Exhibit 753-10
Invalidity of the ’753 Patent by Saulsbury

The asserted claims of U.S. Patent No. 10,873,753 are anticipated and/or rendered obvious in view of U.S. Publication No. 2002/0087821 (“Saulsbury”) (avail. at PA035375-PA035390) alone or in combination with the knowledge of one of skill in the art or other prior art.

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

The claim chart below identifies where each limitation of each asserted claims of the ’753 patent is anticipated and/or rendered obvious in view of Saulsbury. 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;	

IPR2025-00478
Concurrent Ventures LLC, XtreamEdge Inc. EX2044

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
[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.	

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	Saulsbury discloses or renders obvious “at least one of the one or more swappable pods or cards having a plurality of types of memory and memory superposition with selectable arrangement of the plurality of types of memory as multilevel cache, interleaved memory, swapped memory over a specified address range, and memory allocated to specified address ranges according to parameterized task needs,” as shown in at least the following exemplary citations: *See e.g.,* Saulsbury, Abstract: According to the invention, a first processor chip (10) comprising a processing core (12) and at least one bank of memory (14). The at least one bank of memory (14) preferably includes a mode control input (32) for controlling the mode of the at least one bank of memory (14) between physical memory and cache memory. In addition, the first processor chip (10) may further comprise an I/O link (26) configured to facilitate communication between the first processor chip

memory over a specified address range, and memory allocated to specified address ranges according to parameterized task needs.	(10) and other processor chips, and a communication and memory controller (20, 22) in electrical communication with the processing core (12), the at least one bank of memory (14), and the I/O link (26). The communication and memory controller (20, 22) preferably controls the exchange of data between the first processor chip (10) and the other processor chips, as well as receive memory requests from the processing core (12) of the first processor chip (10) and from other processing cores residing on the other processor chips, and process the memory requests with the at least one bank of memory (14). The memory requests from the other processing cores on the other processor chips preferably are received by the first processor chip (10) through the I/O link (26). *See e.g.,* Saulsbury, [0003]: The number of transistors that can be placed on a single integrated circuit chip is approaching one billion. Therefore, in light of processing latencies that occur from accessing memory on separate memory chips, it is desirable to fabricate both the central processing functionality and memory on a single chip. The amount of memory that can be fabricated on a single chip typically is sufficient for low performance applications. However, high performance, memory intensive applications, for example, multi-media applications, typically will require additional memory, such as memory external from the processor chip. Thus, for low performance applications it is desirable to have the memory fabricated on the processor chip to act as simple, physical memory, and for high performance applications, it is desirable to have the memory fabricated on the processor chip to act as cache memory, and more particularly, second-level cache memory. Therefore, what is needed is a processor and memory chip which may be configured to switch the memory mode between physical memory and cache memory, depending on the particular application of the processor and memory chip. *See e.g.,* Saulsbury, [0005]: According to one embodiment of the invention, a first processor chip comprising a processing core and at least one bank of memory. The at least one bank of memory preferably includes a mode control input for controlling the mode of the at least one bank of memory between physical memory and cache memory. *See e.g.,* Saulsbury, [0006]:

	In addition, the first processor chip may further comprise an I/O link configured to facilitate communication between the first processor chip and other processor chips, and a communication and memory controller in electrical communication with the processing core, the at least one bank of memory, and the I/O link. One function of the communication and memory controller is to control the exchange of data between the first processor chip and the other processor chips. In accordance with one embodiment of the invention, the other processor chips are similar to processor chip described in this patent application. In addition, the communication and memory controller is configured to receive memory requests from the processing core of the first processor chip and from other processing cores residing on the other processor chips, and process the memory requests with the at least one bank of memory. The memory requests from the other processing cores on the other processor chips are received by the first processor chip through the I/O link. *See e.g.,* Saulsbury, [0008]: According to another embodiment, the present invention may comprise, in a computer system, a scalable computer processing architecture including one or more processor chips each having a processing core and at least one bank of memory. The at least one bank of memory on each of the processor chips include a mode control input for controlling the mode of the at least one bank of memory between physical memory and cache memory. In addition, each processor chip may further comprise an I/O link configured to facilitate communication between the processor chips, and a communication and memory controller in electrical communication with the processing core, the at least one bank of memory, and the I/O link. The communication and memory controller controls the exchange of data between the processor chips, as well as receives memory requests from the processing cores of the processor chips and processes them with the at least one bank of memory. *See e.g.,* Saulsbury, [0017]: With reference to FIG. 1, one embodiment of a processor chip 10 in accordance with the present invention is shown. In particular, processor chip 10 comprises a processing core 12, a plurality of

	memory banks 14, a memory controller 20, a distributed shared memory controller 22, an external memory interface 24, a high-speed I/O link 26, a boot interface 28, and a diagnostic interface 30. *See e.g.,* Saulsbury, [0020]: Connected between processing core 12 and memory 14 is a memory controller 20. Memory controller 20 communicates with processing core 12 and memory 14, and as discussed in more detail below, handles the memory I/O requests to memory 14 from processing core 12 and from other processors and I/O devices. Connected to memory controller 20 is a distributed shared memory (DSM) controller 22, which controls and routes I/O requests and data messages from processing core 12 to off-chip devices, such as other processor chips and/or I/O peripheral devices. In addition, as discussed in more detail below, DSM controller 22 may be configured to receive I/O requests and data messages from off-chip devices, and route the requests and messages to memory controller 20 for access to memory 14 or processing core 12. In addition, while FIG. 1 shows memory controller 20 and DSM controller 22 as two separate units, one skilled in the art will appreciate that memory controller 20 and DSM controller 22 can be configured as one unit. That is, one controller can be configured to process the control functions of both memory controller 20 and DSM controller 22. Thus, the present invention is not limited to the illustrated embodiment. *See e.g.,* Saulsbury, [0022]: Processor chip 10 further comprises an external memory interface 24. As discussed in greater detail below, external memory interface 24 is connected to memory controller 20 and is configured to communicate memory I/O requests from memory controller 20 to external memory. Finally, as mentioned briefly above, processor chip 10 further comprises a boot interface 28 and a diagnostic interface 30. Boot interface 28 is connected to processing core 12 and is configured to receive a bootstrap program for cold booting processing core 12 when needed. Similarly, diagnostic interface 30 also is connected to processing core 12 and configured to provide external access to the processing core for diagnostic purposes. *See e.g.,* Saulsbury, [0024]:

	As mentioned briefly above, processing core 12 comprises a scalable VLIW processing core, which may be configured as a single processing pipeline or as multiple processing pipelines. In addition, each processing pipeline may comprise one or more processing paths for processing instructions. Thus, a single processing pipeline can function as a single pipeline with a single processing path for processing one instruction at a time, as a single pipeline having multiple processing paths for processing multiple instructions independently, or as a single VLIW pipeline having multiple processing paths for processing multiple sub-instructions in a single VLIW instruction word. Similarly, a multi-pipeline processing core can function as multiple autonomous processing cores of as one or more synchronized VLIW processing cores. This enables an operating system to dynamically choose between a synchronized VLIW operation or a parallel multi-thread or multi-strand paradigm. In accordance with one embodiment of the invention, processing core 12 may comprise any number of pipelines and each of the pipelines may comprise any number of processing paths. For example, the processing core may comprise X-number of pipelines, each having Y-number of processing paths, such that the total number of processing paths is X * Y. *See e.g.,* Saulsbury, [0025]: In accordance with one embodiment of the present invention, when processing core 12 is operating in the synchronized VLIW operation mode, an application program compiler typically creates a VLIW instruction word comprising a plurality of sub-instructions appended together, which are then processed in parallel by processing core 12. The number of sub-instructions in the VLIW instruction word matches the total number of available processing paths in the one or more processing core pipelines. Thus, each processing path processes VLIW sub-instructions so that all the sub-instructions are processed in parallel. In accordance with this particular aspect of the present invention, the sub-instructions in a VLIW instruction word issue together. Thus, if one of the processing paths is stalled, all the sub-instructions will stall until all of the processing paths clear. Then, all the sub-instructions in the VLIW instruction word will issue at the same time. As one skilled in the art will appreciate, even though the sub-instructions issue simultaneously, the processing of each sub-instruction may complete at different times or clock cycles, because different instruction types may have different processing latencies. *See e.g.,* Saulsbury, [0026]:

	In accordance with an alternative embodiment of the present invention, when the multi-pathed/multi-pipelined processing core is operating in the parallel multi-thread/multi-strand mode, the program instructions are not necessarily tied together in a VLIW instruction word. Thus, as instructions are retrieved from an instruction cache, the operating system determines which pipeline is to process a particular instruction stream. Thus, with this particular configuration, each pipeline can act as an independent processor, processing instructions independent of instructions in the other pipelines. In addition, in accordance with one embodiment of the present invention, by using the multi-threaded mode, the same program instructions can be processed simultaneously by two separate pipelines using two separate blocks of data, thus achieving a fault tolerant processing core. The remainder of the discussion herein will be directed to a synchronized VLIW operation mode. However, the present invention is not limited to this particular configuration. *See e.g.,* Saulsbury, [0067]: In accordance with one embodiment of the present invention, memory 14 comprises DRAM memory and can be configured as either cache memory with associated tags or as directly accessible physical memory. Alternatively, memory 14 can be a combination of both cache and physical memory. *See e.g.,* Saulsbury, [0068]: In accordance with another embodiment of the present invention, memory 14 includes a control input bit 32 (FIG. 1) which controls the mode of memory 14. For example, when control bit 32 is in one state, memory 14 operates as cache memory, and when control bit 32 is in another state, memory 14 operates as physical memory. In the embodiment illustrated in FIG. 1, each memory bank 14 includes its own control bit 32. However, in an alternative embodiment of the present invention, one control bit 32 may be configured to control the operation of all memory banks 14. *See e.g.,* Saulsbury, [0069]: Memory controller 20 (FIG. 1) is configured to receive memory I/O and synchronization requests from processing core 12 and DSM controller 22, and pass the requests to the on-chip memory 14 or to external memory through external memory interface 24. Memory controller 24 ensures that

	memory access requests from processing core 12 to on-chip memory 14 or to external memory are consistent with the state of the memory being accessed. In the case of a memory inconsistency, DSM controller 22 may be invoked to resolve the inconsistency. In accordance with this aspect of the invention, DSM controller 22 either changes the state of the memory being accessed, or causes an exception for processing core 12. *See e.g.,* Saulsbury, [0070]: In accordance with another embodiment of the present invention, memory controller 20 may be configured to handle certain implementation dependent load and store operations with on-chip memory 14, external memory, or memory residing on the other processor chips. For example, memory controller 20 may be configured to control endianness, caching and prefetching operations for certain program or operating systems implementations. *See e.g.,* Saulsbury, [0074]: While one embodiment of DSM controller 22 is disclosed herein as having two communication engines (e.g., local and remote), one skilled in the art will appreciate that other configurations for DSM controller 22 may be used. For example, DSM controller 22 may be configured with only one communication engine, or alternatively, several engines may be used. In addition, DSM controller 22 may be configured with a plurality of routing tables or routing instructions for controlling message and I/O routing, and DSM controller 22 may be programmed to handle auto-routing functions. Finally, even though memory controller 20 and DSM controller 22 are illustrated in FIG. 1 and described herein as separate devices, one skilled in the art will appreciate that memory controller 20 and DSM controller 22 can be configured as a single device which handles the functions of both devices. Thus, the configurations and operations of memory controller 20 and DSM controller 22 are not limited to the disclosure herein. *See e.g.,* Saulsbury, [0077]: External memory interface 24 (FIG. 1) comprises a read/write interface to external memory. The external memory can be any desired type, e.g. volatile, non-volatile, etc. External memory interface 24 is an expansion port on processor chip 10, which allows memory in addition to the on-chip

memory to be connected to the chip. As discussed above, access and use of the external memory via external memory interface 24 is dictated by memory controller 20 and DSM controller 22. That is, memory controller 20 directs the memory I/O requests across external memory interface 24 to the external memory.

See e.g., Saulsbury, FIG. 1:

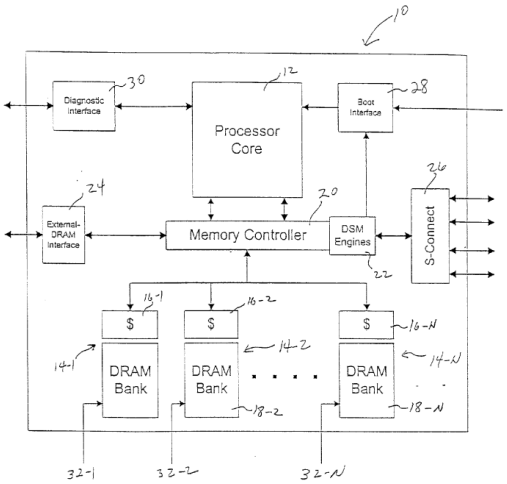
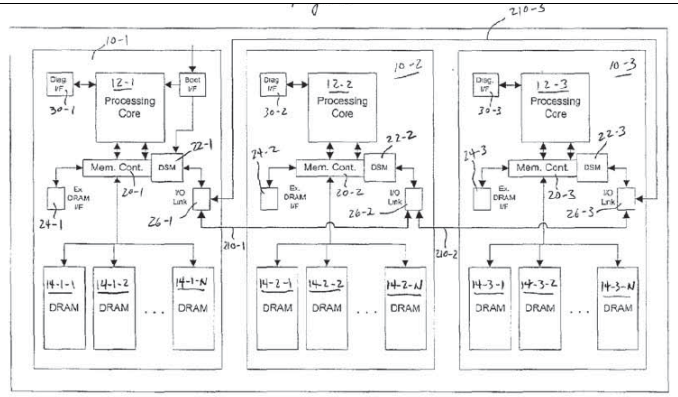


Fig. 1

See e.g., Saulsbury, FIG. 5:

	 Fig. 5

[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	*See* claim element 1.c.

of swappable pods or cards;	
[7.c] communicating a data flow in accordance with the implemented data flow processing architecture among one or more hardware modules or one or more software modules in each of the plurality of swappable pods or cards, including the user-definable hardware modules or user-definable software modules; and	*See* claim elements 1.c and 1.d.
[7.d] selecting a memory superposition in at least one of the plurality of swappable pods or cards having a plurality of types of memory with selectable arrangement of the plurality of types of memory as multilevel cache, interleaved memory, swapped memory over a specified address range, and memory allocated to	*See* claim 4.

specified address ranges according to parameterized task needs.	
8. The method of claim 7, wherein the communicating the data flow comprises: communicating the data flow through the messaging interface network, comprising a network coupled to the plurality of swappable pods or cards and supporting messaging-based communication using packets each having a header with a chassis identifier, a board identifier, a module identifier, an instance identifier, and a type identifier, so that each type of module, each instance of a type of module, and each module on each board in each chassis can be addressed through the header.	*See* claim element 1.e.

9. The method of claim 7, wherein the communicating the data flow comprises: bypassing the messaging interface network and communicating through a direct connection among two or more pods, cards or components.	*See* claim 2.
10. The method of claim 7, further comprising: allocating at least one processor of a processor card, from among the plurality of swappable pods or cards, to offloading hardware tasks to software, orchestrating data flows, orchestrating configurations of hardware modules, or executing in cooperation 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 processor to perform a method comprising:	*See* claim elements 1.pre and 1.a-1.e.
[13.a] configuring a plurality of swappable pods or cards in one or more chassis, coupled through a messaging interface network, to implement a data flow processing architecture;	*See* claim elements 1.a-1.c.
[13.b] configuring user-definable hardware modules or user-definable software modules in each of one or more of the plurality of swappable pods or cards;	*See* claim element 1.c.
[13.c] communicating a data flow in accordance with the implemented data flow processing architecture among one or more hardware modules or one or more	*See* claim elements 1.c and 1.d.

software modules in each of the plurality of swappable pods or cards, including the user-definable hardware modules or user-definable software modules; and	
[13.d] communicating the data flow through the messaging interface network, comprising a network coupled to the plurality of swappable pods or cards and supporting messaging-based communication using packets each having a header with a chassis identifier, a board identifier, a module identifier, an instance identifier, and a type identifier, so that each type of module, each instance of a type of module, and each module on each board in each chassis can be addressed through the header.	*See* claim elements 1.e.

14. The computer-readable media of claim 13, wherein the method further comprises: allocating at least one processor of a processor card having a plurality of processors, from among the plurality of swappable pods or cards, to offloading hardware tasks to software, orchestrating data flows, orchestrating configurations of hardware modules, or executing in cooperation with a sandbox of one of the plurality of swappable pods or cards.	*See* claim 3.
15. The computer-readable media of claim 13, wherein the method further comprises: selecting a memory superposition in at least one of the plurality of swappable pods or cards having a plurality of types of memory with	*See* claim 4.

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.	