,” as shown in at least the following exemplary citations: *See e.g.,* Da Silva, 7:51-67: The virtual chassis system 100 is also operable to include network nodes 110 with one or more different types of node architectures, such as single module, stackable, or multi-slot

IPR2025-00486
Concurrent Ventures LLC, XtreamEdge Inc. EX2042

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	chassis-based architectures. FIG. 4 illustrates a schematic block diagram of an embodiment of network nodes 110 in a virtual chassis system 100 with different types of node architectures. In this example, network node 110 a has a multi-slot chassis-based architecture with a plurality of network interface modules 152 a-n. In general, multi-slot chassis-based architectures share one enclosure, control manager modules (CMMs) 150 a-b and a common power supply with one or more network interface modules (NIMs) 152 a-n, such as line cards or port modules. The network interface modules 152 n include a queuing module 212 and switching module 210 and are connected by a fabric switch 214 integrated into the backplane of the chassis. *See e.g.,* Da Silva, 8:1-23: Network node 110 b in this example has a stackable node architecture and includes a plurality of network elements 140 a-n coupled by backplane connections 142. Each network element 140 a-n is operable as a stand-alone node and includes its own enclosure, control manager module (CMM) 150, switching module 210, queuing module 212 and power supply. In some stack architectures, one network element (Network Element 140 a in this example) is designated as the main or master unit of the stack for management purposes. *See e.g.,* Da Silva, 8:23-26: Network node 110 c has a single module node architecture, such as a single stackable element 140 or alternatively, a multi-slot chassis-based architecture with a single network interface module 152. *See e.g.,* Da Silva, FIG. 4:

IPR2025-00486
Concurrent Ventures LLC, XtreamEdge Inc. EX2042

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Fig. 4

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

[1.d] the plurality of swappable pods or cards being user-configurable to implement data flow processing architectures; and	
[1.e] a network coupled to the one or more swappable pods or cards and supporting messaging-based communication using packets each having a header with a chassis identifier, a board identifier, a module identifier, an instance identifier, and a type identifier, so that each type of module, each instance of a type of module, and each module on each board in each chassis can be addressed through the header.	Da Silva discloses or renders obvious "a network coupled to the one or more swappable pods or cards and supporting messaging-based communication using packets each having a header with a chassis identifier, a board identifier, a module identifier, an instance identifier, and a type identifier, so that each type of module, each instance of a type of module, and each module on each board in each chassis can be addressed through the header," as shown in at least the following exemplary citations: *See e.g.,* Da Silva, Abstract: A virtual chassis system includes a plurality of network nodes configured with a master virtual chassis address. The network nodes are connected by virtual fabric link (VFLs) that provide a connection for exchange of packets between the network nodes. The packets include source MAC addresses and associated hardware device information, such as source chassis ID, source network interface identifier and source port identifier information. The network nodes use this information to maintain synchronized MAC address tables for forwarding of packets in the virtual chassis system. *See e.g.,* Da Silva, 4:23-33: During normal operation, the network nodes 110 within the virtual-chassis system share the master MAC address for system identification by a wide variety of layer 2 and layer 3 protocols. For example, the spanning tree protocol and LACP protocols use the master MAC address as the identifier for the virtual chassis system 110. Internet Protocol (IP) routing also utilizes the master MAC address to identify the virtual chassis system 100 to external network elements in the network, e.g. peers use the master MAC address as the Ethernet destination address for packets destined to the virtual chassis system 100. As such, the network nodes 110 within the virtual chassis system 100 are viewed as a single logical

	node by external network nodes 112. In addition, the network nodes 110 within a virtual chassis system 100 are managed as a single node with a unified administration, operations and maintenance management system. *See e.g.,* Da Silva, 4:52-54: Within the virtual chassis system 100, a network node 110 is assigned a globally unique identifier called a chassis identifier or chassis ID. The network node 110 assigns an internal VFL identifier (VFID) to each of its configured VFLs 120 within the virtual chassis system 100. Since the VFID for a VFL is utilized for internal identification and configuration of VFLs 120, a network node 110 may assign the same or a different VFID to a VFL 120 as assigned by another network node 110. *See e.g.,* Da Silva, 5:13-18: During source learning, the network nodes 110 exchange packets with prepended headers over the VFLs 120 that include source MAC addresses and associated hardware device information, such as source chassis ID, source network interface identifier and source port identifier information. The network nodes 110 use this information to maintain synchronized MAC address tables with minimum messaging based MAC table synchronization. Utilizing the synchronized MAC address table, the network nodes 110 are operable to process and forward packets between the network nodes 110 in the virtual chassis system 100. *See e.g.,* Da Silva, 9:1-18: The network nodes 110 are assigned a unique chassis identifier in the virtual chassis system 100. The chassis ID for each network node 110 is unique and global and through the topology discovery, the network nodes 110 are aware of the chassis ID of its peer network nodes 110 in the virtual chassis system 100. In addition, unique hardware device identifiers or module identifiers (MIDs) for various components, such as the switching modules 210 and port interfaces in the network nodes 110, are generated allowing for management of local and remote objects. In an embodiment, the hardware device

	identifiers MIDs for the switching modules 210 have global significance within the virtual chassis system while MIDs for other components, such as queuing modules 212, may have only local significance. For example, the hardware device identifiers' assigned to the switching modules 210 are known by other network nodes 110 while hardware device identifiers for other devices are restricted to a local network node 110 and have no significance to other network nodes 110. *See e.g.,* Da Silva, 9:25-40: FIG. 5 illustrates a schematic block diagram of an embodiment of a network interface module (NIM) 152 operating in a prepended header mode in more detail. Though a network interface module 152 is illustrated, a stackable network element 140 or single module network element is operable to perform similar functions for operating in a prepended header mode. Switching module 210 includes a plurality of external ports 240 that are connected to external nodes 112 from the virtual chassis system 100. One or more of the external ports 240 may include member ports for a VC-LAG 114, LAG 116, single trunk or other trunk group, fixed link, etc. The external ports 240 may have the same physical interface type, such as copper ports (CAT-5E/CAT-6), multi-mode fiber ports (SX) or single-mode fiber ports (LX). In another embodiment, the external ports 240 may have one or more different physical interface types. *See e.g.,* Da Silva, 9:41-57: The external ports 240 are assigned external port interface identifiers (Port ID), e.g., device port values, such as gport and dport values, associated with the switching module 210. In an embodiment, the chassis ID of the network node 110, the MID of the switching module 210 and the external port interface identifier (Port ID) are used as a global unique identifier of a physical external port interface 240 in a network node 110 in the virtual chassis system 100. In another embodiment, globally unique module identifiers (MID) are assigned to the switching modules 210 of the network nodes in the virtual chassis system based on the chassis identifiers. For example, switching MIDs 0-31 are assigned to chassis ID=1, switching MIDs 32-63 are assigned to chassis ID=2, etc. In this case, the globally unique switching MIDs and external port identifiers (Port ID) are used as a global unique identifier

	of a physical external port interface 240 in a network node 110 in the virtual chassis system 100. *See e.g.,* Da Silva, 10:7-15: The hardware device information includes, for example, chassis ID, MID of a switching module 210 and/or port interface ID of a port 240 associated with the destination MAC address. The MAC/HDI forwarding table 250 may include one or more tables, such as source trunk map, trunk bitmap table, trunk group tables, VLAN mapping table, etc. In an embodiment, the MAC/HDI forwarding table 250 or parts thereof may be located in the queuing module of the NIM 152 or other module as well. *See e.g.,* Da Silva, 11:35-46: FIG. 6 illustrates a schematic block diagram of an embodiment of a pre-pended header of a packet in the virtual chassis system 100. The pre-pended header 300 includes fields for source HDI 302, destination HDI 304, VLAN ID 306, packet type 308, source MAC address 310 and destination MAC address 312. In an embodiment, the pre-pended header may also include, load balance identifier 314 and packet priority 316. The destination HDI 304 includes, for example, the port identifier (either device port (dport) and/or global port value (GPV)), MID of switching module 210 and chassis ID of the destination network node 110 associated with the destination MAC address. *See e.g.,* Da Silva, FIG. 4:

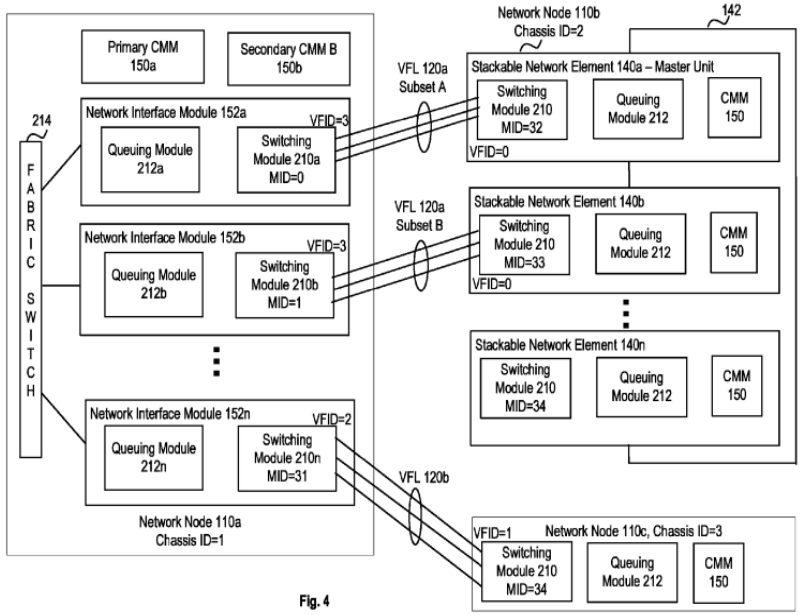


Fig. 4

See e.g., Da Silva, FIG.6:

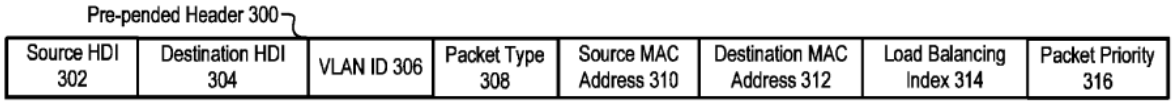
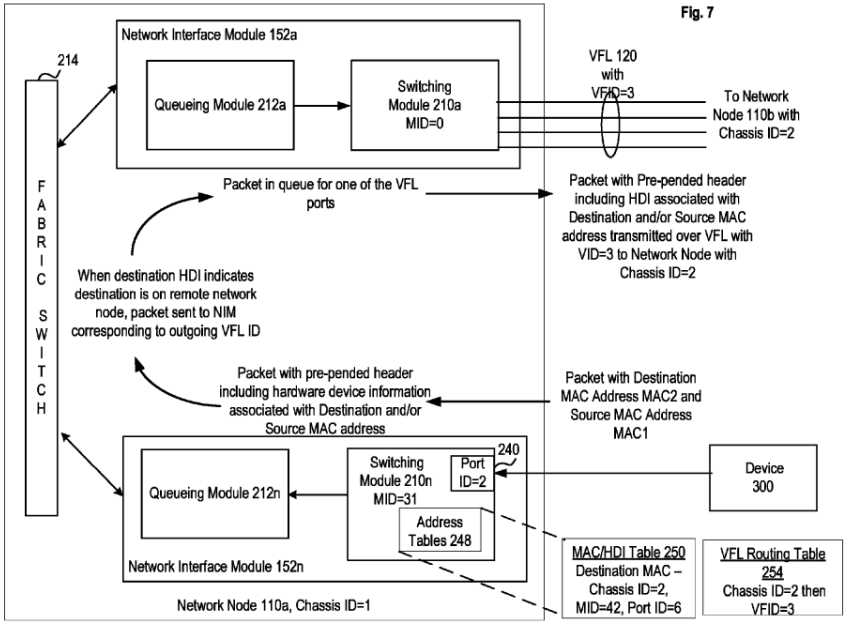


Fig. 6

See e.g., Da Silva, FIG.7:

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

3. The platform of claim 1, wherein the one or more swappable pods or cards comprises at least one processor card having a at least one processor allocatable for offloading hardware tasks to software, orchestrating data flows, orchestrating configurations of hardware modules, and executing in cooperation with a sandbox of one of the plurality of swappable pods or cards.	
4. The platform of claim 1, further comprising: at least one of the one or more swappable pods or cards having a plurality of types of memory and memory superposition with selectable arrangement of the plurality of types of memory as multilevel cache, interleaved memory, swapped memory over a specified address range, and	

memory allocated to specified address ranges according to parameterized task needs.	

[7.pre] A method for data flow processing in a platform, comprising:	*See* claim element 1.pre.
[7.a]. configuring a plurality of swappable pods or cards in one or more chassis, coupled through a messaging interface network, to implement a data flow processing architecture;	*See* claim elements 1.pre and 1.a.
[7.b] configuring user-definable hardware modules or user-definable software modules in each of one or more of the plurality of swappable pods or cards;	*See* claim element 1.c.
[7.c] communicating a data flow in accordance with the implemented data flow processing architecture among one	*See* claim elements 1.c and 1.d.

or more hardware modules or one or more software modules in each of the plurality of swappable pods or cards, including the user-definable hardware modules or user-definable software modules; and	
[7.d] selecting a memory superposition in at least one of the plurality of swappable pods or cards having a plurality of types of memory with selectable arrangement of the plurality of types of memory as multilevel cache, interleaved memory, swapped memory over a specified address range, and memory allocated to specified address ranges according to parameterized task needs.	*See* claim 4.
8. The method of claim 7, wherein the	*See* claim element 1.e.

communicating the data flow comprises: communicating the data flow through the messaging interface network, comprising a network coupled to the plurality of swappable pods or cards and supporting messaging-based communication using packets each having a header with a chassis identifier, a board identifier, a module identifier, an instance identifier, and a type identifier, so that each type of module, each instance of a type of module, and each module on each board in each chassis can be addressed through the header.	
9. The method of claim 7, wherein the communicating the data flow comprises: bypassing the messaging interface network and	*See* claim 2.

communicating through a direct connection among two or more pods, cards or components.	
10. The method of claim 7, further comprising: allocating at least one processor of a processor card, from among the plurality of swappable pods or cards, to offloading hardware tasks to software, orchestrating data flows, orchestrating configurations of hardware modules, or executing in cooperation with a sandbox of one of the plurality of swappable pods or cards.	*See* claim 3.
[13.pre] A tangible, non-transitory, computer-readable media having instructions thereupon which, when executed by a processor, cause the	*See* claim elements 1.pre and 1.a-1.e.

processor to perform a method comprising:	
[13.a] configuring a plurality of swappable pods or cards in one or more chassis, coupled through a messaging interface network, to implement a data flow processing architecture;	*See* claim elements 1.a-1.c.
[13.b] configuring user-definable hardware modules or user-definable software modules in each of one or more of the plurality of swappable pods or cards;	*See* claim element 1.c.
[13.c] communicating a data flow in accordance with the implemented data flow processing architecture among one or more hardware modules or one or more software modules in each of the plurality of swappable pods or cards, including the user-definable hardware modules or user-	*See* claim elements 1.c and 1.d.

definable software modules; and	
[13.d] communicating the data flow through the messaging interface network, comprising a network coupled to the plurality of swappable pods or cards and supporting messaging-based communication using packets each having a header with a chassis identifier, a board identifier, a module identifier, an instance identifier, and a type identifier, so that each type of module, each instance of a type of module, and each module on each board in each chassis can be addressed through the header.	*See* claim element 1.e.
14. The computer-readable media of claim 13, wherein the method further comprises: allocating at least one processor of a processor	*See* claim 3.

card having a plurality of processors, from among the plurality of swappable pods or cards, to offloading hardware tasks to software, orchestrating data flows, orchestrating configurations of hardware modules, or executing in cooperation with a sandbox of one of the plurality of swappable pods or cards.	
15. The computer-readable media of claim 13, wherein the method further comprises: selecting a memory superposition in at least one of the plurality of swappable pods or cards having a plurality of types of memory with selectable arrangement of the plurality of types of memory as multilevel cache, interleaved memory, swapped memory over a specified address range, and	*See* claim 4.

memory allocated to specified address ranges according to parameterized task needs.	