Exhibit 596-04

Invalidity of the ’596 Patent by Dongare ’439 and Other Prior Art

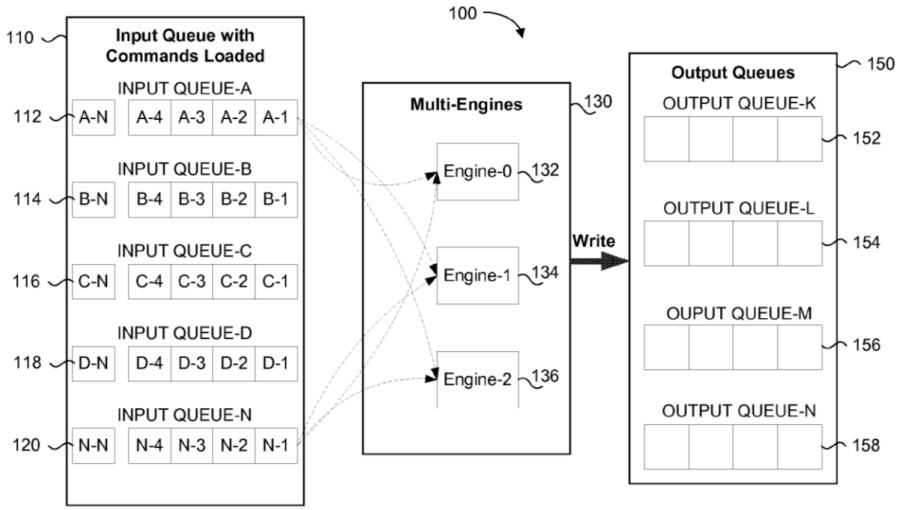
The asserted claims of U.S. Patent No. 8,924,596 are anticipated and/or rendered obvious in view of US8559439B1 (“Dongare ’439”) alone or in combination with knowledge of one of skill in the art or other prior art.

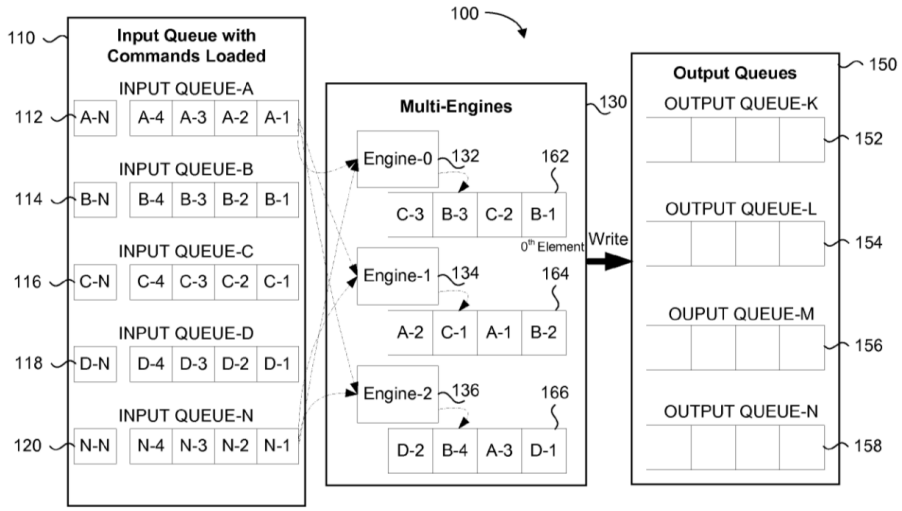
This claim chart is based on Defendants’ present understanding of the asserted claims, Plaintiffs’ vague and overbroad infringement contentions (and corresponding, implicit claim constructions), and Defendants’ investigation to date. Defendants are not adopting Plaintiffs’ apparent constructions, nor 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.

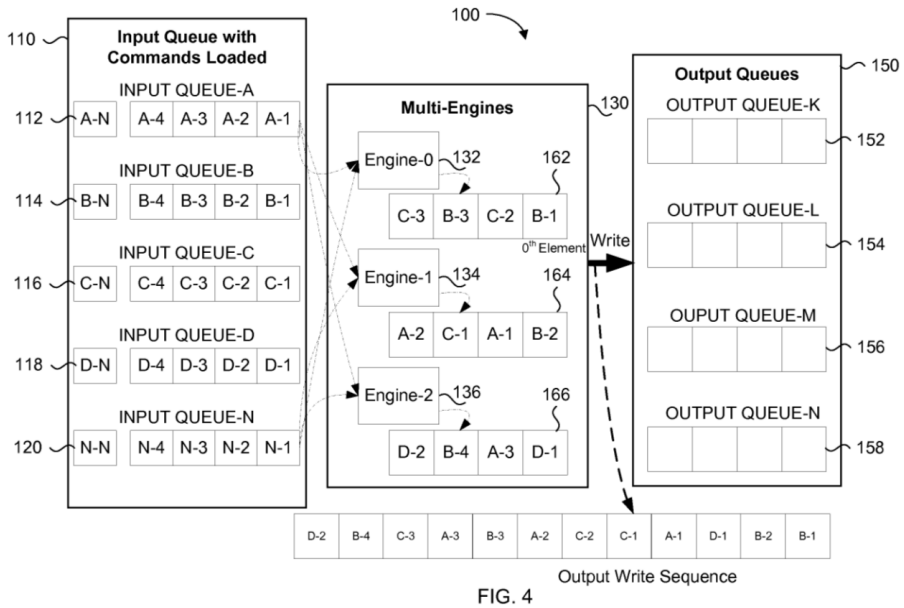
The claim chart below identifies where each limitation of each asserted claim of the ’596 patent can be found in Dongare ’439. This analysis is based on Plaintiff’s interpretation of the claims as indicated in their Infringement Contentions. The citations provided below are exemplary, rather than exhaustive, and Defendants reserve the right to rely upon any other portion of the cited references.

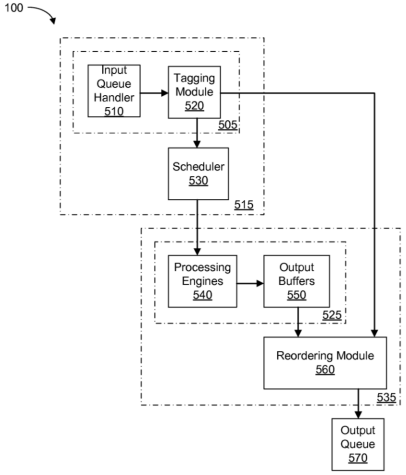
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
[1pre] A system for dividing and synchronizing a processing task across a plurality of processing elements comprising:	To the extent the preamble is considered a claim limitation, Dongare ’439 teaches “[a] system for dividing and synchronizing a processing task across a plurality of processing elements,” including at least the following exemplary citations. For example, Dongare ’439 teaches a system for dividing and synchronizing processing tasks by disclosing, as non–limiting examples, a multi-engine processing system for use in a RAID controller that divides and synchronizes RAID commands. *See, e.g.:* “A method and apparatus for queue-ordering commands in multi-engines, multi-queues and/or multi-flows environment is provided. Commands from single/multiple queues and multi-flows are processed by multi-engines with different processing time and/or out of order, which breaks sequential order of commands from same input queue and commands are distributed across multiple engines' output buffer after processing. Processed commands are stored in dedicated command output buffer associated with each engine temporarily. The processed commands are re-ordered while writing out. Also commands can be scheduled to idle engines to achieve maximum throughput, thus utilizing the engines in an optimal manner.” Dongare ’439 at Abstract.

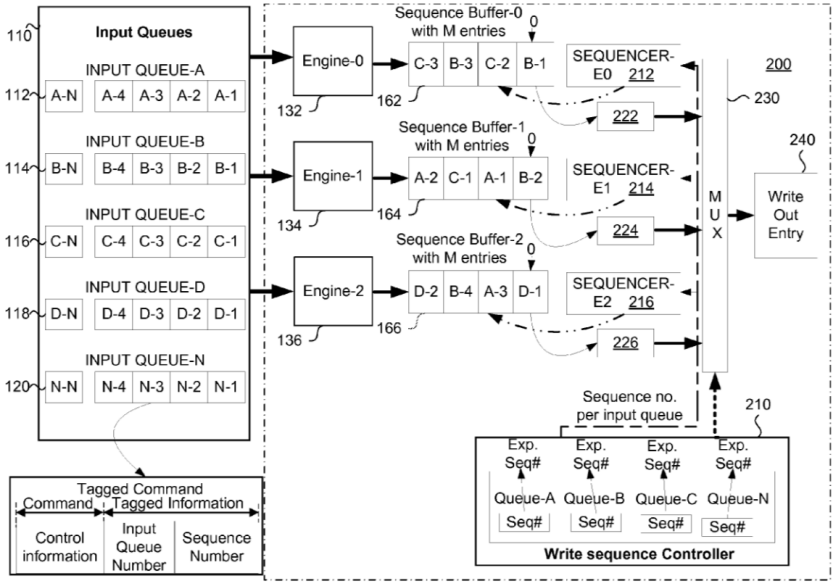
IPR2025-00478
Concurrent Ventures LLC, XtreamEdge Inc. EX2027

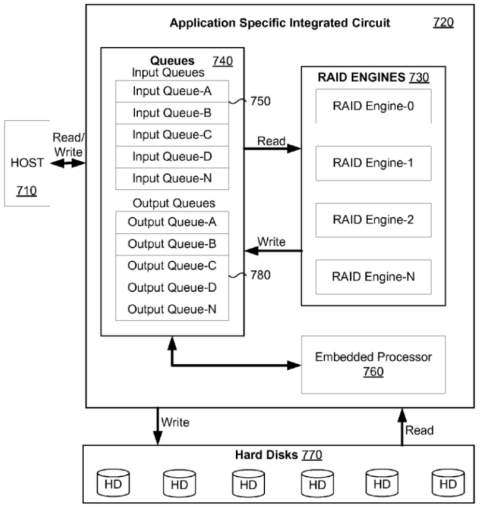
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 1 Dongare ’439 at Figure 1.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 2 Dongare ’439 at Figure 2.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 4 Dongare ’439 at Figure 4.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 5 Dongare ’439 at Figure 5.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 6 Dongare ’439 at Figure 6.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 7 Dongare ’439 at Figure 7. “Multiple engines or processors are used to increase the throughput of a device or a network system. Typically, segments of one or more input streams are parallel processed in multiple processors to increase the processing efficiency or throughput of the system. However, out of order processing of the individual segments may result in a loss of order in the sequence of the segments within the input stream and the resulting output stream may not correspond to the input stream. Additional processing is then required to restore the order of the segments of the output stream relative to the input stream resulting in a compromise of the efficiencies gained by multi-processing.” Dongare ’439 at col. 1, ll. 21–32. “In a first aspect, the present disclosure provides a method for queue ordering in a multi-engine processing system having a plurality of processing engines to process a plurality of input streams and to provide a corresponding plurality of output streams. The method comprises: receiving the plurality of input streams, each input stream having a plurality of input segments, the plurality of input segments ordered relative to other input segments within the input stream; appending an identity tag to the plurality of input segments in each input stream to produce a plurality of tagged input segments, the identity tag including information indicating the input stream and the relative order within the input stream of each of the plurality of input segments; iteratively assigning each of the plurality of tagged input segments to any one of the plurality of

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	processing engines based on availability of the processing engines in the relative order of the tagged input segments within the respective input stream; processing each of the plurality of tagged input segments in its respective assigned processing engine to produce a corresponding plurality of tagged output segments; storing each of the plurality of tagged output segments in a dedicated output buffer associated with its respective assigned processing engine; and, reordering the plurality of tagged output segments to produce the plurality of output streams, each output stream having the same relative order as in the corresponding input stream." Dongare '439 at col. 1, ll. 36–60. "In another aspect, the present disclosure provides a multi-engine processing system comprising: an input queue handler configured to receive a plurality of input streams, each input stream having a plurality of input segments, the plurality of input segments ordered relative to other input segments within the input stream; a tagging module configured to append an identity tag to a plurality of input segments in each input stream to produce a plurality of tagged input segments, the identity tag including information indicating the input stream and the relative order within the input stream of each of the plurality of input segments; a scheduler configured to iteratively assign each of the plurality of tagged input segments to any one of a plurality of processing engines based on availability of the processing engines in the relative order of the tagged input segments within the respective input stream, each of the plurality of tagged input segments being processed in its respective assigned processing engine to produce a corresponding plurality of tagged output segments; a plurality of dedicated output buffers associated with the plurality of processing engines, the plurality of dedicated output buffers configured to store each of the plurality of tagged output segments; and, a reordering module configured to reorder the plurality of tagged output segments to produce a plurality of output streams, each output stream having the same relative order of output segments as in the corresponding input stream." Dongare '439 at col. 1, l. 61 – col. 2, l. 19. "Several methods have been proposed to reorder output streams resulting from out-of-order multi-processing. U.S. Pat. No. 7,072,342 describes a system and method of reordering out-of-order packets. In this method, while packets may be processed out of order, the processed packets or commands are written into command FIFO buffers of a command queue only when a current sequence number of the command FIFO buffer matches the packet sequence number of the processed packet. Thus, command FIFO buffer holds ordered commands when read out on a FIFO basis. When there are multiple command buffers, each processed command is associated with a lane number and a sequence number and each command FIFO buffer is dedicated to each input lane. Processed commands are written to a particular command FIFO buffer only when its lane number and sequence number matches with lane and sequence number of the processed command. Consequently, a task has to wait until the sequence number of the command matches the sequence

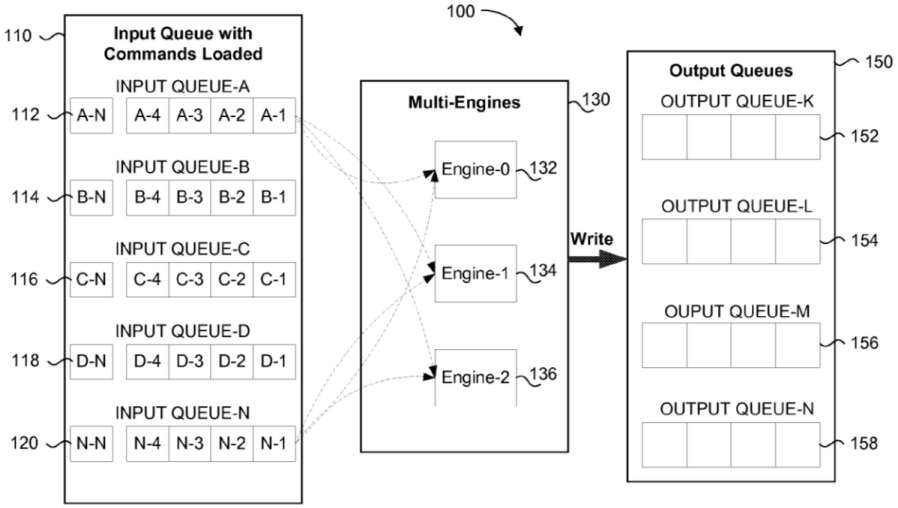
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	number of the command FIFO buffer, and in effect, one or more tasks remain idle. In addition, the number of command FIFO buffers should be equal to number of input command lanes rendering the system inflexible and rigid." Dongare '439 at col. 2, l. 58 – col. 3, l. 12. "Generally, the present disclosure provides a method and apparatus for queue ordering in a multi-processing system. The method and apparatus can be implemented in multi-engine, multi-queue, and/or multi-flow environments, such as in multi-engine processor systems or in a network domain. In a multi-engine processor or a multi-threading system, tasks from same program can be executed concomitantly on different processors to achieve maximum throughput. However, the completed task has to be provided in the order it was posted to the multi-processor system. The method and apparatus of the present disclosure provide for reordering completed tasks efficiently." Dongare '439 at col. 4, ll. 44–55. "FIG. 1 illustrates a multi-engine processing system 100 capable of handling a plurality of input queues or input streams 110. The input queues are sequentially ordered commands or segments to be processed by the plurality of engines 130. The processed commands are written out to the plurality of output queues 150. In the example shown in FIG. 1, the plurality of input queues are received and stored in input queue buffer-A 112, input queue buffer-B 114, input queue buffer-C 116, input queue buffer-D 118, and input queue buffer-N 120. The plurality of processing engines 130 includes Engine-0132, Engine-1134, and Engine-2136. The plurality of output queues 150 are written out and stored in output queue buffer-K 152, output queue buffer-L 154, output queue buffer-M 156, and output queue buffer-N 158. Thus, command streams A, B, C, D through N can be received and stored in the respective input queue buffers and the processed commands can be written out and stored in output queue buffers. As shown in FIG. 1, the input queue buffer-A 112 has commands A-1, A-2, . . . A-N, input queue buffer-B 114 has commands B-1, B-2, . . . B-N, input queue buffer-C 116 has commands C-1, C-2, . . . C-N, input queue buffer-D 118 has commands D-1, D-2, . . . D-N, and input queue buffer-N 120 has commands N-1, N-2, . . . N-N." Dongare '439 at col. 5, ll. 3–26. "In the example embodiment shown in FIG. 2, the three engines are identical and are capable of processing any command assigned to them. However, depending on the nature of a command being processed, for example, the byte count and functionality of the command, a command provided earlier to one engine may take more time to process than another command assigned to other engine. Thus, processed commands would be ready to write out at different time points. Also, the sequence of commands from the same input queue would be disturbed at the output side and may be distributed across the dedicated buffers of the respective engines. As shown in FIG. 2, each engine has a dedicated output buffer. For example, Engine-0132 has a

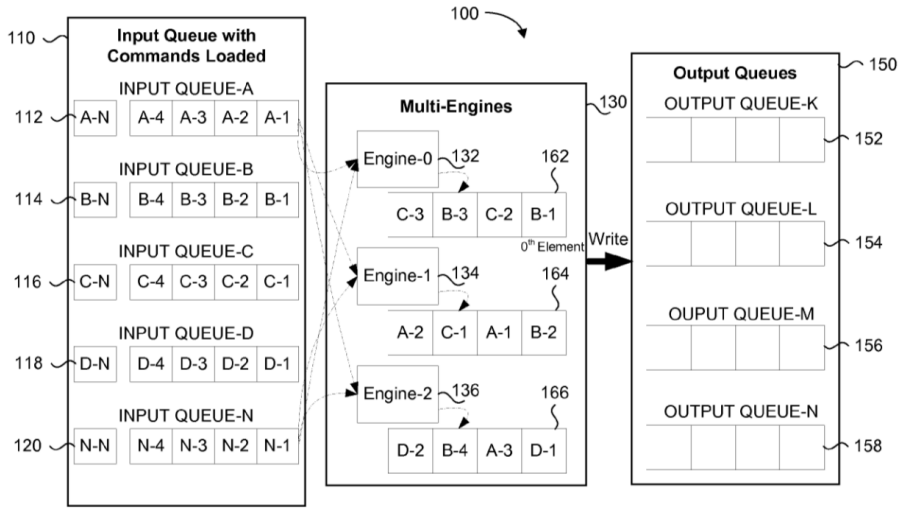
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	dedicated output buffer 162, Engine-1134 has a dedicated output buffer 164, and Engine-2136 has a dedicated output buffer 166." Dongare '439 at col. 6, ll. 28–42. "In an example embodiment, the method further comprises storing the plurality of tagged input segments in a shared buffer for pre-processing or filtering prior to assigning each of the plurality of tagged input segments to any one of the plurality of processing engines." Dongare '439 at col. 8, ll. 41–45. "FIG. 4 illustrates the required queue ordering while writing out the processed commands to the output queues, each output queue having the same relative sequence as in the corresponding input queue. As described earlier, the plurality of output queues 150 are written out and stored in output queue buffer-K 152, output queue buffer-L 154, output queue buffer-M 156, and output queue buffer-N 158. The various command input queues A-N are iteratively scheduled or assigned to be processed on any one of the plurality of processing engines based on the availability of the processing engines in the relative order if the commands within their respective command input queue as discussed earlier with respect to the example embodiment of FIG. 2." Dongare '439 at col. 8, ll. 51–63. "In order to maintain the original order or sequence of the commands within a given command input queue, the processed commands must be written out to the command output queues in the same sequence as it was read out from the respective command input queue. It follows that A-1 should be written before A-2, A-2 should be written before A-3 and so on. As shown in FIG. 4, commands are written in the same relative order as in the command input queue: B-1, B-2, D-1, A-1, C-1, C-2, A-2, B-3, A-3, C-3, B-4, and D-2, i.e., the relative order of commands within a command output queue is maintained in the write out sequence irrespective of the processing engine that processed each command. In the example embodiment of FIG. 4, processed commands from command input queue A are written out as: A-1, A-2 and A-3, wherein commands A-1 and A-2 commands were processed by Engine-1 and command A-3 was processed by Engine-2. In the example embodiment, command A-3 was processed earlier to command A-2 but it was not written out since command A-2 was not processed. Processed command A-3 was stored in the dedicated output buffer 166 of Engine-2136 until command A-2 was processed and ready for write out. Similarly, processed commands from command input queue B are written out as: B-1, B-2, B-3 and B-4, wherein commands B-1 and B-3 were processed by Engine-0, command B-2 was processed by Engine-1 and command B4 was processed by Engine-2. In the example embodiment commands B-1 and B-2 were processed at the same time by different engines and hence could be written out sequentially. However, command B-4 was processed earlier by Engine-2 but it was not written out as command B-3 was not processed. Processed command B-4 was stored in dedicated output buffer 166 of Engine-2136 until command

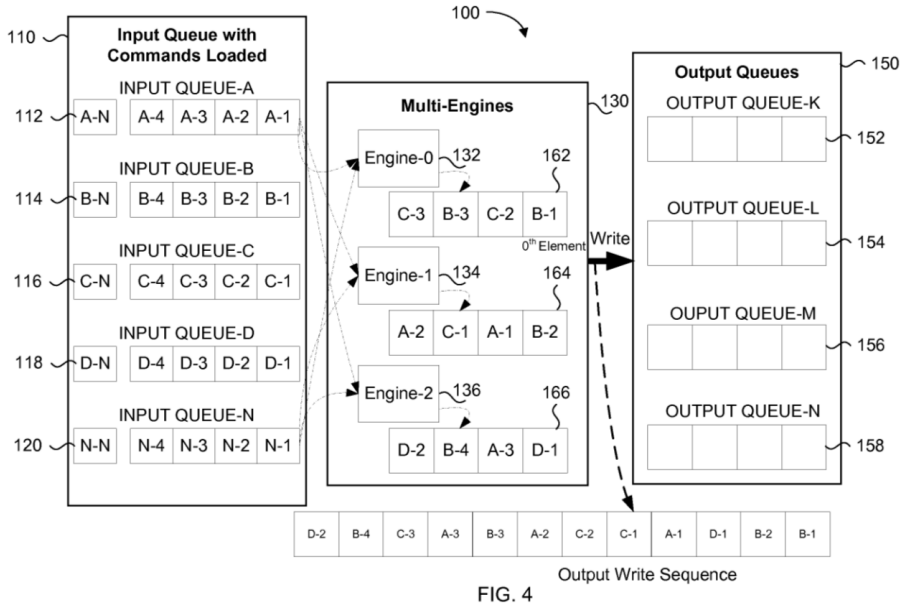
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	B-3 was processed and ready for write out. Processed commands from command input queue C are written as: C-1, C-2, C-3; wherein command C-1 was processed by Engine-1, commands C-2 and C-3 were processed by Engine-0. In the example embodiment command C-2 was processed earlier than command C-1, but it was not written out until the processing of command C-1 was completed. Processed command C-2 was stored in dedicated output buffer 162 of Engine-0136 until command C-1 was processed and ready for write out. Processed commands from command input queue D are written as: D-1, D-2; wherein commands D-1 and D-2 were processed by same engine sequentially and hence could be written out in that order without any waiting period." Dongare '439 at col. 9, ll. 5–50. "In an aspect, the present disclosure provides a multi-engine processing system comprising: an input queue handler configured to receive a plurality of input streams, each input stream having a plurality of input segments, the plurality of input segments ordered relative to other input segments within the input stream; a tagging module configured to append an identity tag to a plurality of input segments in each input stream to produce a plurality of tagged input segments, the identity tag including information indicating the input stream and the relative order within the input stream of each of the plurality of input segments; a scheduler configured to iteratively assign each of the plurality of tagged input segments to any one of a plurality of processing engines based on availability of the processing engines in the relative order of the tagged input segments within the respective input stream, each of the plurality of tagged input segments being processed in its respective assigned processing engine to produce a corresponding plurality of tagged output segments; a plurality of dedicated output buffers associated with the plurality of processing engines, the plurality of dedicated output buffers configured to store each of the plurality of tagged output segments; and, a reordering module configured to reorder the plurality of tagged output segments to produce a plurality of output streams, each output stream having the same relative order of output segments as in the corresponding input stream." Dongare '439 at col. 9, l. 59 – col. 10, l. 17. "FIG. 5 is a schematic representation of a multi-engine processing system in accordance with this disclosure. The multi-engine processing system 100 comprises an input queue handler 510; a tagging module 520; a scheduler 530; a plurality of processing engines 540; a plurality of output buffers 550; a reordering module 560; and an output queue 570. The input queue handler 510 is configured to receive a plurality of input streams. Each input stream has a plurality of input segments and the plurality of input segments ordered relative to other input segments within the input stream. The tagging module 520 is configured to append an identity tag to a plurality of input segments in each input stream to produce a plurality of tagged input segments. The identity tag includes information indicating the input stream and the relative order within the input stream of each of the plurality of input segments. The scheduler 530 is configured to iteratively assign

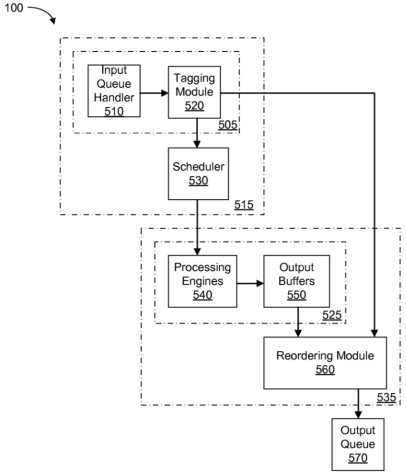
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	each of the plurality of tagged input segments to any one of a plurality of processing engines 540 based on availability of the processing engines 540 in the relative order of the tagged input segments within the respective input stream. Each of the plurality of tagged input segments is processed in its respective assigned processing engine 540 to produce a corresponding plurality of tagged output segments. Each of the plurality of processing engines 540 has a corresponding plurality of dedicated output buffers 550. The plurality of dedicated output buffers 550 are configured to store each of the plurality of tagged output segments. The reordering module 560 is configured to reorder the plurality of tagged output segments to produce a plurality of output queues or streams 570. Each output stream has the same relative order of output segments as in the corresponding input stream." Dongare '439 at col. 10, l. 50 – col. 11, l. 13. "It is noted that the elements of the multi-engine processing system 100 may be implemented as a software routine, hardware circuit, firmware, or a combination thereof. Furthermore, one or more elements of the multi-engine processing system may be implemented as a single element. For example, the input queue handler 510 and the tagging module may be implemented as one element 505. The scheduler may be combined with the input queue handler 510 and the tagging module may be implemented as one element 515, which handles all the pre-processing steps. The plurality of processing engines 540 along with their respective dedicated output buffers 550 may be implemented as one element 525. The reordering module may be combined with the processing engines and their dedicated buffers an element 535. In some examples, the output queues 570 and input queues (not shown in FIG. 5) may be implemented as one element. Other combinations of elements are also possible." Dongare '439 at col. 11, ll. 14–30. "FIG. 6 illustrates an example embodiment of an apparatus 200 for queue ordering in a multi-engine processing environment. Commands in the various command input queues are tagged by appending unique information or identity tags indicating the command input queue identification number (queue number) and its position (sequence number) within the respective command input queue. The identity tags may be appended by tagging module 520 while posting commands in the input queues. In other embodiments, commands may be tagged by the tagging mechanism after reading it from input queues and prior to scheduling the commands for processing by the various engines. Commands from command input queues are processed by an engine of the multi-engine processing system in any order resulting in out-of-order processed commands as described with respect to FIG. 4. Processed commands are stored in dedicated output command buffers of the respective engine that processed the command. Since commands from the same command input queue can be processed by multiple engines in an out-of-order sequence and due to the variable time taken by the engines to process assigned/scheduled

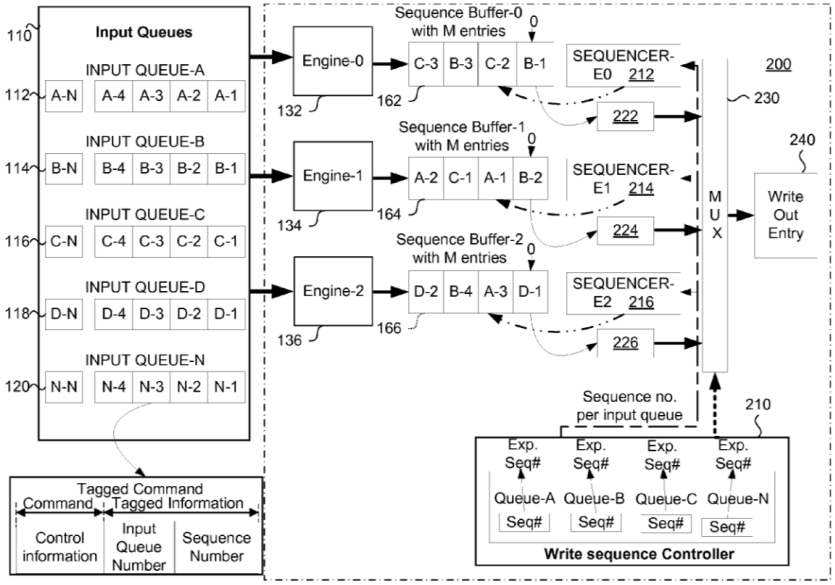
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	commands, the processed commands may be distributed across multiple output buffers." Dongare '439 at col. 11, ll. 31–53. "FIG. 7 provides an illustration of an application of the apparatus of FIG. 5 in a multi-engine processing system. In applications such as RAID controllers, the Host computer 710 is connected to a RAID ASIC 720 via a PCIe protocol bus. RAID controller ASIC 720 has multiple RAID engines 730 to process data and commands to meet required throughput. The Host 710 and RAID controller 720 communicate to each other by using a set of queues 740 on the RAID controller 720. The queues 740 in the controller are classified as input queues 750 from the Host, and output queues from the RAID engines 730. The Host 710 writes commands for the RAID controller into the input queues 750. The input queues 750 are read by the embedded processor 760 and commands are created for RAID engines 730 to process data. Newly generated commands are loaded to input queues 750 dedicated to RAID engines 730. RAID engines 730 reads commands from input queues 750 and after processing them (e.g., read/write operations from Hard Disks 770), writes the processed commands back to output queues 780. The embedded processor 760 monitors output queues 780 for completion of RAID commands in the same order as was loaded in input queues 750. After completing the Host command, the embedded processor 760 will signal the Host 710 that the command is complete by writing into the outbound queue 780. A write to the outbound queue can generate an interrupt to the Host 710." Dongare '439 at col. 14, ll. 31–55.
[1a] an input queue implemented in hardware;	Dongare '439 teaches "an input queue implemented in hardware," including at least the following exemplary citations. For example, Dongare '439 teaches an input queue implemented in hardware, by disclosing, as non–limiting examples, an input queue. Defendants have proposed a construction that "queue implemented in hardware" means "queue implemented in physical circuits without software intervention." The input queue in Dongare '439 meets Defendants' construction because, for example, Dongare '439 teaches a hardware implementation of its teachings, which includes its teachings of queues. *See, e.g.*, Dongare '439 at 11:14–15 ("It is noted that the elements of the multi-engine processing system 100 may be implemented as a software routine, hardware circuit, firmware, or a combination thereof."); *see also id.* at 15:21–26 ("In other instances, well-known electrical structures and circuits are shown in block diagram form in order not to obscure the understanding. For example, specific details are not provided as to whether the embodiments described herein are implemented as a software routine, hardware circuit, firmware, or a combination thereof."). *See, e.g.:*

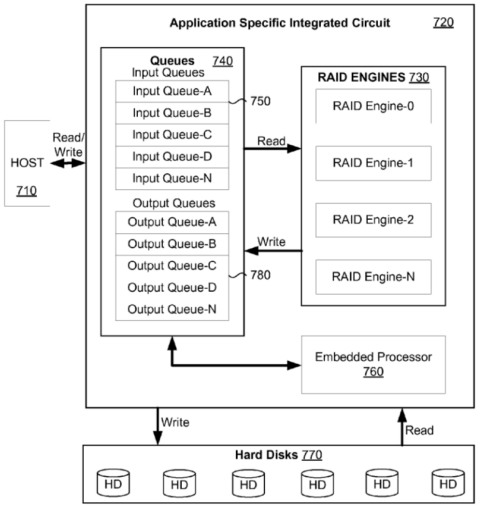
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	"A method and apparatus for queue-ordering commands in multi-engines, multi-queues and/or multi-flows environment is provided. Commands from single/multiple queues and multi-flows are processed by multi-engines with different processing time and/or out of order, which breaks sequential order of commands from same input queue and commands are distributed across multiple engines' output buffer after processing. Processed commands are stored in dedicated command output buffer associated with each engine temporarily. The processed commands are re-ordered while writing out. Also commands can be scheduled to idle engines to achieve maximum throughput, thus utilizing the engines in an optimal manner." Dongare '439 at Abstract.  FIG. 1 Dongare '439 at Figure 1.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 2 Dongare ’439 at Figure 2.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 4 Dongare ’439 at Figure 4.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 5 Dongare ’439 at Figure 5.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 6 Dongare ’439 at Figure 6.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 7 Dongare ’439 at Figure 7. “In a first aspect, the present disclosure provides a method for queue ordering in a multi-engine processing system having a plurality of processing engines to process a plurality of input streams and to provide a corresponding plurality of output streams. The method comprises: receiving the plurality of input streams, each input stream having a plurality of input segments, the plurality of input segments ordered relative to other input segments within the input stream; appending an identity tag to the plurality of input segments in each input stream to produce a plurality of tagged input segments, the identity tag including information indicating the input stream and the relative order within the input stream of each of the plurality of input segments; iteratively assigning each of the plurality of tagged input segments to any one of the plurality of processing engines based on availability of the processing engines in the relative order of the tagged input segments within the respective input stream; processing each of the plurality of tagged input segments in its respective assigned processing engine to produce a corresponding plurality of tagged output segments; storing each of the plurality of tagged output segments in a dedicated output buffer associated with its respective assigned processing engine; and, reordering the plurality of tagged output segments to produce the plurality of output streams, each output stream having the same relative order as in the corresponding input stream.” Dongare ’439 at col. 1, ll. 36–60.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	"In another aspect, the present disclosure provides a multi-engine processing system comprising: an input queue handler configured to receive a plurality of input streams, each input stream having a plurality of input segments, the plurality of input segments ordered relative to other input segments within the input stream; a tagging module configured to append an identity tag to a plurality of input segments in each input stream to produce a plurality of tagged input segments, the identity tag including information indicating the input stream and the relative order within the input stream of each of the plurality of input segments; a scheduler configured to iteratively assign each of the plurality of tagged input segments to any one of a plurality of processing engines based on availability of the processing engines in the relative order of the tagged input segments within the respective input stream, each of the plurality of tagged input segments being processed in its respective assigned processing engine to produce a corresponding plurality of tagged output segments; a plurality of dedicated output buffers associated with the plurality of processing engines, the plurality of dedicated output buffers configured to store each of the plurality of tagged output segments; and, a reordering module configured to reorder the plurality of tagged output segments to produce a plurality of output streams, each output stream having the same relative order of output segments as in the corresponding input stream." Dongare '439 at col. 1, l. 61 – col. 2, l. 19. "Several methods have been proposed to reorder output streams resulting from out-of-order multi-processing. U.S. Pat. No. 7,072,342 describes a system and method of reordering out-of-order packets. In this method, while packets may be processed out of order, the processed packets or commands are written into command FIFO buffers of a command queue only when a current sequence number of the command FIFO buffer matches the packet sequence number of the processed packet. Thus, command FIFO buffer holds ordered commands when read out on a FIFO basis. When there are multiple command buffers, each processed command is associated with a lane number and a sequence number and each command FIFO buffer is dedicated to each input lane. Processed commands are written to a particular command FIFO buffer only when its lane number and sequence number matches with lane and sequence number of the processed command. Consequently, a task has to wait until the sequence number of the command matches the sequence number of the command FIFO buffer, and in effect, one or more tasks remain idle. In addition, the number of command FIFO buffers should be equal to number of input command lanes rendering the system inflexible and rigid." Dongare '439 at col. 2, l. 58 – col. 3, l. 12. "Generally, the present disclosure provides a method and apparatus for queue ordering in a multi-processing system. The method and apparatus can be implemented in multi-engine, multi-queue, and/or multi-flow environments, such as in multi-engine processor systems or in a network domain.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	In a multi-engine processor or a multi-threading system, tasks from same program can be executed concomitantly on different processors to achieve maximum throughput. However, the completed task has to be provided in the order it was posted to the multi-processor system. The method and apparatus of the present disclosure provide for reordering completed tasks efficiently." Dongare '439 at col. 4, ll. 44–55. "FIG. 1 illustrates a multi-engine processing system 100 capable of handling a plurality of input queues or input streams 110. The input queues are sequentially ordered commands or segments to be processed by the plurality of engines 130. The processed commands are written out to the plurality of output queues 150. In the example shown in FIG. 1, the plurality of input queues are received and stored in input queue buffer-A 112, input queue buffer-B 114, input queue buffer-C 116, input queue buffer-D 118, and input queue buffer-N 120. The plurality of processing engines 130 includes Engine-0132, Engine-1134, and Engine-2136. The plurality of output queues 150 are written out and stored in output queue buffer-K 152, output queue buffer-L 154, output queue buffer-M 156, and output queue buffer-N 158. Thus, command streams A, B, C, D through N can be received and stored in the respective input queue buffers and the processed commands can be written out and stored in output queue buffers. As shown in FIG. 1, the input queue buffer-A 112 has commands A-1, A-2, . . . A-N, input queue buffer-B 114 has commands B-1, B-2, . . . B-N, input queue buffer-C 116 has commands C-1, C-2, . . . C-N, input queue buffer-D 118 has commands D-1, D-2, . . . D-N, and input queue buffer-N 120 has commands N-1, N-2, . . . N-N." Dongare '439 at col. 5, ll. 3–26. "It is noted that, for the purposes of the present disclosure, commands within a command input queue are related only in their sequence and are otherwise independent of other commands in the command input queue. Hence, the commands within a particular command input queue can be processed independent of each other. Also, the various command input queues are themselves independent of each other. Furthermore, the command input queues may be buffered in one or more input queue buffer and one input queue buffer may hold commands from more than one command input queue. For a discussion of inter-related commands within a command input queue and inter-related command input queues, reference is made to co-pending US Patent Application entitled Method and Apparatus for a Multi-Engine Descriptor Controller." Dongare '439 at col. 5, ll. 27–41. "Commands within command input queue are appended with identity tags to indicate the command input queue to which the command belongs and to indicate the relative position or order of the command within the command input queue. For example, in FIG. 1, command A-2 refers to the second command in command input queue A, while command C-N refers to the Nth command in command input queue C. Depending on

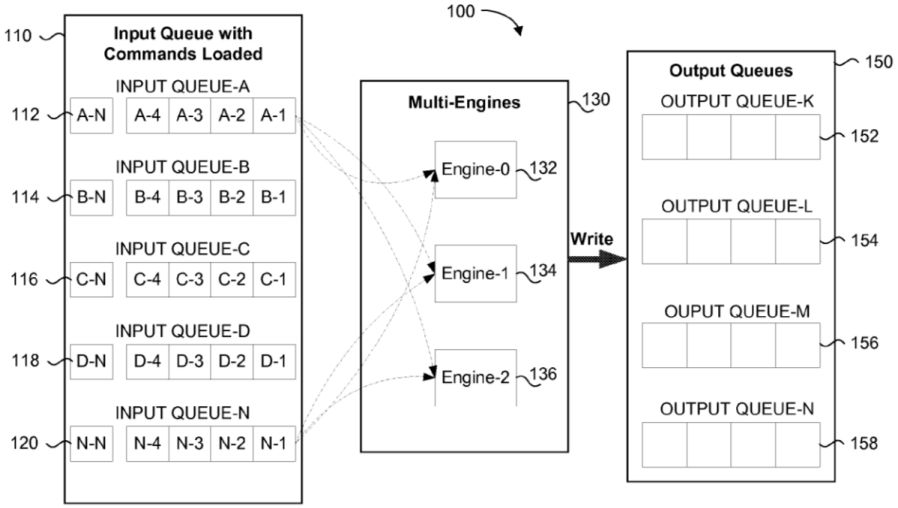
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	system requirements, the identity tags for each command may be appended while receiving and storing the individual commands in the input queue buffer. Alternatively, the identity tags may be appended prior to assigning the commands for processing." Dongare '439 at col. 5, ll. 42–53. "Commands read out from the input queues can be directly scheduled or assigned to any one of the plurality of the engines for processing based on the availability of the processing engine. In other example embodiments, read out commands can be stored in a shared buffer for pre-processing or filtering before scheduling it to a free engine for processing. When commands are stored in shared buffer, they can be provided to an idle engine immediately without fetching." Dongare '439 at col. 5, l. 62 – col. 6, l. 2. "In the example shown in FIG. 1, any one of Engine-0132, Engine-1134, or Engine-2136 is capable of processing the commands read out from the input queues. In an example embodiment, all engines may be identical in architecture and functionality or, in other example embodiments, each engine may differ in functionality and/or architecture, but capable of processing any command assigned to it. Commands from same input queue can be processed on multiple engines depending on availability of the engines at any given instance in order to use the engines optimally. The commands from different input queues may be interleaved and assigned (or scheduled) to an available engine provided the relative order of the commands within the respective command input queue is maintained. For example, Engine-0 can process commands B-1, C-2, B-3, and C-3, as shown in FIG. 2, provided commands C-1 and B-2 have been assigned for processing by other engines prior to scheduling commands C-2 and B-3 for processing by Engine-0." Dongare '439 at col. 6, ll. 3–20. "In the example embodiment shown in FIG. 2, the three engines are identical and are capable of processing any command assigned to them. However, depending on the nature of a command being processed, for example, the byte count and functionality of the command, a command provided earlier to one engine may take more time to process than another command assigned to other engine. Thus, processed commands would be ready to write out at different time points. Also, the sequence of commands from the same input queue would be disturbed at the output side and may be distributed across the dedicated buffers of the respective engines. As shown in FIG. 2, each engine has a dedicated output buffer. For example, Engine-0132 has a dedicated output buffer 162, Engine-1134 has a dedicated output buffer 164, and Engine-2136 has a dedicated output buffer 166." Dongare '439 at col. 6, ll. 28–42. "The scheduling of commands from various command input queues shown in the example embodiment of FIG. 2 with respect to each engine will now be described. Engine-0132 was assigned and has processed commands from command input queues B and C. Commands B-1 and B-3 belong to command input queue B

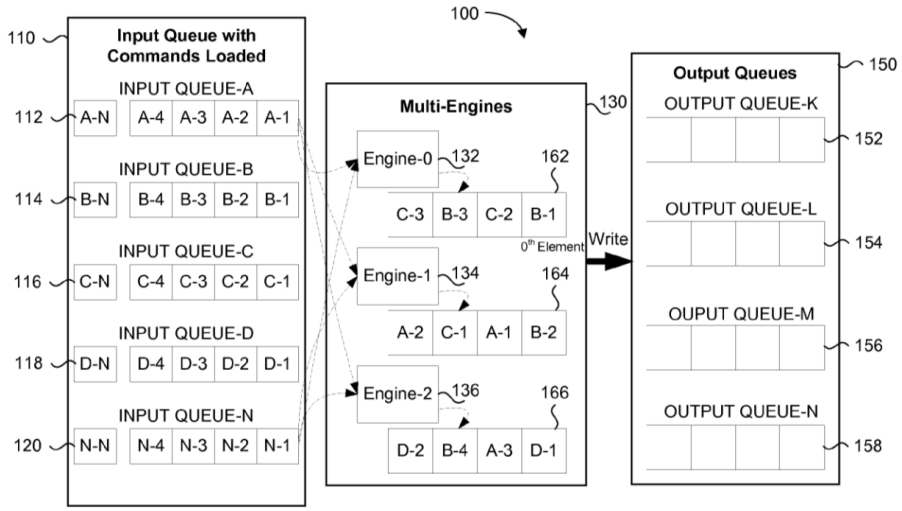
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	and commands C-2 and C-3 belong to command input queue C. Engine-0132 has processed these commands from different command input queues with sequence order: B-1, C-2, B-3, C-3 respectively and has stored the processed commands in its dedicated output buffer 162. Engine-1134 was assigned and has processed commands from command input queues A, B and C. Commands A-1 and A-2 belong to command input queue A, command B-2 belongs to command input queue B, and command C-1 belongs to command input queue C. Engine-1134 has processed these commands from different command input queues with sequence order: B-2, A-1, C-1, A-2 respectively and has stored the processed commands in its dedicated output buffer 164. Engine-2136 was assigned and has processed commands from command input queues A, B and D. Command A-3 belongs to command input queue A, command B-4 belongs to command input queue B, and commands D-1 and D-2 belong to command input queue D. Engine-2136 has processed these commands from different command input queues with sequence order: D-1, A-3, B-4, D-2 respectively and stored the processed commands in its dedicated output buffer 166." Dongare '439 at col. 7, ll. 6–33. "In an example embodiment, the method further comprises storing the plurality of tagged input segments in a shared buffer for pre-processing or filtering prior to assigning each of the plurality of tagged input segments to any one of the plurality of processing engines." Dongare '439 at col. 8, ll. 41–45. "FIG. 4 illustrates the required queue ordering while writing out the processed commands to the output queues, each output queue having the same relative sequence as in the corresponding input queue. As described earlier, the plurality of output queues 150 are written out and stored in output queue buffer-K 152, output queue buffer-L 154, output queue buffer-M 156, and output queue buffer-N 158. The various command input queues A-N are iteratively scheduled or assigned to be processed on any one of the plurality of processing engines based on the availability of the processing engines in the relative order if the commands within their respective command input queue as discussed earlier with respect to the example embodiment of FIG. 2." Dongare '439 at col. 8, ll. 51–63. "In an aspect, the present disclosure provides a multi-engine processing system comprising: an input queue handler configured to receive a plurality of input streams, each input stream having a plurality of input segments, the plurality of input segments ordered relative to other input segments within the input stream; a tagging module configured to append an identity tag to a plurality of input segments in each input stream to produce a plurality of tagged input segments, the identity tag including information indicating the input stream and the relative order within the input stream of each of the plurality of input segments; a scheduler configured to iteratively assign each of the plurality of tagged input segments to any one of a plurality of

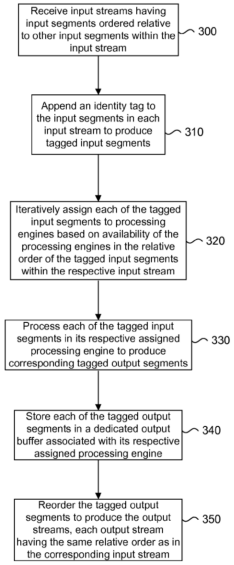
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	processing engines based on availability of the processing engines in the relative order of the tagged input segments within the respective input stream, each of the plurality of tagged input segments being processed in its respective assigned processing engine to produce a corresponding plurality of tagged output segments; a plurality of dedicated output buffers associated with the plurality of processing engines, the plurality of dedicated output buffers configured to store each of the plurality of tagged output segments; and, a reordering module configured to reorder the plurality of tagged output segments to produce a plurality of output streams, each output stream having the same relative order of output segments as in the corresponding input stream." Dongare '439 at col. 9, l. 59 – col. 10, l. 17. "In an example embodiment, the input queue handler comprises input buffers for storing the received plurality of input streams prior to iteratively assigning each of the plurality of tagged input segments to any one of the plurality of processing engines." Dongare '439 at col. 10, ll. 40–44. "In an example embodiment, the input queue handler comprises a shared buffer for storing the plurality of tagged input segments for pre-processing or filtering prior to assigning each of the plurality of tagged input segments to any one of the plurality of processing engines." Dongare '439 at col. 10, ll. 45–49. "FIG. 5 is a schematic representation of a multi-engine processing system in accordance with this disclosure. The multi-engine processing system 100 comprises an input queue handler 510; a tagging module 520; a scheduler 530; a plurality of processing engines 540; a plurality of output buffers 550; a reordering module 560; and an output queue 570. The input queue handler 510 is configured to receive a plurality of input streams. Each input stream has a plurality of input segments and the plurality of input segments ordered relative to other input segments within the input stream. The tagging module 520 is configured to append an identity tag to a plurality of input segments in each input stream to produce a plurality of tagged input segments. The identity tag includes information indicating the input stream and the relative order within the input stream of each of the plurality of input segments. The scheduler 530 is configured to iteratively assign each of the plurality of tagged input segments to any one of a plurality of processing engines 540 based on availability of the processing engines 540 in the relative order of the tagged input segments within the respective input stream. Each of the plurality of tagged input segments is processed in its respective assigned processing engine 540 to produce a corresponding plurality of tagged output segments. Each of the plurality of processing engines 540 has a corresponding plurality of dedicated output buffers 550. The plurality of dedicated output buffers 550 are configured to store each of the plurality of tagged output segments. The reordering module 560 is configured to reorder the plurality of tagged output segments to produce a plurality

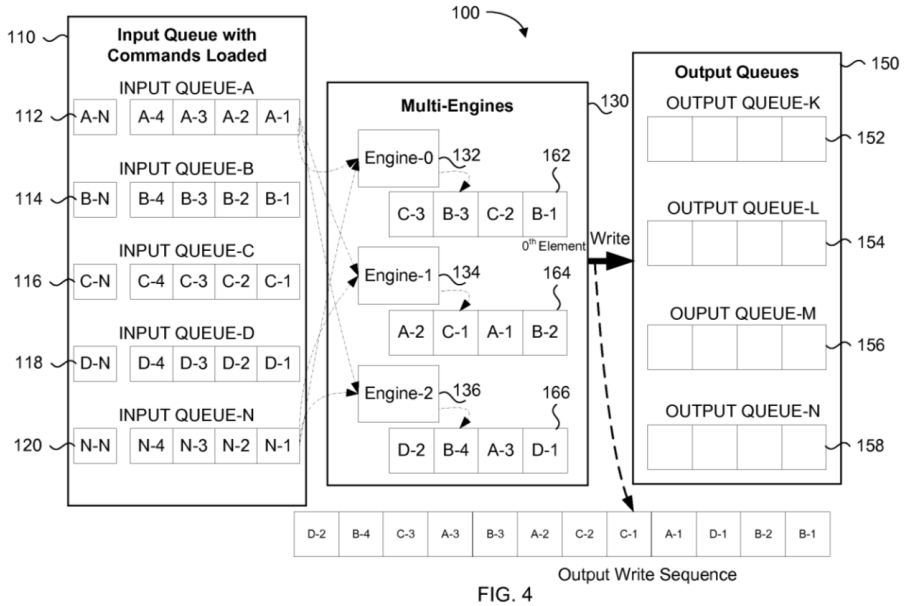
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	of output queues or streams 570. Each output stream has the same relative order of output segments as in the corresponding input stream." Dongare '439 at col. 10, l. 50 – col. 11, l. 13. "It is noted that the elements of the multi-engine processing system 100 may be implemented as a software routine, hardware circuit, firmware, or a combination thereof. Furthermore, one or more elements of the multi-engine processing system may be implemented as a single element. For example, the input queue handler 510 and the tagging module may be implemented as one element 505. The scheduler may be combined with the input queue handler 510 and the tagging module may be implemented as one element 515, which handles all the pre-processing steps. The plurality of processing engines 540 along with their respective dedicated output buffers 550 may be implemented as one element 525. The reordering module may be combined with the processing engines and their dedicated buffers an element 535. In some examples, the output queues 570 and input queues (not shown in FIG. 5) may be implemented as one element. Other combinations of elements are also possible." Dongare '439 at col. 11, ll. 14–30. "FIG. 6 illustrates an example embodiment of an apparatus 200 for queue ordering in a multi-engine processing environment. Commands in the various command input queues are tagged by appending unique information or identity tags indicating the command input queue identification number (queue number) and its position (sequence number) within the respective command input queue. The identity tags may be appended by tagging module 520 while posting commands in the input queues. In other embodiments, commands may be tagged by the tagging mechanism after reading it from input queues and prior to scheduling the commands for processing by the various engines. Commands from command input queues are processed by an engine of the multi-engine processing system in any order resulting in out-of-order processed commands as described with respect to FIG. 4. Processed commands are stored in dedicated output command buffers of the respective engine that processed the command. Since commands from the same command input queue can be processed by multiple engines in an out-of-order sequence and due to the variable time taken by the engines to process assigned/scheduled commands, the processed commands may be distributed across multiple output buffers." Dongare '439 at col. 11, ll. 31–53. "In an example embodiment, commands are tagged with input queue number and sequence number after reading commands from the command input queue. Sequence number indicates position or order of the command within a command input queue. This tagging information makes each command unique in the system at that time/moment. The queue number and the sequence number are utilized at the output side while queue ordering. Tagging information is preserved or maintained by the processing engines. The tagged

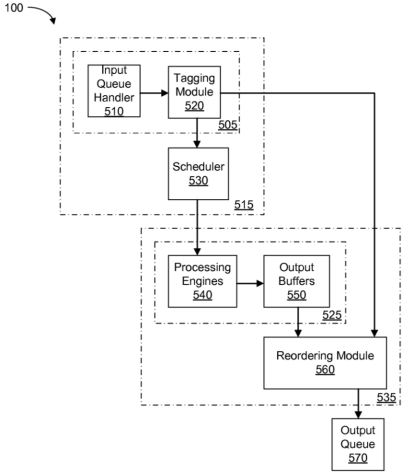
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	information is used by sequencers at output side to select the entry for write out." Dongare '439 at col. 12, ll. 2–12. "FIG. 7 provides an illustration of an application of the apparatus of FIG. 5 in a multi-engine processing system. In applications such as RAID controllers, the Host computer 710 is connected to a RAID ASIC 720 via a PCIe protocol bus. RAID controller ASIC 720 has multiple RAID engines 730 to process data and commands to meet required throughput. The Host 710 and RAID controller 720 communicate to each other by using a set of queues 740 on the RAID controller 720. The queues 740 in the controller are classified as input queues 750 from the Host, and output queues from the RAID engines 730. The Host 710 writes commands for the RAID controller into the input queues 750. The input queues 750 are read by the embedded processor 760 and commands are created for RAID engines 730 to process data. Newly generated commands are loaded to input queues 750 dedicated to RAID engines 730. RAID engines 730 reads commands from input queues 750 and after processing them (e.g., read/write operations from Hard Disks 770), writes the processed commands back to output queues 780. The embedded processor 760 monitors output queues 780 for completion of RAID commands in the same order as was loaded in input queues 750. After completing the Host command, the embedded processor 760 will signal the Host 710 that the command is complete by writing into the outbound queue 780. A write to the outbound queue can generate an interrupt to the Host 710." Dongare '439 at col. 14, ll. 31–55.
[1b] an output queue implemented in hardware;	Dongare '439 teaches "an output queue implemented in hardware," including at least the following exemplary citations. For example, Dongare '439 teaches an output queue implemented in hardware, by disclosing, as non–limiting examples, an output queue. Defendants have proposed a construction that "queue implemented in hardware" means "queue implemented in physical circuits without software intervention." The output queue in Dongare '439 meets Defendants' construction because, for example, Dongare '439 teaches a hardware implementation of its teachings, which includes its teachings of queues. *See, e.g.*, Dongare '439 at 11:14–15 ("It is noted that the elements of the multi-engine processing system 100 may be implemented as a software routine, hardware circuit, firmware, or a combination thereof."); *see also id.* at 15:21–26 ("In other instances, well-known electrical structures and circuits are shown in block diagram form in order not to obscure the understanding. For example, specific details are not provided as to whether the embodiments described herein are implemented as a software routine, hardware circuit, firmware, or a combination thereof."). *See, e.g.:*

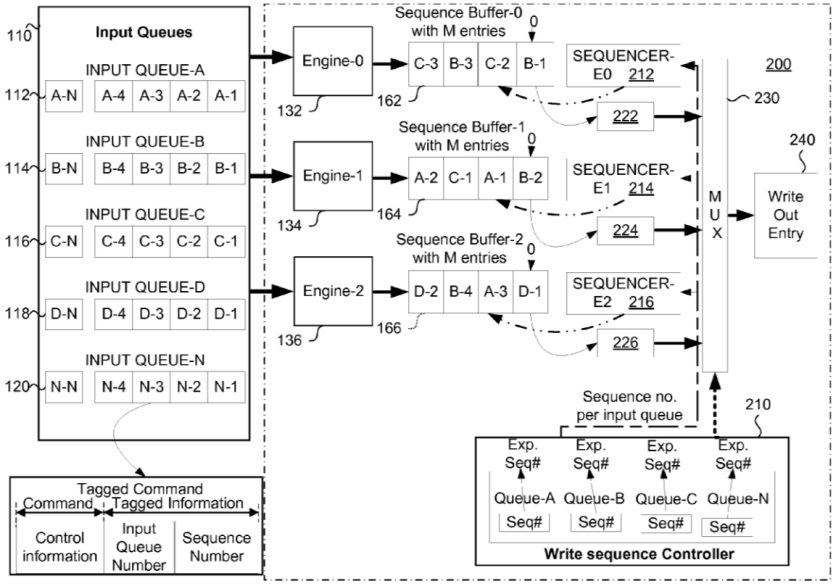
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	"A method and apparatus for queue-ordering commands in multi-engines, multi-queues and/or multi-flows environment is provided. Commands from single/multiple queues and multi-flows are processed by multi-engines with different processing time and/or out of order, which breaks sequential order of commands from same input queue and commands are distributed across multiple engines' output buffer after processing. Processed commands are stored in dedicated command output buffer associated with each engine temporarily. The processed commands are re-ordered while writing out. Also commands can be scheduled to idle engines to achieve maximum throughput, thus utilizing the engines in an optimal manner." Dongare '439 at Abstract.  FIG. 1 Dongare '439 at Figure 1.

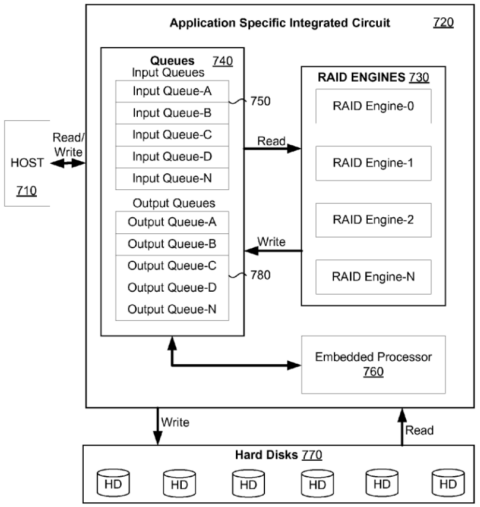
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 2 Dongare ’439 at Figure 2.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 3 Dongare ’439 at Figure 3.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 4 Dongare ’439 at Figure 4.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 5 Dongare ’439 at Figure 5.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 6 Dongare ’439 at Figure 6.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 7 Dongare ’439 at Figure 7. “In a first aspect, the present disclosure provides a method for queue ordering in a multi-engine processing system having a plurality of processing engines to process a plurality of input streams and to provide a corresponding plurality of output streams. The method comprises: receiving the plurality of input streams, each input stream having a plurality of input segments, the plurality of input segments ordered relative to other input segments within the input stream; appending an identity tag to the plurality of input segments in each input stream to produce a plurality of tagged input segments, the identity tag including information indicating the input stream and the relative order within the input stream of each of the plurality of input segments; iteratively assigning each of the plurality of tagged input segments to any one of the plurality of processing engines based on availability of the processing engines in the relative order of the tagged input segments within the respective input stream; processing each of the plurality of tagged input segments in its respective assigned processing engine to produce a corresponding plurality of tagged output segments; storing each of the plurality of tagged output segments in a dedicated output buffer associated with its respective assigned processing engine; and, reordering the plurality of tagged output segments to produce the plurality of output streams, each output stream having the same relative order as in the corresponding input stream.” Dongare ’439 at col. 1, ll. 36–60.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	"In another aspect, the present disclosure provides a multi-engine processing system comprising: an input queue handler configured to receive a plurality of input streams, each input stream having a plurality of input segments, the plurality of input segments ordered relative to other input segments within the input stream; a tagging module configured to append an identity tag to a plurality of input segments in each input stream to produce a plurality of tagged input segments, the identity tag including information indicating the input stream and the relative order within the input stream of each of the plurality of input segments; a scheduler configured to iteratively assign each of the plurality of tagged input segments to any one of a plurality of processing engines based on availability of the processing engines in the relative order of the tagged input segments within the respective input stream, each of the plurality of tagged input segments being processed in its respective assigned processing engine to produce a corresponding plurality of tagged output segments; a plurality of dedicated output buffers associated with the plurality of processing engines, the plurality of dedicated output buffers configured to store each of the plurality of tagged output segments; and, a reordering module configured to reorder the plurality of tagged output segments to produce a plurality of output streams, each output stream having the same relative order of output segments as in the corresponding input stream." Dongare '439 at col. 1, l. 61 – col. 2, l. 19. "Several methods have been proposed to reorder output streams resulting from out-of-order multi-processing. U.S. Pat. No. 7,072,342 describes a system and method of reordering out-of-order packets. In this method, while packets may be processed out of order, the processed packets or commands are written into command FIFO buffers of a command queue only when a current sequence number of the command FIFO buffer matches the packet sequence number of the processed packet. Thus, command FIFO buffer holds ordered commands when read out on a FIFO basis. When there are multiple command buffers, each processed command is associated with a lane number and a sequence number and each command FIFO buffer is dedicated to each input lane. Processed commands are written to a particular command FIFO buffer only when its lane number and sequence number matches with lane and sequence number of the processed command. Consequently, a task has to wait until the sequence number of the command matches the sequence number of the command FIFO buffer, and in effect, one or more tasks remain idle. In addition, the number of command FIFO buffers should be equal to number of input command lanes rendering the system inflexible and rigid." Dongare '439 at col. 2, l. 58 – col. 3, l. 12. "Generally, the present disclosure provides a method and apparatus for queue ordering in a multi-processing system. The method and apparatus can be implemented in multi-engine, multi-queue, and/or multi-flow environments, such as in multi-engine processor systems or in a network domain.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	In a multi-engine processor or a multi-threading system, tasks from same program can be executed concomitantly on different processors to achieve maximum throughput. However, the completed task has to be provided in the order it was posted to the multi-processor system. The method and apparatus of the present disclosure provide for reordering completed tasks efficiently." Dongare '439 at col. 4, ll. 44–55. "FIG. 1 illustrates a multi-engine processing system 100 capable of handling a plurality of input queues or input streams 110. The input queues are sequentially ordered commands or segments to be processed by the plurality of engines 130. The processed commands are written out to the plurality of output queues 150. In the example shown in FIG. 1, the plurality of input queues are received and stored in input queue buffer-A 112, input queue buffer-B 114, input queue buffer-C 116, input queue buffer-D 118, and input queue buffer-N 120. The plurality of processing engines 130 includes Engine-0132, Engine-1134, and Engine-2136. The plurality of output queues 150 are written out and stored in output queue buffer-K 152, output queue buffer-L 154, output queue buffer-M 156, and output queue buffer-N 158. Thus, command streams A, B, C, D through N can be received and stored in the respective input queue buffers and the processed commands can be written out and stored in output queue buffers. As shown in FIG. 1, the input queue buffer-A 112 has commands A-1, A-2, . . . A-N, input queue buffer-B 114 has commands B-1, B-2, . . . B-N, input queue buffer-C 116 has commands C-1, C-2, . . . C-N, input queue buffer-D 118 has commands D-1, D-2, . . . D-N, and input queue buffer-N 120 has commands N-1, N-2, . . . N-N." Dongare '439 at col. 5, ll. 3–26. "In the example embodiment shown in FIG. 2, the three engines are identical and are capable of processing any command assigned to them. However, depending on the nature of a command being processed, for example, the byte count and functionality of the command, a command provided earlier to one engine may take more time to process than another command assigned to other engine. Thus, processed commands would be ready to write out at different time points. Also, the sequence of commands from the same input queue would be disturbed at the output side and may be distributed across the dedicated buffers of the respective engines. As shown in FIG. 2, each engine has a dedicated output buffer. For example, Engine-0132 has a dedicated output buffer 162, Engine-1134 has a dedicated output buffer 164, and Engine-2136 has a dedicated output buffer 166." Dongare '439 at col. 6, ll. 28–42. "The scheduling of commands from various command input queues shown in the example embodiment of FIG. 2 with respect to each engine will now be described. Engine-0132 was assigned and has processed commands from command input queues B and C. Commands B-1 and B-3 belong to command input queue B and commands C-2 and C-3 belong to command input queue C. Engine-0132 has processed these commands

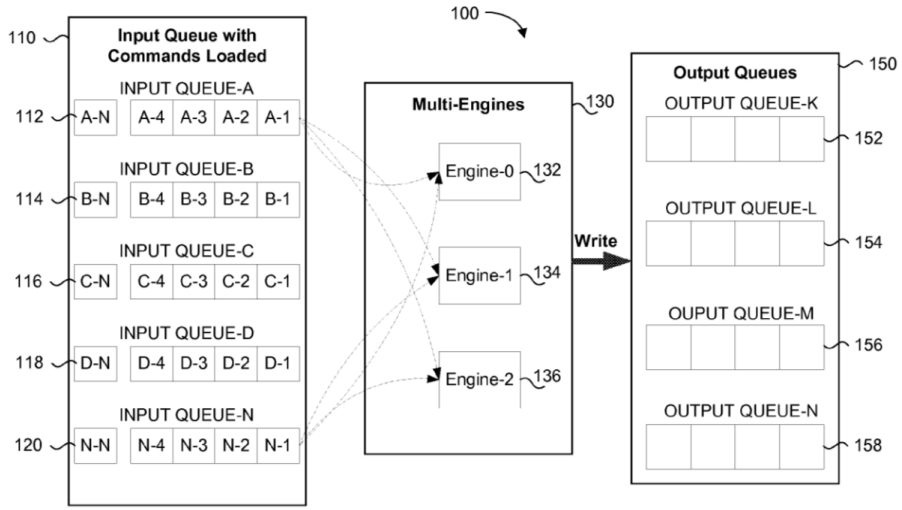
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	from different command input queues with sequence order: B-1, C-2, B-3, C-3 respectively and has stored the processed commands in its dedicated output buffer 162. Engine-1134 was assigned and has processed commands from command input queues A, B and C. Commands A-1 and A-2 belong to command input queue A, command B-2 belongs to command input queue B, and command C-1 belongs to command input queue C. Engine-1134 has processed these commands from different command input queues with sequence order: B-2, A-1, C-1, A-2 respectively and has stored the processed commands in its dedicated output buffer 164. Engine-2136 was assigned and has processed commands from command input queues A, B and D. Command A-3 belongs to command input queue A, command B-4 belongs to command input queue B, and commands D-1 and D-2 belong to command input queue D. Engine-2136 has processed these commands from different command input queues with sequence order: D-1, A-3, B-4, D-2 respectively and stored the processed commands in its dedicated output buffer 166." Dongare '439 at col. 7, ll. 6–33. "FIG. 4 illustrates the required queue ordering while writing out the processed commands to the output queues, each output queue having the same relative sequence as in the corresponding input queue. As described earlier, the plurality of output queues 150 are written out and stored in output queue buffer-K 152, output queue buffer-L 154, output queue buffer-M 156, and output queue buffer-N 158. The various command input queues A-N are iteratively scheduled or assigned to be processed on any one of the plurality of processing engines based on the availability of the processing engines in the relative order if the commands within their respective command input queue as discussed earlier with respect to the example embodiment of FIG. 2." Dongare '439 at col. 8, ll. 51–63. "The processed commands are stored in the dedicated output buffer associated with its respective assigned processing engines and may be in any order relative to other processed commands within its command input queue and also may be interleaved with processed commands from other command input queues. For example, the dedicated output buffer 166 associated with Engine-2136 has processed commands D-1, A-3, B-4 and D-2 from commands input queues D, A and B." Dongare '439 at col. 8, l. 64 – col. 9, l. 4. "In order to maintain the original order or sequence of the commands within a given command input queue, the processed commands must be written out to the command output queues in the same sequence as it was read out from the respective command input queue. It follows that A-1 should be written before A-2, A-2 should be written before A-3 and so on. As shown in FIG. 4, commands are written in the same relative order as in the command input queue: B-1, B-2, D-1, A-1, C-1, C-2, A-2, B-3, A-3, C-3, B-4, and D-2, i.e., the

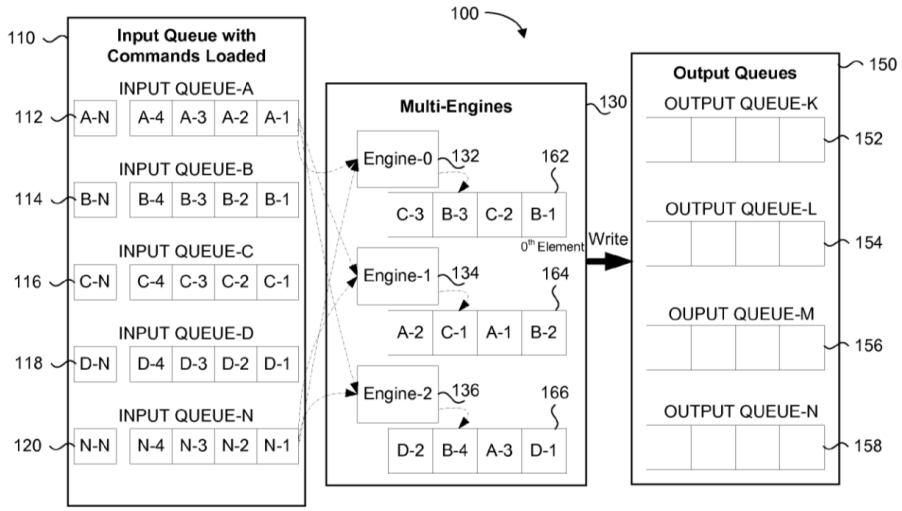
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	relative order of commands within a command output queue is maintained in the write out sequence irrespective of the processing engine that processed each command. In the example embodiment of FIG. 4, processed commands from command input queue A are written out as: A-1, A-2 and A-3, wherein commands A-1 and A-2 commands were processed by Engine-1 and command A-3 was processed by Engine-2. In the example embodiment, command A-3 was processed earlier to command A-2 but it was not written out since command A-2 was not processed. Processed command A-3 was stored in the dedicated output buffer 166 of Engine-2136 until command A-2 was processed and ready for write out. Similarly, processed commands from command input queue B are written out as: B-1, B-2, B-3 and B-4, wherein commands B-1 and B-3 were processed by Engine-0, command B-2 was processed by Engine-1 and command B4 was processed by Engine-2. In the example embodiment commands B-1 and B-2 were processed at the same time by different engines and hence could be written out sequentially. However, command B-4 was processed earlier by Engine-2 but it was not written out as command B-3 was not processed. Processed command B-4 was stored in dedicated output buffer 166 of Engine-2136 until command B-3 was processed and ready for write out. Processed commands from command input queue C are written as: C-1, C-2, C-3; wherein command C-1 was processed by Engine-1, commands C-2 and C-3 were processed by Engine-0. In the example embodiment command C-2 was processed earlier than command C-1, but it was not written out until the processing of command C-1 was completed. Processed command C-2 was stored in dedicated output buffer 162 of Engine-0136 until command C-1 was processed and ready for write out. Processed commands from command input queue D are written as: D-1, D-2; wherein commands D-1 and D-2 were processed by same engine sequentially and hence could be written out in that order without any waiting period." Dongare '439 at col. 9, ll. 5–50. "In an aspect, the present disclosure provides a multi-engine processing system comprising: an input queue handler configured to receive a plurality of input streams, each input stream having a plurality of input segments, the plurality of input segments ordered relative to other input segments within the input stream; a tagging module configured to append an identity tag to a plurality of input segments in each input stream to produce a plurality of tagged input segments, the identity tag including information indicating the input stream and the relative order within the input stream of each of the plurality of input segments; a scheduler configured to iteratively assign each of the plurality of tagged input segments to any one of a plurality of processing engines based on availability of the processing engines in the relative order of the tagged input segments within the respective input stream, each of the plurality of tagged input segments being processed in its respective assigned processing engine to produce a corresponding plurality of tagged output segments; a

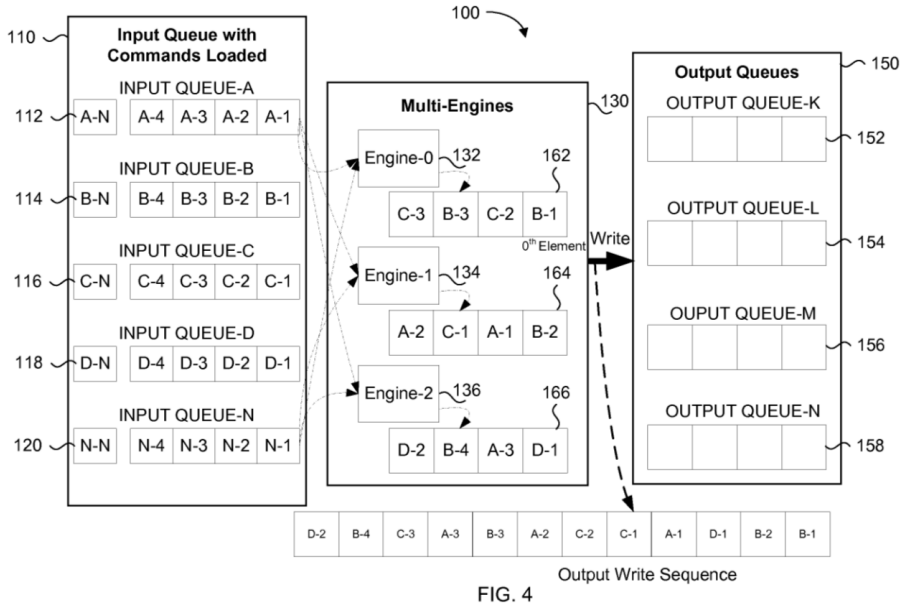
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	plurality of dedicated output buffers associated with the plurality of processing engines, the plurality of dedicated output buffers configured to store each of the plurality of tagged output segments; and, a reordering module configured to reorder the plurality of tagged output segments to produce a plurality of output streams, each output stream having the same relative order of output segments as in the corresponding input stream." Dongare '439 at col. 9, l. 59 – col. 10, l. 17. "FIG. 5 is a schematic representation of a multi-engine processing system in accordance with this disclosure. The multi-engine processing system 100 comprises an input queue handler 510; a tagging module 520; a scheduler 530; a plurality of processing engines 540; a plurality of output buffers 550; a reordering module 560; and an output queue 570. The input queue handler 510 is configured to receive a plurality of input streams. Each input stream has a plurality of input segments and the plurality of input segments ordered relative to other input segments within the input stream. The tagging module 520 is configured to append an identity tag to a plurality of input segments in each input stream to produce a plurality of tagged input segments. The identity tag includes information indicating the input stream and the relative order within the input stream of each of the plurality of input segments. The scheduler 530 is configured to iteratively assign each of the plurality of tagged input segments to any one of a plurality of processing engines 540 based on availability of the processing engines 540 in the relative order of the tagged input segments within the respective input stream. Each of the plurality of tagged input segments is processed in its respective assigned processing engine 540 to produce a corresponding plurality of tagged output segments. Each of the plurality of processing engines 540 has a corresponding plurality of dedicated output buffers 550. The plurality of dedicated output buffers 550 are configured to store each of the plurality of tagged output segments. The reordering module 560 is configured to reorder the plurality of tagged output segments to produce a plurality of output queues or streams 570. Each output stream has the same relative order of output segments as in the corresponding input stream." Dongare '439 at col. 10, l. 50 – col. 11, l. 13. "It is noted that the elements of the multi-engine processing system 100 may be implemented as a software routine, hardware circuit, firmware, or a combination thereof. Furthermore, one or more elements of the multi-engine processing system may be implemented as a single element. For example, the input queue handler 510 and the tagging module may be implemented as one element 505. The scheduler may be combined with the input queue handler 510 and the tagging module may be implemented as one element 515, which handles all the pre-processing steps. The plurality of processing engines 540 along with their respective dedicated output buffers 550 may be implemented as one element 525. The reordering module may be combined with the processing engines and their dedicated buffers an element 535. In some examples, the

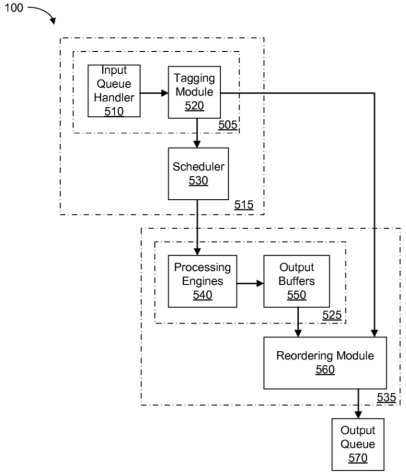
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	output queues 570 and input queues (not shown in FIG. 5) may be implemented as one element. Other combinations of elements are also possible." Dongare '439 at col. 11, ll. 14–30. "FIG. 6 illustrates an example embodiment of an apparatus 200 for queue ordering in a multi-engine processing environment. Commands in the various command input queues are tagged by appending unique information or identity tags indicating the command input queue identification number (queue number) and its position (sequence number) within the respective command input queue. The identity tags may be appended by tagging module 520 while posting commands in the input queues. In other embodiments, commands may be tagged by the tagging mechanism after reading it from input queues and prior to scheduling the commands for processing by the various engines. Commands from command input queues are processed by an engine of the multi-engine processing system in any order resulting in out-of-order processed commands as described with respect to FIG. 4. Processed commands are stored in dedicated output command buffers of the respective engine that processed the command. Since commands from the same command input queue can be processed by multiple engines in an out-of-order sequence and due to the variable time taken by the engines to process assigned/scheduled commands, the processed commands may be distributed across multiple output buffers." Dongare '439 at col. 11, ll. 31–53. "Command Output Buffer: After processing of commands, each engine writes the processed commands in its dedicated command output buffer. Processed commands (along with their identity tags, also referred to as tagged output segments) are written to the command output buffer in the sequence they are processed by engines. The depth of the command output buffers can be chosen based on the bandwidth required by the system. The depth of a command output buffer refers to the number of processed commands or segments that can be stored in the command output buffer. By having a dedicated command buffer for each engine, each engine can directly access its command output buffer to write out processed commands without any waiting time. Furthermore, the engine can be freed to process the next command in the pipeline thereby increasing processing throughput. Thus, engine utilization can be optimized and engine idle times can be reduced. The processed commands in the command output buffers wait for write out until it is selected by the sequencer. Thus, processed commands from the command output buffers are read out in proper queue order without the requirement for additional sorting means." Dongare '439 at col. 12, ll. 14–36. "FIG. 7 provides an illustration of an application of the apparatus of FIG. 5 in a multi-engine processing system. In applications such as RAID controllers, the Host computer 710 is connected to a RAID ASIC 720

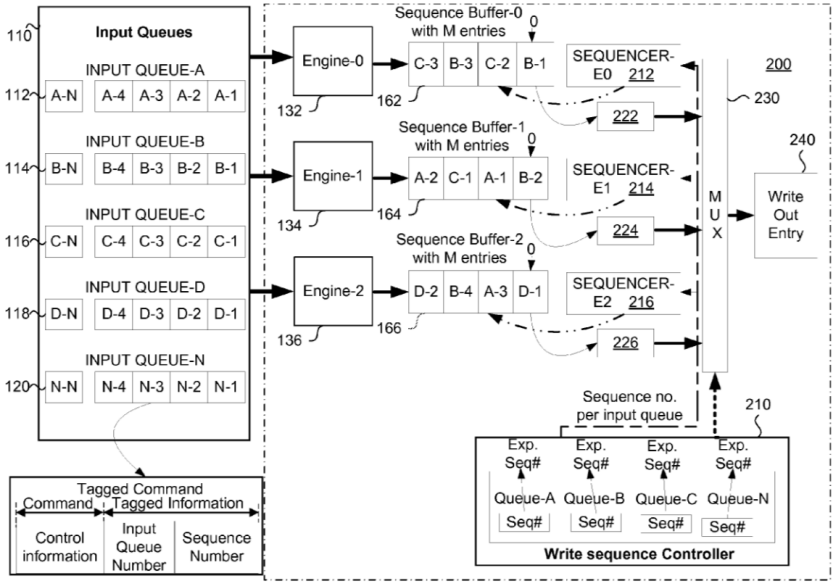
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	via a PCIe protocol bus. RAID controller ASIC 720 has multiple RAID engines 730 to process data and commands to meet required throughput. The Host 710 and RAID controller 720 communicate to each other by using a set of queues 740 on the RAID controller 720. The queues 740 in the controller are classified as input queues 750 from the Host, and output queues from the RAID engines 730. The Host 710 writes commands for the RAID controller into the input queues 750. The input queues 750 are read by the embedded processor 760 and commands are created for RAID engines 730 to process data. Newly generated commands are loaded to input queues 750 dedicated to RAID engines 730. RAID engines 730 reads commands from input queues 750 and after processing them (e.g., read/write operations from Hard Disks 770), writes the processed commands back to output queues 780. The embedded processor 760 monitors output queues 780 for completion of RAID commands in the same order as was loaded in input queues 750. After completing the Host command, the embedded processor 760 will signal the Host 710 that the command is complete by writing into the outbound queue 780. A write to the outbound queue can generate an interrupt to the Host 710." Dongare '439 at col. 14, ll. 31–55.
[1c] a first processing element having access to said input queue and said output queue;	Dongare '439 teaches "a first processing element having access to said input queue and said output queue," including at least the following exemplary citations. For example, Dongare '439 teaches a first processing element having access to said input queue and said output queue, by disclosing, as non–limiting examples, a host processor that communicates with an ASIC that contains input and output queues via a PCIe interface. *See, e.g.*, Dongare '439 at 14:31–34 ("In applications such as RAID controllers, the Host computer 710 is connected to a RAID ASIC 720 via a PCIe protocol bus.") The host processor has access to both the input queue and output queue. *See, e.g.:* "A method and apparatus for queue-ordering commands in multi-engines, multi-queues and/or multi-flows environment is provided. Commands from single/multiple queues and multi-flows are processed by multi-engines with different processing time and/or out of order, which breaks sequential order of commands from same input queue and commands are distributed across multiple engines' output buffer after processing. Processed commands are stored in dedicated command output buffer associated with each engine temporarily. The processed commands are re-ordered while writing out. Also commands can be scheduled to idle engines to achieve maximum throughput, thus utilizing the engines in an optimal manner." Dongare '439 at Abstract.

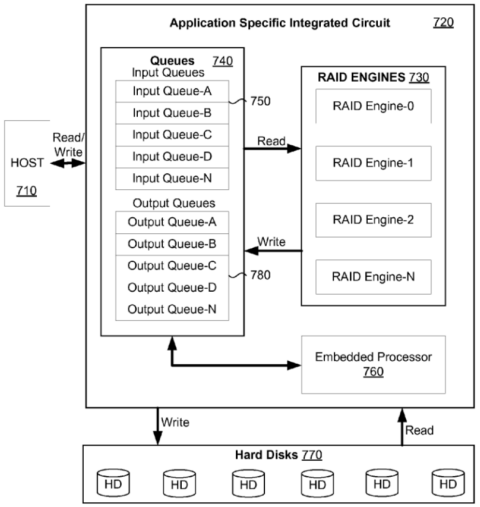
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 1 Dongare ’439 at Figure 1.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 2 Dongare ’439 at Figure 2.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 4 Dongare ’439 at Figure 4.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 5 Dongare ’439 at Figure 5.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 6 Dongare ’439 at Figure 6.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 7 Dongare ’439 at Figure 7. “Multiple engines or processors are used to increase the throughput of a device or a network system. Typically, segments of one or more input streams are parallel processed in multiple processors to increase the processing efficiency or throughput of the system. However, out of order processing of the individual segments may result in a loss of order in the sequence of the segments within the input stream and the resulting output stream may not correspond to the input stream. Additional processing is then required to restore the order of the segments of the output stream relative to the input stream resulting in a compromise of the efficiencies gained by multi-processing.” Dongare ’439 at col. 1, ll. 21–32. “In a first aspect, the present disclosure provides a method for queue ordering in a multi-engine processing system having a plurality of processing engines to process a plurality of input streams and to provide a corresponding plurality of output streams. The method comprises: receiving the plurality of input streams, each input stream having a plurality of input segments, the plurality of input segments ordered relative to other input segments within the input stream; appending an identity tag to the plurality of input segments in each input stream to produce a plurality of tagged input segments, the identity tag including information indicating the input stream and the relative order within the input stream of each of the plurality of input segments; iteratively assigning each of the plurality of tagged input segments to any one of the plurality of

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	processing engines based on availability of the processing engines in the relative order of the tagged input segments within the respective input stream; processing each of the plurality of tagged input segments in its respective assigned processing engine to produce a corresponding plurality of tagged output segments; storing each of the plurality of tagged output segments in a dedicated output buffer associated with its respective assigned processing engine; and, reordering the plurality of tagged output segments to produce the plurality of output streams, each output stream having the same relative order as in the corresponding input stream." Dongare '439 at col. 1, ll. 36–60. "In another aspect, the present disclosure provides a multi-engine processing system comprising: an input queue handler configured to receive a plurality of input streams, each input stream having a plurality of input segments, the plurality of input segments ordered relative to other input segments within the input stream; a tagging module configured to append an identity tag to a plurality of input segments in each input stream to produce a plurality of tagged input segments, the identity tag including information indicating the input stream and the relative order within the input stream of each of the plurality of input segments; a scheduler configured to iteratively assign each of the plurality of tagged input segments to any one of a plurality of processing engines based on availability of the processing engines in the relative order of the tagged input segments within the respective input stream, each of the plurality of tagged input segments being processed in its respective assigned processing engine to produce a corresponding plurality of tagged output segments; a plurality of dedicated output buffers associated with the plurality of processing engines, the plurality of dedicated output buffers configured to store each of the plurality of tagged output segments; and, a reordering module configured to reorder the plurality of tagged output segments to produce a plurality of output streams, each output stream having the same relative order of output segments as in the corresponding input stream." Dongare '439 at col. 1, l. 61 – col. 2, l. 19. "Generally, the present disclosure provides a method and apparatus for queue ordering in a multi-processing system. The method and apparatus can be implemented in multi-engine, multi-queue, and/or multi-flow environments, such as in multi-engine processor systems or in a network domain. In a multi-engine processor or a multi-threading system, tasks from same program can be executed concomitantly on different processors to achieve maximum throughput. However, the completed task has to be provided in the order it was posted to the multi-processor system. The method and apparatus of the present disclosure provide for reordering completed tasks efficiently." Dongare '439 at col. 4, ll. 44–55. "FIG. 1 illustrates a multi-engine processing system 100 capable of handling a plurality of input queues or input streams 110. The input queues are sequentially ordered commands or segments to be processed by the

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	plurality of engines 130. The processed commands are written out to the plurality of output queues 150. In the example shown in FIG. 1, the plurality of input queues are received and stored in input queue buffer-A 112, input queue buffer-B 114, input queue buffer-C 116, input queue buffer-D 118, and input queue buffer-N 120. The plurality of processing engines 130 includes Engine-0132, Engine-1134, and Engine-2136. The plurality of output queues 150 are written out and stored in output queue buffer-K 152, output queue buffer-L 154, output queue buffer-M 156, and output queue buffer-N 158. Thus, command streams A, B, C, D through N can be received and stored in the respective input queue buffers and the processed commands can be written out and stored in output queue buffers. As shown in FIG. 1, the input queue buffer-A 112 has commands A-1, A-2, . . . A-N, input queue buffer-B 114 has commands B-1, B-2, . . . B-N, input queue buffer-C 116 has commands C-1, C-2, . . . C-N, input queue buffer-D 118 has commands D-1, D-2, . . . D-N, and input queue buffer-N 120 has commands N-1, N-2, . . . N-N." Dongare '439 at col. 5, ll. 3–26. "Commands read out from the input queues can be directly scheduled or assigned to any one of the plurality of the engines for processing based on the availability of the processing engine. In other example embodiments, read out commands can be stored in a shared buffer for pre-processing or filtering before scheduling it to a free engine for processing. When commands are stored in shared buffer, they can be provided to an idle engine immediately without fetching." Dongare '439 at col. 5, l. 62 – col. 6, l. 2. "In the example shown in FIG. 1, any one of Engine-0132, Engine-1134, or Engine-2136 is capable of processing the commands read out from the input queues. In an example embodiment, all engines may be identical in architecture and functionality or, in other example embodiments, each engine may differ in functionality and/or architecture, but capable of processing any command assigned to it. Commands from same input queue can be processed on multiple engines depending on availability of the engines at any given instance in order to use the engines optimally. The commands from different input queues may be interleaved and assigned (or scheduled) to an available engine provided the relative order of the commands within the respective command input queue is maintained. For example, Engine-0 can process commands B-1, C-2, B-3, and C-3, as shown in FIG. 2, provided commands C-1 and B-2 have been assigned for processing by other engines prior to scheduling commands C-2 and B-3 for processing by Engine-0." Dongare '439 at col. 6, ll. 3–20. "Once processed, the processed commands can be written to output queues with a single write channel or multiple write channels. In certain example embodiments, an input queue may also act as an output queue. The processed command may be buffered in one or more output queue buffers and one output queue buffer

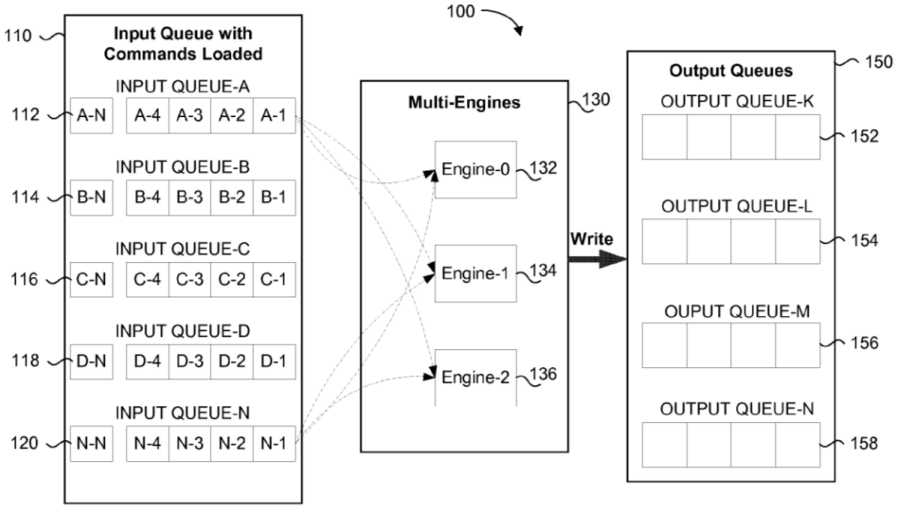
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	may hold processed commands from more than one command output queue." Dongare '439 at col. 6, ll. 21–27. "In the example embodiment shown in FIG. 2, the three engines are identical and are capable of processing any command assigned to them. However, depending on the nature of a command being processed, for example, the byte count and functionality of the command, a command provided earlier to one engine may take more time to process than another command assigned to other engine. Thus, processed commands would be ready to write out at different time points. Also, the sequence of commands from the same input queue would be disturbed at the output side and may be distributed across the dedicated buffers of the respective engines. As shown in FIG. 2, each engine has a dedicated output buffer. For example, Engine-0132 has a dedicated output buffer 162, Engine-1134 has a dedicated output buffer 164, and Engine-2136 has a dedicated output buffer 166." Dongare '439 at col. 6, ll. 28–42. "The scheduling of commands from various command input queues shown in the example embodiment of FIG. 2 with respect to each engine will now be described. Engine-0132 was assigned and has processed commands from command input queues B and C. Commands B-1 and B-3 belong to command input queue B and commands C-2 and C-3 belong to command input queue C. Engine-0132 has processed these commands from different command input queues with sequence order: B-1, C-2, B-3, C-3 respectively and has stored the processed commands in its dedicated output buffer 162. Engine-1134 was assigned and has processed commands from command input queues A, B and C. Commands A-1 and A-2 belong to command input queue A, command B-2 belongs to command input queue B, and command C-1 belongs to command input queue C. Engine-1134 has processed these commands from different command input queues with sequence order: B-2, A-1, C-1, A-2 respectively and has stored the processed commands in its dedicated output buffer 164. Engine-2136 was assigned and has processed commands from command input queues A, B and D. Command A-3 belongs to command input queue A, command B-4 belongs to command input queue B, and commands D-1 and D-2 belong to command input queue D. Engine-2136 has processed these commands from different command input queues with sequence order: D-1, A-3, B-4, D-2 respectively and stored the processed commands in its dedicated output buffer 166." Dongare '439 at col. 7, ll. 6–33. "In an example embodiment, the method further comprises storing the plurality of tagged input segments in a shared buffer for pre-processing or filtering prior to assigning each of the plurality of tagged input segments to any one of the plurality of processing engines." Dongare '439 at col. 8, ll. 41–45.

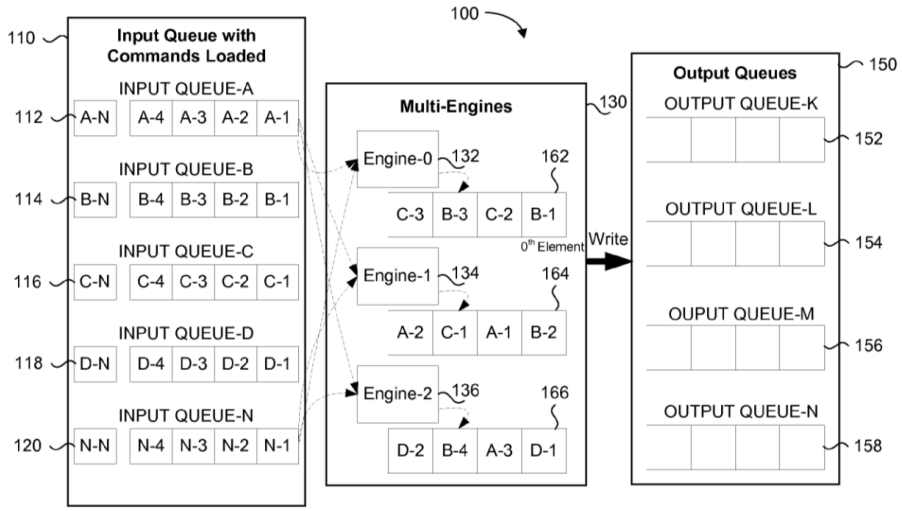
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	"FIG. 4 illustrates the required queue ordering while writing out the processed commands to the output queues, each output queue having the same relative sequence as in the corresponding input queue. As described earlier, the plurality of output queues 150 are written out and stored in output queue buffer-K 152, output queue buffer-L 154, output queue buffer-M 156, and output queue buffer-N 158. The various command input queues A-N are iteratively scheduled or assigned to be processed on any one of the plurality of processing engines based on the availability of the processing engines in the relative order if the commands within their respective command input queue as discussed earlier with respect to the example embodiment of FIG. 2." Dongare '439 at col. 8, ll. 51–63. "In order to maintain the original order or sequence of the commands within a given command input queue, the processed commands must be written out to the command output queues in the same sequence as it was read out from the respective command input queue. It follows that A-1 should be written before A-2, A-2 should be written before A-3 and so on. As shown in FIG. 4, commands are written in the same relative order as in the command input queue: B-1, B-2, D-1, A-1, C-1, C-2, A-2, B-3, A-3, C-3, B-4, and D-2, i.e., the relative order of commands within a command output queue is maintained in the write out sequence irrespective of the processing engine that processed each command. In the example embodiment of FIG. 4, processed commands from command input queue A are written out as: A-1, A-2 and A-3, wherein commands A-1 and A-2 commands were processed by Engine-1 and command A-3 was processed by Engine-2. In the example embodiment, command A-3 was processed earlier to command A-2 but it was not written out since command A-2 was not processed. Processed command A-3 was stored in the dedicated output buffer 166 of Engine-2136 until command A-2 was processed and ready for write out. Similarly, processed commands from command input queue B are written out as: B-1, B-2, B-3 and B-4, wherein commands B-1 and B-3 were processed by Engine-0, command B-2 was processed by Engine-1 and command B4 was processed by Engine-2. In the example embodiment commands B-1 and B-2 were processed at the same time by different engines and hence could be written out sequentially. However, command B-4 was processed earlier by Engine-2 but it was not written out as command B-3 was not processed. Processed command B-4 was stored in dedicated output buffer 166 of Engine-2136 until command B-3 was processed and ready for write out. Processed commands from command input queue C are written as: C-1, C-2, C-3; wherein command C-1 was processed by Engine-1, commands C-2 and C-3 were processed by Engine-0. In the example embodiment command C-2 was processed earlier than command C-1, but it was not written out until the processing of command C-1 was completed. Processed command C-2 was stored in dedicated output buffer 162 of Engine-0136 until command C-1 was processed and ready for write out.

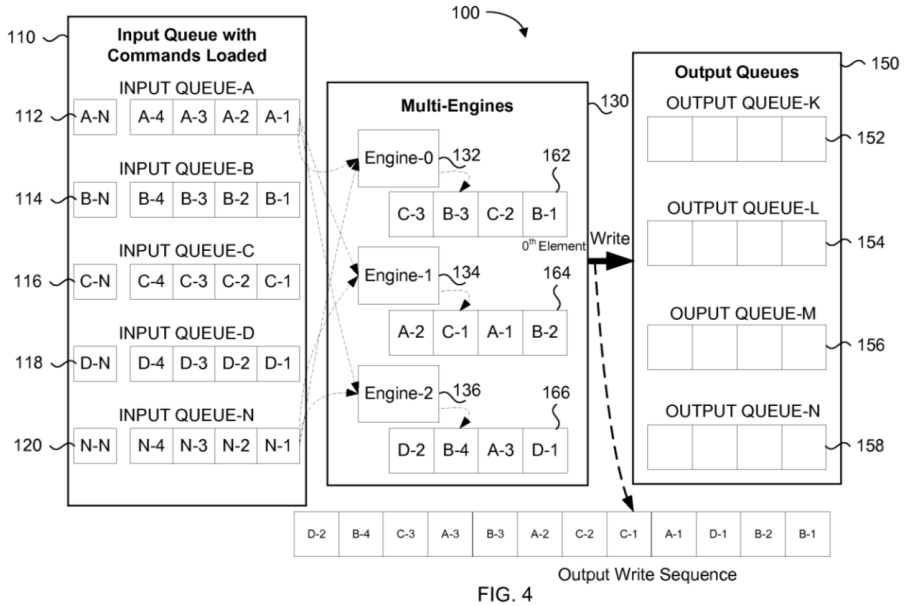
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	Processed commands from command input queue D are written as: D-1, D-2; wherein commands D-1 and D-2 were processed by same engine sequentially and hence could be written out in that order without any waiting period." Dongare '439 at col. 9, ll. 5–50. "In an aspect, the present disclosure provides a multi-engine processing system comprising: an input queue handler configured to receive a plurality of input streams, each input stream having a plurality of input segments, the plurality of input segments ordered relative to other input segments within the input stream; a tagging module configured to append an identity tag to a plurality of input segments in each input stream to produce a plurality of tagged input segments, the identity tag including information indicating the input stream and the relative order within the input stream of each of the plurality of input segments; a scheduler configured to iteratively assign each of the plurality of tagged input segments to any one of a plurality of processing engines based on availability of the processing engines in the relative order of the tagged input segments within the respective input stream, each of the plurality of tagged input segments being processed in its respective assigned processing engine to produce a corresponding plurality of tagged output segments; a plurality of dedicated output buffers associated with the plurality of processing engines, the plurality of dedicated output buffers configured to store each of the plurality of tagged output segments; and, a reordering module configured to reorder the plurality of tagged output segments to produce a plurality of output streams, each output stream having the same relative order of output segments as in the corresponding input stream." Dongare '439 at col. 9, l. 59 – col. 10, l. 17. "FIG. 5 is a schematic representation of a multi-engine processing system in accordance with this disclosure. The multi-engine processing system 100 comprises an input queue handler 510; a tagging module 520; a scheduler 530; a plurality of processing engines 540; a plurality of output buffers 550; a reordering module 560; and an output queue 570. The input queue handler 510 is configured to receive a plurality of input streams. Each input stream has a plurality of input segments and the plurality of input segments ordered relative to other input segments within the input stream. The tagging module 520 is configured to append an identity tag to a plurality of input segments in each input stream to produce a plurality of tagged input segments. The identity tag includes information indicating the input stream and the relative order within the input stream of each of the plurality of input segments. The scheduler 530 is configured to iteratively assign each of the plurality of tagged input segments to any one of a plurality of processing engines 540 based on availability of the processing engines 540 in the relative order of the tagged input segments within the respective input stream. Each of the plurality of tagged input segments is processed in its respective assigned processing engine 540 to produce a corresponding plurality of tagged output segments. Each of the plurality of processing engines 540 has a corresponding plurality of dedicated output buffers 550. The plurality of

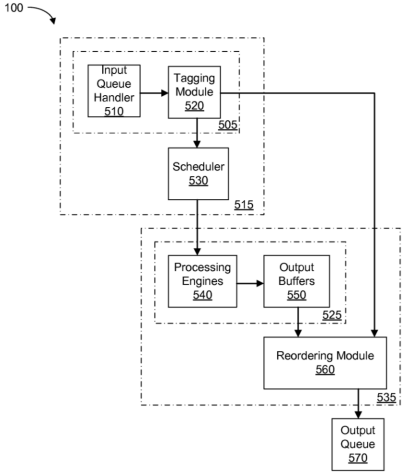
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	dedicated output buffers 550 are configured to store each of the plurality of tagged output segments. The reordering module 560 is configured to reorder the plurality of tagged output segments to produce a plurality of output queues or streams 570. Each output stream has the same relative order of output segments as in the corresponding input stream." Dongare '439 at col. 10, l. 50 – col. 11, l. 13. "FIG. 6 illustrates an example embodiment of an apparatus 200 for queue ordering in a multi-engine processing environment. Commands in the various command input queues are tagged by appending unique information or identity tags indicating the command input queue identification number (queue number) and its position (sequence number) within the respective command input queue. The identity tags may be appended by tagging module 520 while posting commands in the input queues. In other embodiments, commands may be tagged by the tagging mechanism after reading it from input queues and prior to scheduling the commands for processing by the various engines. Commands from command input queues are processed by an engine of the multi-engine processing system in any order resulting in out-of-order processed commands as described with respect to FIG. 4. Processed commands are stored in dedicated output command buffers of the respective engine that processed the command. Since commands from the same command input queue can be processed by multiple engines in an out-of-order sequence and due to the variable time taken by the engines to process assigned/scheduled commands, the processed commands may be distributed across multiple output buffers." Dongare '439 at col. 11, ll. 31–53. "In an example embodiment, commands are tagged with input queue number and sequence number after reading commands from the command input queue. Sequence number indicates position or order of the command within a command input queue. This tagging information makes each command unique in the system at that time/moment. The queue number and the sequence number are utilized at the output side while queue ordering. Tagging information is preserved or maintained by the processing engines. The tagged information is used by sequencers at output side to select the entry for write out." Dongare '439 at col. 12, ll. 2–12. "Command Output Buffer: After processing of commands, each engine writes the processed commands in its dedicated command output buffer. Processed commands (along with their identity tags, also referred to as tagged output segments) are written to the command output buffer in the sequence they are processed by engines. The depth of the command output buffers can be chosen based on the bandwidth required by the system. The depth of a command output buffer refers to the number of processed commands or segments that can be stored in the

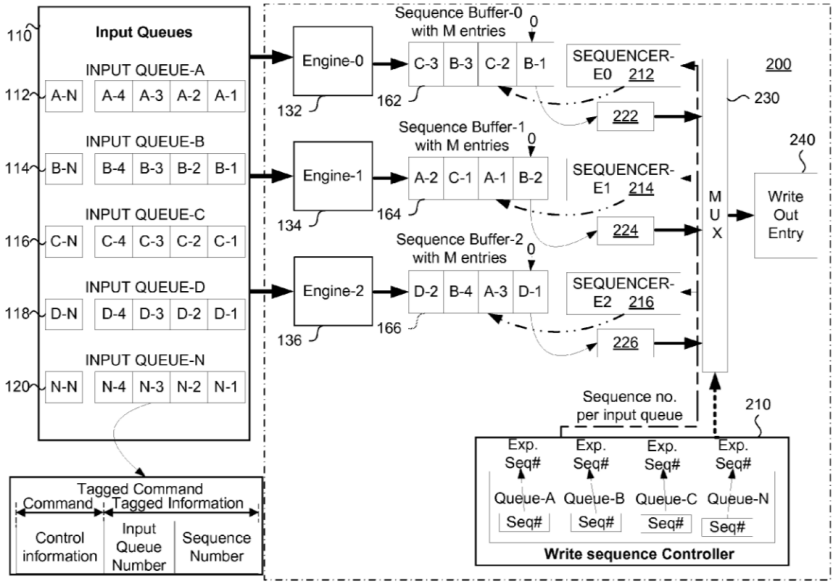
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	command output buffer. By having a dedicated command buffer for each engine, each engine can directly access its command output buffer to write out processed commands without any waiting time. Furthermore, the engine can be freed to process the next command in the pipeline thereby increasing processing throughput. Thus, engine utilization can be optimized and engine idle times can be reduced. The processed commands in the command output buffers wait for write out until it is selected by the sequencer. Thus, processed commands from the command output buffers are read out in proper queue order without the requirement for additional sorting means." Dongare '439 at col. 12, ll. 14–36. "FIG. 7 provides an illustration of an application of the apparatus of FIG. 5 in a multi-engine processing system. In applications such as RAID controllers, the Host computer 710 is connected to a RAID ASIC 720 via a PCIe protocol bus. RAID controller ASIC 720 has multiple RAID engines 730 to process data and commands to meet required throughput. The Host 710 and RAID controller 720 communicate to each other by using a set of queues 740 on the RAID controller 720. The queues 740 in the controller are classified as input queues 750 from the Host, and output queues from the RAID engines 730. The Host 710 writes commands for the RAID controller into the input queues 750. The input queues 750 are read by the embedded processor 760 and commands are created for RAID engines 730 to process data. Newly generated commands are loaded to input queues 750 dedicated to RAID engines 730. RAID engines 730 reads commands from input queues 750 and after processing them (e.g., read/write operations from Hard Disks 770), writes the processed commands back to output queues 780. The embedded processor 760 monitors output queues 780 for completion of RAID commands in the same order as was loaded in input queues 750. After completing the Host command, the embedded processor 760 will signal the Host 710 that the command is complete by writing into the outbound queue 780. A write to the outbound queue can generate an interrupt to the Host 710." Dongare '439 at col. 14, ll. 31–55.
[1d] at least one second processing element in communication with said first processing element;	Dongare '439 teaches "at least one second processing element in communication with said first processing element," including at least the following exemplary citations. For example, Dongare '439 teaches at least one second processing element in communication with said first processing element, by disclosing, as non–limiting examples, an embedded processor in an ASIC that communicates with a host processor. The first processing element is the host processor, while the second processing element would be the embedded processor. The embedded processor, the second processing element is in communication with the first processing element. *See, e.g.:*

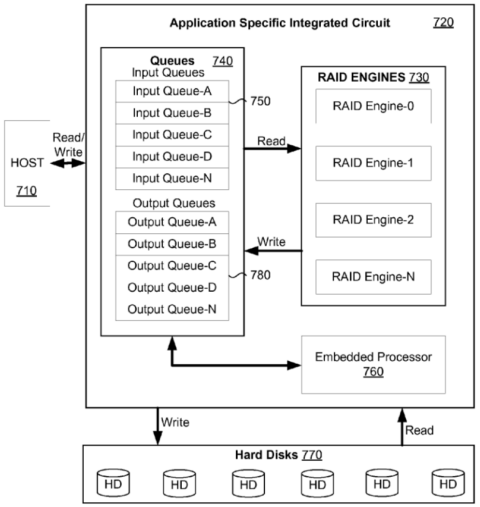
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	"A method and apparatus for queue-ordering commands in multi-engines, multi-queues and/or multi-flows environment is provided. Commands from single/multiple queues and multi-flows are processed by multi-engines with different processing time and/or out of order, which breaks sequential order of commands from same input queue and commands are distributed across multiple engines' output buffer after processing. Processed commands are stored in dedicated command output buffer associated with each engine temporarily. The processed commands are re-ordered while writing out. Also commands can be scheduled to idle engines to achieve maximum throughput, thus utilizing the engines in an optimal manner." Dongare '439 at Abstract.  FIG. 1 Dongare '439 at Figure 1.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 2 Dongare ’439 at Figure 2.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 4 Dongare ’439 at Figure 4.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 5 Dongare ’439 at Figure 5.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 6 Dongare ’439 at Figure 6.

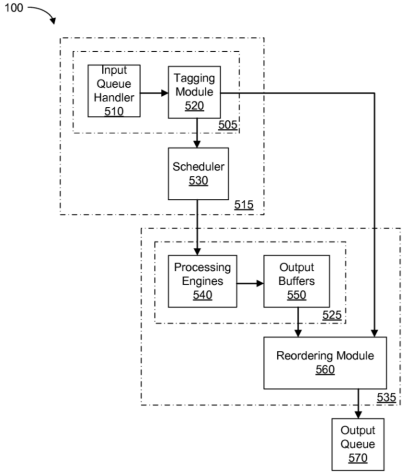
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 7 Dongare ’439 at Figure 7. “Multiple engines or processors are used to increase the throughput of a device or a network system. Typically, segments of one or more input streams are parallel processed in multiple processors to increase the processing efficiency or throughput of the system. However, out of order processing of the individual segments may result in a loss of order in the sequence of the segments within the input stream and the resulting output stream may not correspond to the input stream. Additional processing is then required to restore the order of the segments of the output stream relative to the input stream resulting in a compromise of the efficiencies gained by multi-processing.” Dongare ’439 at col. 1, ll. 21–32. “In a first aspect, the present disclosure provides a method for queue ordering in a multi-engine processing system having a plurality of processing engines to process a plurality of input streams and to provide a corresponding plurality of output streams. The method comprises: receiving the plurality of input streams, each input stream having a plurality of input segments, the plurality of input segments ordered relative to other input segments within the input stream; appending an identity tag to the plurality of input segments in each input stream to produce a plurality of tagged input segments, the identity tag including information indicating the input stream and the relative order within the input stream of each of the plurality of input segments; iteratively assigning each of the plurality of tagged input segments to any one of the plurality of

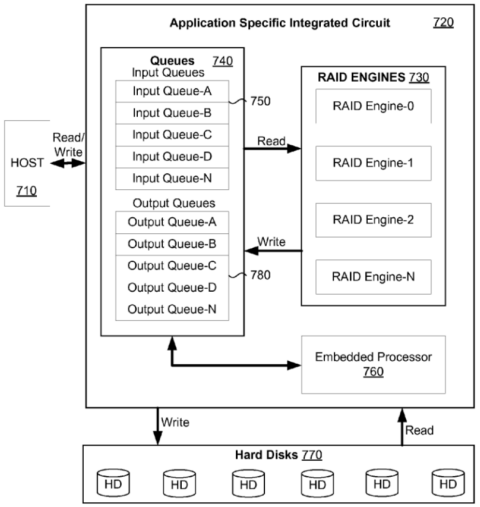
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	processing engines based on availability of the processing engines in the relative order of the tagged input segments within the respective input stream; processing each of the plurality of tagged input segments in its respective assigned processing engine to produce a corresponding plurality of tagged output segments; storing each of the plurality of tagged output segments in a dedicated output buffer associated with its respective assigned processing engine; and, reordering the plurality of tagged output segments to produce the plurality of output streams, each output stream having the same relative order as in the corresponding input stream." Dongare '439 at col. 1, ll. 36–60. "In another aspect, the present disclosure provides a multi-engine processing system comprising: an input queue handler configured to receive a plurality of input streams, each input stream having a plurality of input segments, the plurality of input segments ordered relative to other input segments within the input stream; a tagging module configured to append an identity tag to a plurality of input segments in each input stream to produce a plurality of tagged input segments, the identity tag including information indicating the input stream and the relative order within the input stream of each of the plurality of input segments; a scheduler configured to iteratively assign each of the plurality of tagged input segments to any one of a plurality of processing engines based on availability of the processing engines in the relative order of the tagged input segments within the respective input stream, each of the plurality of tagged input segments being processed in its respective assigned processing engine to produce a corresponding plurality of tagged output segments; a plurality of dedicated output buffers associated with the plurality of processing engines, the plurality of dedicated output buffers configured to store each of the plurality of tagged output segments; and, a reordering module configured to reorder the plurality of tagged output segments to produce a plurality of output streams, each output stream having the same relative order of output segments as in the corresponding input stream." Dongare '439 at col. 1, l. 61 – col. 2, l. 19. "Generally, the present disclosure provides a method and apparatus for queue ordering in a multi-processing system. The method and apparatus can be implemented in multi-engine, multi-queue, and/or multi-flow environments, such as in multi-engine processor systems or in a network domain. In a multi-engine processor or a multi-threading system, tasks from same program can be executed concomitantly on different processors to achieve maximum throughput. However, the completed task has to be provided in the order it was posted to the multi-processor system. The method and apparatus of the present disclosure provide for reordering completed tasks efficiently." Dongare '439 at col. 4, ll. 44–55. "FIG. 1 illustrates a multi-engine processing system 100 capable of handling a plurality of input queues or input streams 110. The input queues are sequentially ordered commands or segments to be processed by the

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	plurality of engines 130. The processed commands are written out to the plurality of output queues 150. In the example shown in FIG. 1, the plurality of input queues are received and stored in input queue buffer-A 112, input queue buffer-B 114, input queue buffer-C 116, input queue buffer-D 118, and input queue buffer-N 120. The plurality of processing engines 130 includes Engine-0132, Engine-1134, and Engine-2136. The plurality of output queues 150 are written out and stored in output queue buffer-K 152, output queue buffer-L 154, output queue buffer-M 156, and output queue buffer-N 158. Thus, command streams A, B, C, D through N can be received and stored in the respective input queue buffers and the processed commands can be written out and stored in output queue buffers. As shown in FIG. 1, the input queue buffer-A 112 has commands A-1, A-2, . . . A-N, input queue buffer-B 114 has commands B-1, B-2, . . . B-N, input queue buffer-C 116 has commands C-1, C-2, . . . C-N, input queue buffer-D 118 has commands D-1, D-2, . . . D-N, and input queue buffer-N 120 has commands N-1, N-2, . . . N-N." Dongare '439 at col. 5, ll. 3–26. "Commands read out from the input queues can be directly scheduled or assigned to any one of the plurality of the engines for processing based on the availability of the processing engine. In other example embodiments, read out commands can be stored in a shared buffer for pre-processing or filtering before scheduling it to a free engine for processing. When commands are stored in shared buffer, they can be provided to an idle engine immediately without fetching." Dongare '439 at col. 5, l. 62 – col. 6, l. 2. "In the example shown in FIG. 1, any one of Engine-0132, Engine-1134, or Engine-2136 is capable of processing the commands read out from the input queues. In an example embodiment, all engines may be identical in architecture and functionality or, in other example embodiments, each engine may differ in functionality and/or architecture, but capable of processing any command assigned to it. Commands from same input queue can be processed on multiple engines depending on availability of the engines at any given instance in order to use the engines optimally. The commands from different input queues may be interleaved and assigned (or scheduled) to an available engine provided the relative order of the commands within the respective command input queue is maintained. For example, Engine-0 can process commands B-1, C-2, B-3, and C-3, as shown in FIG. 2, provided commands C-1 and B-2 have been assigned for processing by other engines prior to scheduling commands C-2 and B-3 for processing by Engine-0." Dongare '439 at col. 6, ll. 3–20. "Once processed, the processed commands can be written to output queues with a single write channel or multiple write channels. In certain example embodiments, an input queue may also act as an output queue. The processed command may be buffered in one or more output queue buffers and one output queue buffer

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	may hold processed commands from more than one command output queue." Dongare '439 at col. 6, ll. 21–27. "In the example embodiment shown in FIG. 2, the three engines are identical and are capable of processing any command assigned to them. However, depending on the nature of a command being processed, for example, the byte count and functionality of the command, a command provided earlier to one engine may take more time to process than another command assigned to other engine. Thus, processed commands would be ready to write out at different time points. Also, the sequence of commands from the same input queue would be disturbed at the output side and may be distributed across the dedicated buffers of the respective engines. As shown in FIG. 2, each engine has a dedicated output buffer. For example, Engine-0132 has a dedicated output buffer 162, Engine-1134 has a dedicated output buffer 164, and Engine-2136 has a dedicated output buffer 166." Dongare '439 at col. 6, ll. 28–42. "In an aspect, the present disclosure provides a multi-engine processing system comprising: an input queue handler configured to receive a plurality of input streams, each input stream having a plurality of input segments, the plurality of input segments ordered relative to other input segments within the input stream; a tagging module configured to append an identity tag to a plurality of input segments in each input stream to produce a plurality of tagged input segments, the identity tag including information indicating the input stream and the relative order within the input stream of each of the plurality of input segments; a scheduler configured to iteratively assign each of the plurality of tagged input segments to any one of a plurality of processing engines based on availability of the processing engines in the relative order of the tagged input segments within the respective input stream, each of the plurality of tagged input segments being processed in its respective assigned processing engine to produce a corresponding plurality of tagged output segments; a plurality of dedicated output buffers associated with the plurality of processing engines, the plurality of dedicated output buffers configured to store each of the plurality of tagged output segments; and, a reordering module configured to reorder the plurality of tagged output segments to produce a plurality of output streams, each output stream having the same relative order of output segments as in the corresponding input stream." Dongare '439 at col. 9, l. 59 – col. 10, l. 17. "FIG. 7 provides an illustration of an application of the apparatus of FIG. 5 in a multi-engine processing system. In applications such as RAID controllers, the Host computer 710 is connected to a RAID ASIC 720 via a PCIe protocol bus. RAID controller ASIC 720 has multiple RAID engines 730 to process data and commands to meet required throughput. The Host 710 and RAID controller 720 communicate to each other by using a set of queues 740 on the RAID controller 720. The queues 740 in the controller are classified as

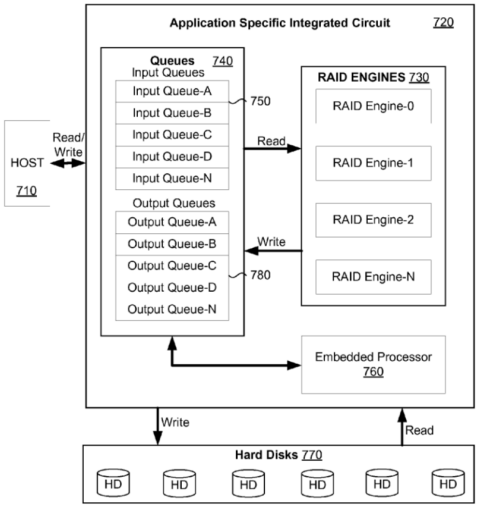
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	input queues 750 from the Host, and output queues from the RAID engines 730. The Host 710 writes commands for the RAID controller into the input queues 750. The input queues 750 are read by the embedded processor 760 and commands are created for RAID engines 730 to process data. Newly generated commands are loaded to input queues 750 dedicated to RAID engines 730. RAID engines 730 reads commands from input queues 750 and after processing them (e.g., read/write operations from Hard Disks 770), writes the processed commands back to output queues 780. The embedded processor 760 monitors output queues 780 for completion of RAID commands in the same order as was loaded in input queues 750. After completing the Host command, the embedded processor 760 will signal the Host 710 that the command is complete by writing into the outbound queue 780. A write to the outbound queue can generate an interrupt to the Host 710." Dongare '439 at col. 14, ll. 31–55.
[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;	
[1f] computer storage storing instructions, which when executed by said at least one second processing element:	Dongare '439 teaches computer storage storing instructions executed by said at least one second processing element, including at least the following exemplary citations. For example, Dongare '439 teaches computer storage storing instructions that are executed by said at least one second processing element, by disclosing, as non–limiting examples, an embedded processor that executes instructions. *See, e.g.,* Dongare '439 at 15:32–41 ("The machine-readable medium can be any suitable tangible, non-transitory medium, including magnetic, optical, or electrical storage medium including a diskette, compact disk read only memory (CD-ROM), memory device (volatile or non-volatile), or similar storage mechanism. The machine-readable medium can contain various sets of instructions, code sequences, configuration information, or other data, which, when executed, cause a processor to perform steps in a method according to an embodiment of the disclosure."). The embedded processor executes those instructions stored in the computer storage. *See, e.g.:*

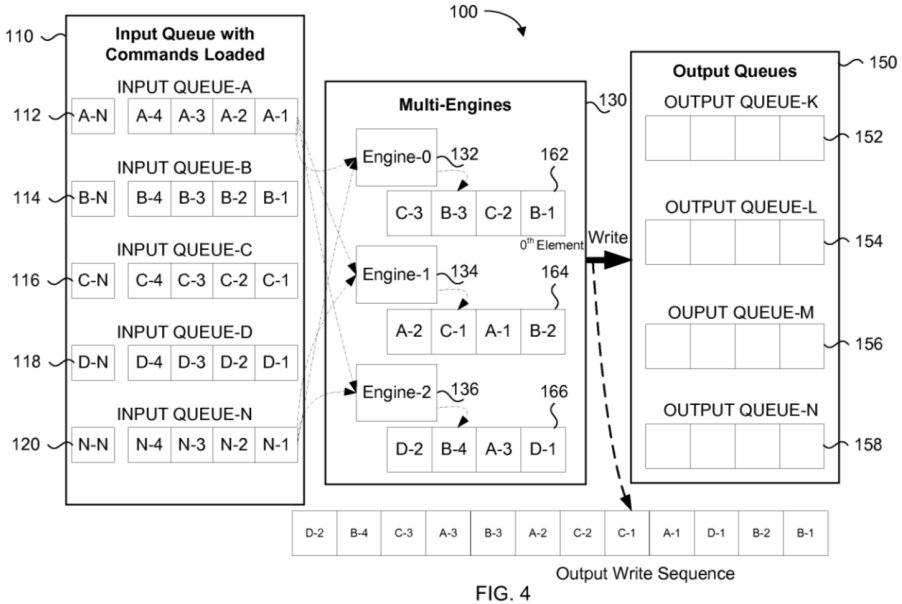
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 5 Dongare ’439 at Figure 5.

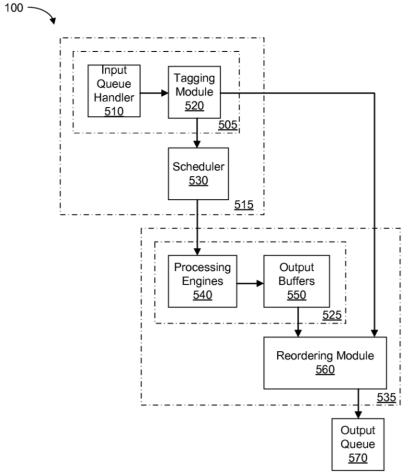
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 7 Dongare ’439 at Figure 7. “In another aspect, the present disclosure provides a non-transitory computer program product having a computer readable program code tangibly embodied therein which, when executed, causes a processor to perform the method for queue ordering in a multi-engine processing system having a plurality of processing engines to process a plurality of input streams and to provide a corresponding plurality of output streams.” Dongare ’439 at col. 2, ll. 20–27. “It is noted that the elements of the multi-engine processing system 100 may be implemented as a software routine, hardware circuit, firmware, or a combination thereof. Furthermore, one or more elements of the multi-engine processing system may be implemented as a single element. For example, the input queue handler 510 and the tagging module may be implemented as one element 505. The scheduler may be combined with the input queue handler 510 and the tagging module may be implemented as one element 515, which handles all the pre-processing steps. The plurality of processing engines 540 along with their respective dedicated output buffers 550 may be implemented as one element 525. The reordering module may be combined with the processing engines and their dedicated buffers an element 535. In some examples, the

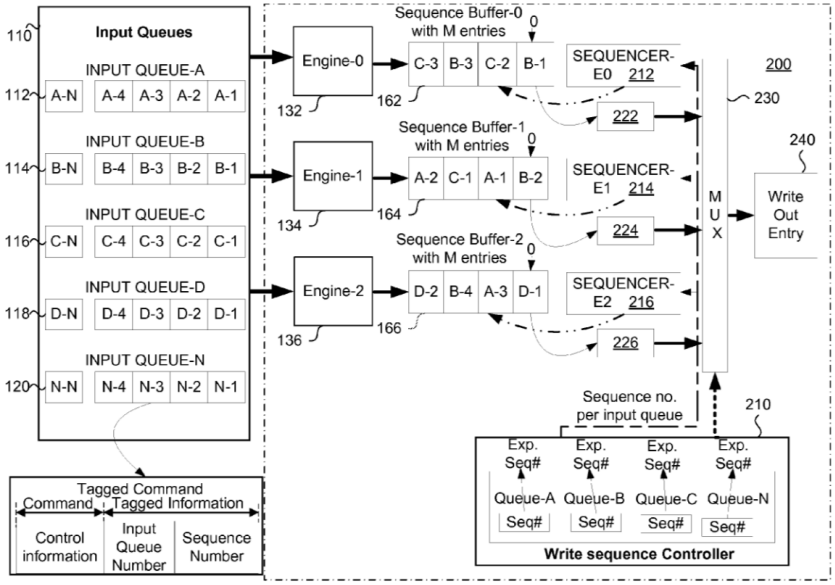
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	output queues 570 and input queues (not shown in FIG. 5) may be implemented as one element. Other combinations of elements are also possible." Dongare '439 at col. 11, ll. 14–30. "FIG. 7 provides an illustration of an application of the apparatus of FIG. 5 in a multi-engine processing system. In applications such as RAID controllers, the Host computer 710 is connected to a RAID ASIC 720 via a PCIe protocol bus. RAID controller ASIC 720 has multiple RAID engines 730 to process data and commands to meet required throughput. The Host 710 and RAID controller 720 communicate to each other by using a set of queues 740 on the RAID controller 720. The queues 740 in the controller are classified as input queues 750 from the Host, and output queues from the RAID engines 730. The Host 710 writes commands for the RAID controller into the input queues 750. The input queues 750 are read by the embedded processor 760 and commands are created for RAID engines 730 to process data. Newly generated commands are loaded to input queues 750 dedicated to RAID engines 730. RAID engines 730 reads commands from input queues 750 and after processing them (e.g., read/write operations from Hard Disks 770), writes the processed commands back to output queues 780. The embedded processor 760 monitors output queues 780 for completion of RAID commands in the same order as was loaded in input queues 750. After completing the Host command, the embedded processor 760 will signal the Host 710 that the command is complete by writing into the outbound queue 780. A write to the outbound queue can generate an interrupt to the Host 710." Dongare '439 at col. 14, ll. 31–55. "Embodiments of the disclosure can be represented as a computer program product stored in a machine-readable medium (also referred to as a computer-readable medium, a processor-readable medium, or a computer usable medium having a computer-readable program code embodied therein). The machine-readable medium can be any suitable tangible, non-transitory medium, including magnetic, optical, or electrical storage medium including a diskette, compact disk read only memory (CD-ROM), memory device (volatile or non-volatile), or similar storage mechanism. The machine-readable medium can contain various sets of instructions, code sequences, configuration information, or other data, which, when executed, cause a processor to perform steps in a method according to an embodiment of the disclosure. Those of ordinary skill in the art will appreciate that other instructions and operations necessary to implement the described implementations can also be stored on the machine-readable medium. The instructions stored on the machine-readable medium can be executed by a processor or other suitable processing device, and can interface with circuitry to perform the described tasks." Dongare '439 at col. 15, ll. 27–47.

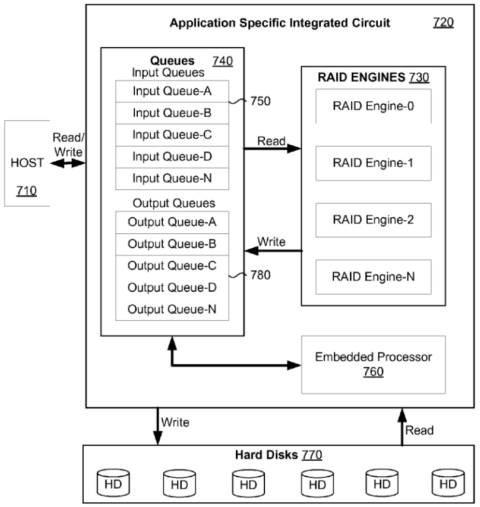
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
[1g] accesses said reservation register and reads said stored value;	
[1h] determines when said read value indicates available space in said input queue for said first processing element to issue a command;	
[1i] notifies said first processing element to issue said command to said input queue; and	Dongare ’439 teaches “notifies said first processing element to issue said command to said input queue,” including at least the following exemplary citations. For example, Dongare ’439 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, signaling of the host processor by the embedded processor when the input queue is available after the input queue was previously full. The embedded processor notifies the first processing element to issue the command to the input queue. *See, e.g.:*

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 7 Dongare ’439 at Figure 7. “FIG. 7 provides an illustration of an application of the apparatus of FIG. 5 in a multi-engine processing system. In applications such as RAID controllers, the Host computer 710 is connected to a RAID ASIC 720 via a PCIe protocol bus. RAID controller ASIC 720 has multiple RAID engines 730 to process data and commands to meet required throughput. The Host 710 and RAID controller 720 communicate to each other by using a set of queues 740 on the RAID controller 720. The queues 740 in the controller are classified as input queues 750 from the Host, and output queues from the RAID engines 730. The Host 710 writes commands for the RAID controller into the input queues 750. The input queues 750 are read by the embedded processor 760 and commands are created for RAID engines 730 to process data. Newly generated commands are loaded to input queues 750 dedicated to RAID engines 730. RAID engines 730 reads commands from input queues 750 and after processing them (e.g., read/write operations from Hard Disks 770), writes the processed commands back to output queues 780. The embedded processor 760 monitors output queues 780 for completion of RAID commands in the same order as was loaded in input queues 750. After completing the Host command, the embedded processor 760 will signal the Host 710 that the command is complete by writing into the outbound queue 780. A write to the outbound queue can generate an interrupt to the Host 710.” Dongare ’439 at col. 14, ll. 31–55.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
[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.	Dongare ’439 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, Dongare ’439 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, a host processor that receives a notification from an embedded processor that the input queue has space available after being previously full, issuing a RAID command to the input queue, and receiving a response to the RAID command from the output queue after the RAID command has been processed. The host processor receives the notification, issues the command, and receives a corresponding response from the output queue. *See, e.g.:*  FIG. 4 Dongare ’439 at Figure 4.

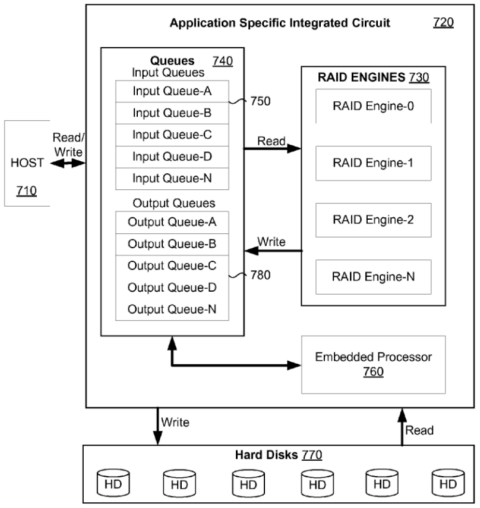
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 5 Dongare ’439 at Figure 5.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 6 Dongare ’439 at Figure 6.

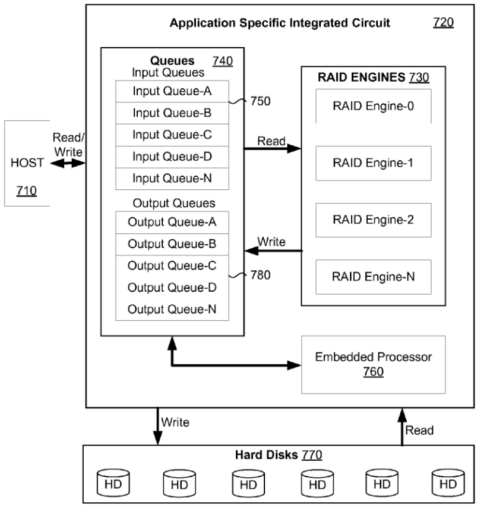
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 7 Dongare ’439 at Figure 7. “Once processed, the processed commands can be written to output queues with a single write channel or multiple write channels. In certain example embodiments, an input queue may also act as an output queue. The processed command may be buffered in one or more output queue buffers and one output queue buffer may hold processed commands from more than one command output queue.” Dongare ’439 at col. 6, ll. 21–27. “In order to maintain the original order or sequence of the commands within a given command input queue, the processed commands must be written out to the command output queues in the same sequence as it was read out from the respective command input queue. It follows that A-1 should be written before A-2, A-2 should be written before A-3 and so on. As shown in FIG. 4, commands are written in the same relative order as in the command input queue: B-1, B-2, D-1, A-1, C-1, C-2, A-2, B-3, A-3, C-3, B-4, and D-2, i.e., the relative order of commands within a command output queue is maintained in the write out sequence irrespective of the processing engine that processed each command. In the example embodiment of FIG. 4, processed commands from command input queue A are written out as: A-1, A-2 and A-3, wherein commands A-1 and A-2 commands were processed by Engine-1 and command A-

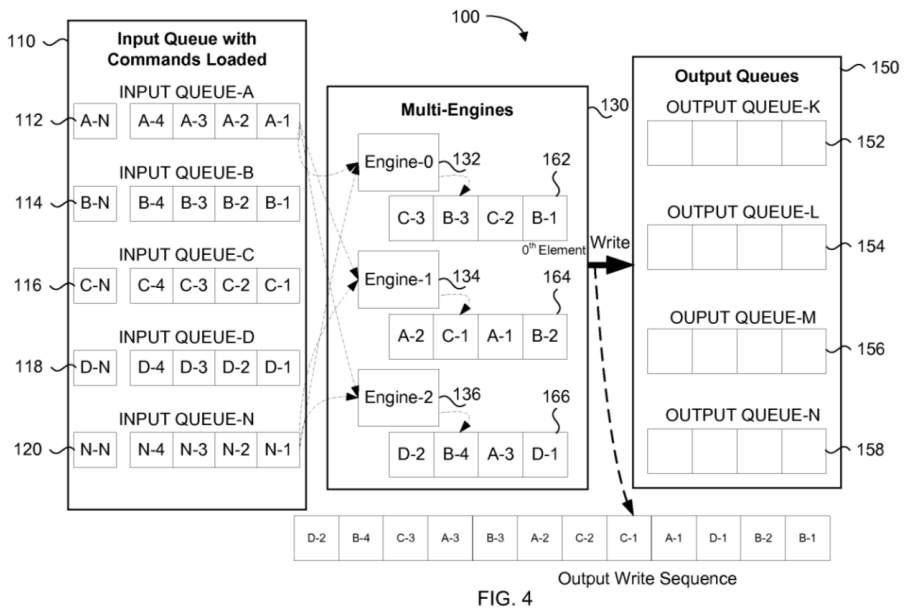
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	3 was processed by Engine-2. In the example embodiment, command A-3 was processed earlier to command A-2 but it was not written out since command A-2 was not processed. Processed command A-3 was stored in the dedicated output buffer 166 of Engine-2136 until command A-2 was processed and ready for write out. Similarly, processed commands from command input queue B are written out as: B-1, B-2, B-3 and B-4, wherein commands B-1 and B-3 were processed by Engine-0, command B-2 was processed by Engine-1 and command B4 was processed by Engine-2. In the example embodiment commands B-1 and B-2 were processed at the same time by different engines and hence could be written out sequentially. However, command B-4 was processed earlier by Engine-2 but it was not written out as command B-3 was not processed. Processed command B-4 was stored in dedicated output buffer 166 of Engine-2136 until command B-3 was processed and ready for write out. Processed commands from command input queue C are written as: C-1, C-2, C-3; wherein command C-1 was processed by Engine-1, commands C-2 and C-3 were processed by Engine-0. In the example embodiment command C-2 was processed earlier than command C-1, but it was not written out until the processing of command C-1 was completed. Processed command C-2 was stored in dedicated output buffer 162 of Engine-0136 until command C-1 was processed and ready for write out. Processed commands from command input queue D are written as: D-1, D-2; wherein commands D-1 and D-2 were processed by same engine sequentially and hence could be written out in that order without any waiting period." Dongare '439 at col. 9, ll. 5–50. "In an aspect, the present disclosure provides a multi-engine processing system comprising: an input queue handler configured to receive a plurality of input streams, each input stream having a plurality of input segments, the plurality of input segments ordered relative to other input segments within the input stream; a tagging module configured to append an identity tag to a plurality of input segments in each input stream to produce a plurality of tagged input segments, the identity tag including information indicating the input stream and the relative order within the input stream of each of the plurality of input segments; a scheduler configured to iteratively assign each of the plurality of tagged input segments to any one of a plurality of processing engines based on availability of the processing engines in the relative order of the tagged input segments within the respective input stream, each of the plurality of tagged input segments being processed in its respective assigned processing engine to produce a corresponding plurality of tagged output segments; a plurality of dedicated output buffers associated with the plurality of processing engines, the plurality of dedicated output buffers configured to store each of the plurality of tagged output segments; and, a reordering module configured to reorder the plurality of tagged output segments to produce a plurality of output streams,

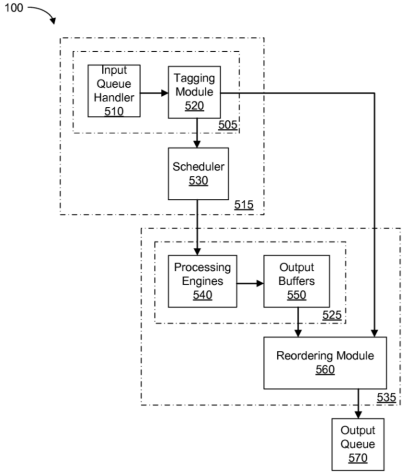
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	each output stream having the same relative order of output segments as in the corresponding input stream." Dongare '439 at col. 9, l. 59 – col. 10, l. 17. "FIG. 7 provides an illustration of an application of the apparatus of FIG. 5 in a multi-engine processing system. In applications such as RAID controllers, the Host computer 710 is connected to a RAID ASIC 720 via a PCIe protocol bus. RAID controller ASIC 720 has multiple RAID engines 730 to process data and commands to meet required throughput. The Host 710 and RAID controller 720 communicate to each other by using a set of queues 740 on the RAID controller 720. The queues 740 in the controller are classified as input queues 750 from the Host, and output queues from the RAID engines 730. The Host 710 writes commands for the RAID controller into the input queues 750. The input queues 750 are read by the embedded processor 760 and commands are created for RAID engines 730 to process data. Newly generated commands are loaded to input queues 750 dedicated to RAID engines 730. RAID engines 730 reads commands from input queues 750 and after processing them (e.g., read/write operations from Hard Disks 770), writes the processed commands back to output queues 780. The embedded processor 760 monitors output queues 780 for completion of RAID commands in the same order as was loaded in input queues 750. After completing the Host command, the embedded processor 760 will signal the Host 710 that the command is complete by writing into the outbound queue 780. A write to the outbound queue can generate an interrupt to the Host 710." Dongare '439 at col. 14, ll. 31–55.
2. The system of claim 1, wherein said input queue is a first-in-first-out (FIFO) buffer.	Dongare '439 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, Dongare '439 teaches wherein said input queue is a first-in-first-out (FIFO) buffer, by disclosing, as non–limiting examples, FIFO buffers. *See, e.g.:* "Several methods have been proposed to reorder output streams resulting from out-of-order multi-processing. U.S. Pat. No. 7,072,342 describes a system and method of reordering out-of-order packets. In this method, while packets may be processed out of order, the processed packets or commands are written into command FIFO buffers of a command queue only when a current sequence number of the command FIFO buffer matches the packet sequence number of the processed packet. Thus, command FIFO buffer holds ordered commands when read out on a FIFO basis. When there are multiple command buffers, each processed command is associated with a lane number and a sequence number and each command FIFO buffer is dedicated to each input lane. Processed commands are written to a particular command FIFO buffer only when its lane number and sequence number matches with lane and sequence number of the processed

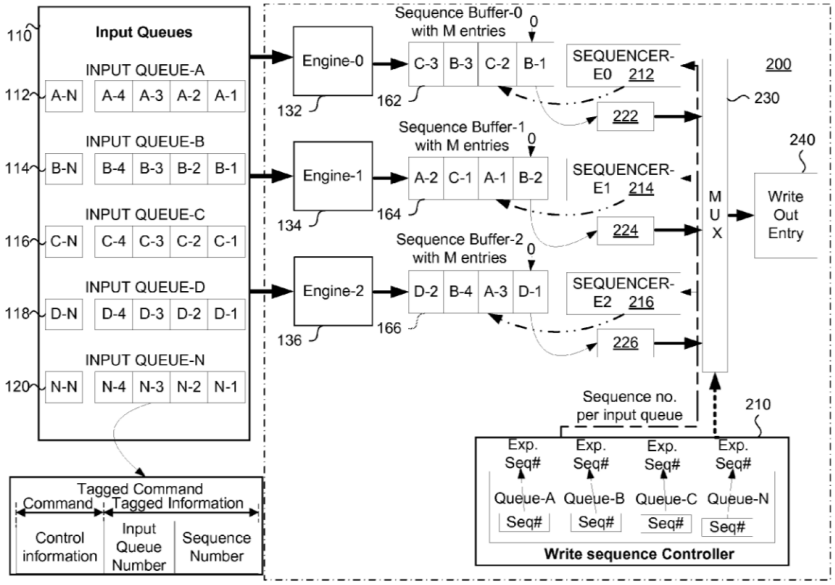
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	command. Consequently, a task has to wait until the sequence number of the command matches the sequence number of the command FIFO buffer, and in effect, one or more tasks remain idle. In addition, the number of command FIFO buffers should be equal to number of input command lanes rendering the system inflexible and rigid." Dongare '439 at col. 2, l. 58 – col. 3, l. 12.
3. The system of claim 1, wherein said system is part of a storage area network (SAN) controller.	Dongare '439 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, Dongare '439 teaches wherein said system is part of a storage area network (SAN) controller, by disclosing, as non–limiting examples, RAID controller ASIC. A RAID Controller ASIC is used by a SAN controller. *See, e.g.:*  FIG. 7 Dongare '439 at Figure 7. "FIG. 7 provides an illustration of an application of the apparatus of FIG. 5 in a multi-engine processing system. In applications such as RAID controllers, the Host computer 710 is connected to a RAID ASIC 720 via a PCIe protocol bus. RAID controller ASIC 720 has multiple RAID engines 730 to process data and commands to meet required throughput. The Host 710 and RAID controller 720 communicate to each other

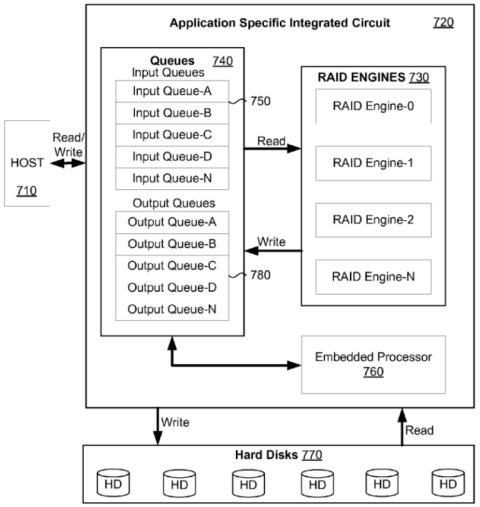
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	by using a set of queues 740 on the RAID controller 720. The queues 740 in the controller are classified as input queues 750 from the Host, and output queues from the RAID engines 730. The Host 710 writes commands for the RAID controller into the input queues 750. The input queues 750 are read by the embedded processor 760 and commands are created for RAID engines 730 to process data. Newly generated commands are loaded to input queues 750 dedicated to RAID engines 730. RAID engines 730 reads commands from input queues 750 and after processing them (e.g., read/write operations from Hard Disks 770), writes the processed commands back to output queues 780. The embedded processor 760 monitors output queues 780 for completion of RAID commands in the same order as was loaded in input queues 750. After completing the Host command, the embedded processor 760 will signal the Host 710 that the command is complete by writing into the outbound queue 780. A write to the outbound queue can generate an interrupt to the Host 710." Dongare '439 at col. 14, ll. 31–55.
4. The system of claim 1, wherein said system is part of a network attached storage (NAS) controller.	Dongare '439 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, Dongare '439 teaches wherein said system is part of a network attached storage (NAS) controller, by disclosing, as non–limiting examples, RAID controller ASIC. A RAID Controller ASIC is used by a NAS controller. *See, e.g.:*

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 7 Dongare ’439 at Figure 7. “FIG. 7 provides an illustration of an application of the apparatus of FIG. 5 in a multi-engine processing system. In applications such as RAID controllers, the Host computer 710 is connected to a RAID ASIC 720 via a PCIe protocol bus. RAID controller ASIC 720 has multiple RAID engines 730 to process data and commands to meet required throughput. The Host 710 and RAID controller 720 communicate to each other by using a set of queues 740 on the RAID controller 720. The queues 740 in the controller are classified as input queues 750 from the Host, and output queues from the RAID engines 730. The Host 710 writes commands for the RAID controller into the input queues 750. The input queues 750 are read by the embedded processor 760 and commands are created for RAID engines 730 to process data. Newly generated commands are loaded to input queues 750 dedicated to RAID engines 730. RAID engines 730 reads commands from input queues 750 and after processing them (e.g., read/write operations from Hard Disks 770), writes the processed commands back to output queues 780. The embedded processor 760 monitors output queues 780 for completion of RAID commands in the same order as was loaded in input queues 750. After completing the Host command, the embedded processor 760 will signal the Host 710 that the command is complete by writing into the outbound queue 780. A write to the outbound queue can generate an interrupt to the Host 710.” Dongare ’439 at col. 14, ll. 31–55.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
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.	
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.	Dongare ’439 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, Dongare ’439 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, RAID engines. The RAID engines execute the command and notify the output queue to issue a response. *See, e.g.:*  FIG. 4 Dongare ’439 at Figure 4.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 5 Dongare ’439 at Figure 5.

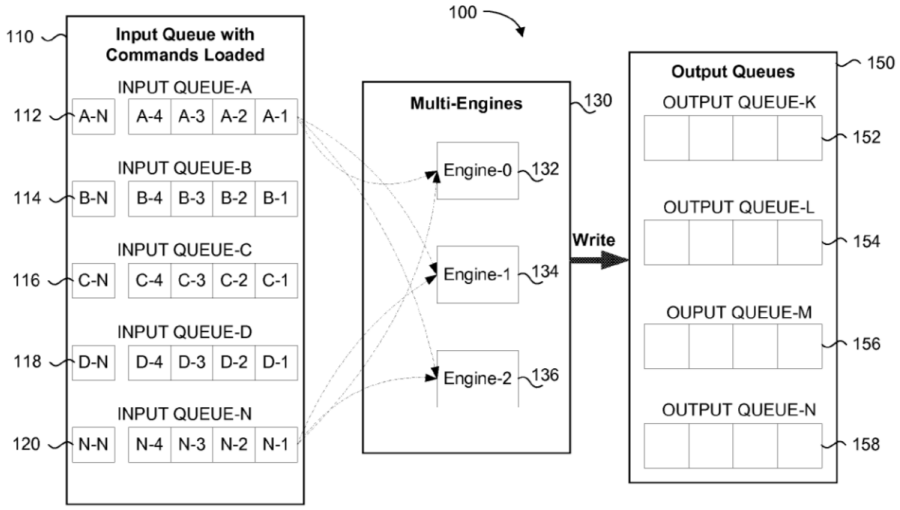
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 6 Dongare ’439 at Figure 6.

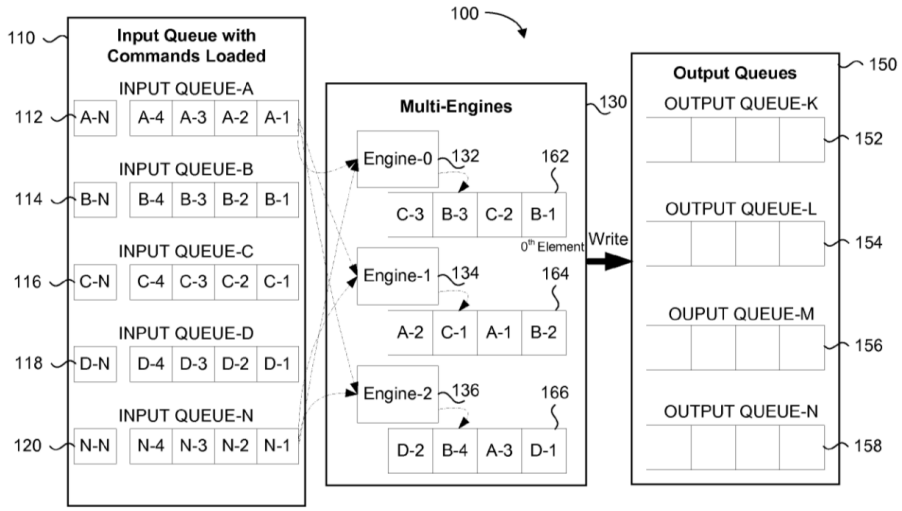
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 7 Dongare ’439 at Figure 7. “In the example shown in FIG. 1, any one of Engine-0132, Engine-1134, or Engine-2136 is capable of processing the commands read out from the input queues. In an example embodiment, all engines may be identical in architecture and functionality or, in other example embodiments, each engine may differ in functionality and/or architecture, but capable of processing any command assigned to it. Commands from same input queue can be processed on multiple engines depending on availability of the engines at any given instance in order to use the engines optimally. The commands from different input queues may be interleaved and assigned (or scheduled) to an available engine provided the relative order of the commands within the respective command input queue is maintained. For example, Engine-0 can process commands B-1, C-2, B-3, and C-3, as shown in FIG. 2, provided commands C-1 and B-2 have been assigned for processing by other engines prior to scheduling commands C-2 and B-3 for processing by Engine-0.” Dongare ’439 at col. 6, ll. 3–20. “Once processed, the processed commands can be written to output queues with a single write channel or multiple write channels. In certain example embodiments, an input queue may also act as an output queue. The processed command may be buffered in one or more output queue buffers and one output queue buffer

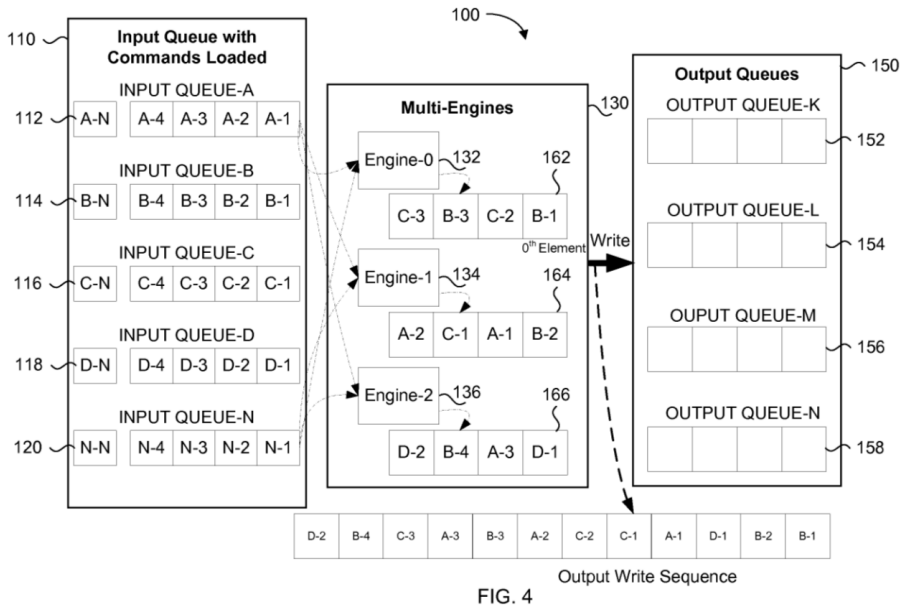
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	may hold processed commands from more than one command output queue." Dongare '439 at col. 6, ll. 21–27. "The scheduling of commands from various command input queues shown in the example embodiment of FIG. 2 with respect to each engine will now be described. Engine-0132 was assigned and has processed commands from command input queues B and C. Commands B-1 and B-3 belong to command input queue B and commands C-2 and C-3 belong to command input queue C. Engine-0132 has processed these commands from different command input queues with sequence order: B-1, C-2, B-3, C-3 respectively and has stored the processed commands in its dedicated output buffer 162. Engine-1134 was assigned and has processed commands from command input queues A, B and C. Commands A-1 and A-2 belong to command input queue A, command B-2 belongs to command input queue B, and command C-1 belongs to command input queue C. Engine-1134 has processed these commands from different command input queues with sequence order: B-2, A-1, C-1, A-2 respectively and has stored the processed commands in its dedicated output buffer 164. Engine-2136 was assigned and has processed commands from command input queues A, B and D. Command A-3 belongs to command input queue A, command B-4 belongs to command input queue B, and commands D-1 and D-2 belong to command input queue D. Engine-2136 has processed these commands from different command input queues with sequence order: D-1, A-3, B-4, D-2 respectively and stored the processed commands in its dedicated output buffer 166." Dongare '439 at col. 7, ll. 6–33. "In order to maintain the original order or sequence of the commands within a given command input queue, the processed commands must be written out to the command output queues in the same sequence as it was read out from the respective command input queue. It follows that A-1 should be written before A-2, A-2 should be written before A-3 and so on. As shown in FIG. 4, commands are written in the same relative order as in the command input queue: B-1, B-2, D-1, A-1, C-1, C-2, A-2, B-3, A-3, C-3, B-4, and D-2, i.e., the relative order of commands within a command output queue is maintained in the write out sequence irrespective of the processing engine that processed each command. In the example embodiment of FIG. 4, processed commands from command input queue A are written out as: A-1, A-2 and A-3, wherein commands A-1 and A-2 commands were processed by Engine-1 and command A-3 was processed by Engine-2. In the example embodiment, command A-3 was processed earlier to command A-2 but it was not written out since command A-2 was not processed. Processed command A-3 was stored in the dedicated output buffer 166 of Engine-2136 until command A-2 was processed and ready for write out.

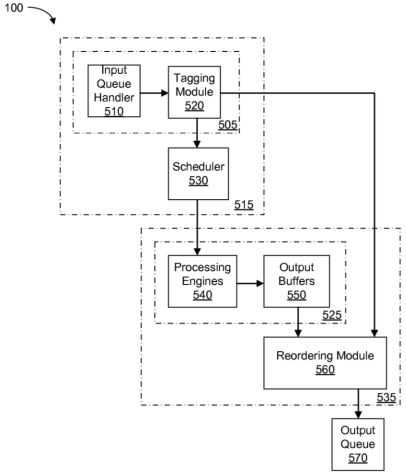
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	Similarly, processed commands from command input queue B are written out as: B-1, B-2, B-3 and B-4, wherein commands B-1 and B-3 were processed by Engine-0, command B-2 was processed by Engine-1 and command B4 was processed by Engine-2. In the example embodiment commands B-1 and B-2 were processed at the same time by different engines and hence could be written out sequentially. However, command B-4 was processed earlier by Engine-2 but it was not written out as command B-3 was not processed. Processed command B-4 was stored in dedicated output buffer 166 of Engine-2136 until command B-3 was processed and ready for write out. Processed commands from command input queue C are written as: C-1, C-2, C-3; wherein command C-1 was processed by Engine-1, commands C-2 and C-3 were processed by Engine-0. In the example embodiment command C-2 was processed earlier than command C-1, but it was not written out until the processing of command C-1 was completed. Processed command C-2 was stored in dedicated output buffer 162 of Engine-0136 until command C-1 was processed and ready for write out. Processed commands from command input queue D are written as: D-1, D-2; wherein commands D-1 and D-2 were processed by same engine sequentially and hence could be written out in that order without any waiting period." Dongare '439 at col. 9, ll. 5–50. "In an aspect, the present disclosure provides a multi-engine processing system comprising: an input queue handler configured to receive a plurality of input streams, each input stream having a plurality of input segments, the plurality of input segments ordered relative to other input segments within the input stream; a tagging module configured to append an identity tag to a plurality of input segments in each input stream to produce a plurality of tagged input segments, the identity tag including information indicating the input stream and the relative order within the input stream of each of the plurality of input segments; a scheduler configured to iteratively assign each of the plurality of tagged input segments to any one of a plurality of processing engines based on availability of the processing engines in the relative order of the tagged input segments within the respective input stream, each of the plurality of tagged input segments being processed in its respective assigned processing engine to produce a corresponding plurality of tagged output segments; a plurality of dedicated output buffers associated with the plurality of processing engines, the plurality of dedicated output buffers configured to store each of the plurality of tagged output segments; and, a reordering module configured to reorder the plurality of tagged output segments to produce a plurality of output streams, each output stream having the same relative order of output segments as in the corresponding input stream." Dongare '439 at col. 9, l. 59 – col. 10, l. 17. "FIG. 7 provides an illustration of an application of the apparatus of FIG. 5 in a multi-engine processing system. In applications such as RAID controllers, the Host computer 710 is connected to a RAID ASIC 720 via a PCIe protocol bus. RAID controller ASIC 720 has multiple RAID engines 730 to process data and

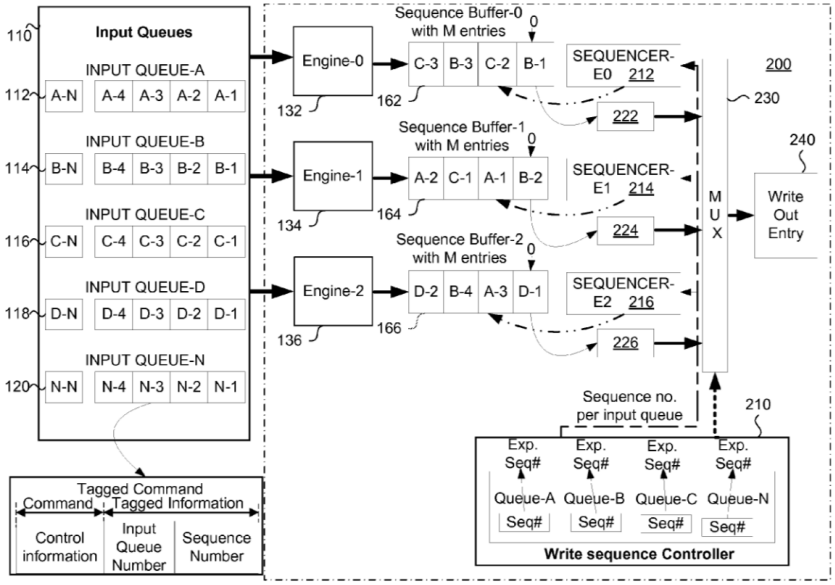
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	commands to meet required throughput. The Host 710 and RAID controller 720 communicate to each other by using a set of queues 740 on the RAID controller 720. The queues 740 in the controller are classified as input queues 750 from the Host, and output queues from the RAID engines 730. The Host 710 writes commands for the RAID controller into the input queues 750. The input queues 750 are read by the embedded processor 760 and commands are created for RAID engines 730 to process data. Newly generated commands are loaded to input queues 750 dedicated to RAID engines 730. RAID engines 730 reads commands from input queues 750 and after processing them (e.g., read/write operations from Hard Disks 770), writes the processed commands back to output queues 780. The embedded processor 760 monitors output queues 780 for completion of RAID commands in the same order as was loaded in input queues 750. After completing the Host command, the embedded processor 760 will signal the Host 710 that the command is complete by writing into the outbound queue 780. A write to the outbound queue can generate an interrupt to the Host 710." Dongare '439 at col. 14, ll. 31–55.
[7pre] A system for dividing and synchronizing a processing task across a plurality of processing elements comprising:	Dongare '439 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;	Dongare '439 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;	Dongare '439 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;	Dongare '439 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].

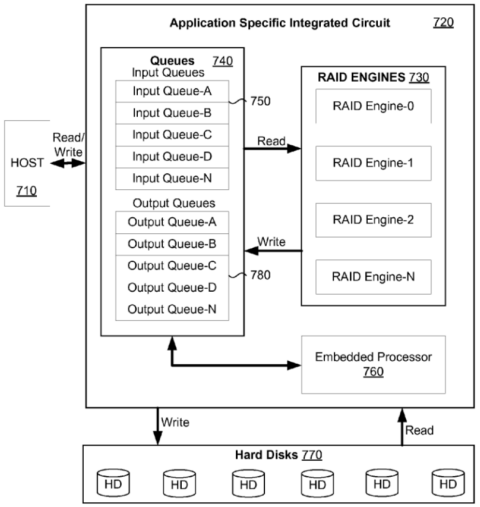
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
[7d] at least one second processing element in communication with said input queue;	Dongare ’439 teaches “at least one second processing element in communication with said input queue,” including at least the following exemplary citations. For example, Dongare ’439 teaches at least one second processing element in communication with said input queue, by disclosing, as non–limiting examples, an embedded processor that communicates with the input queue. The embedded processor, the second processing element is in communication with the input queue. *See, e.g.:* “A method and apparatus for queue-ordering commands in multi-engines, multi-queues and/or multi-flows environment is provided. Commands from single/multiple queues and multi-flows are processed by multi-engines with different processing time and/or out of order, which breaks sequential order of commands from same input queue and commands are distributed across multiple engines' output buffer after processing. Processed commands are stored in dedicated command output buffer associated with each engine temporarily. The processed commands are re-ordered while writing out. Also commands can be scheduled to idle engines to achieve maximum throughput, thus utilizing the engines in an optimal manner.” Dongare ’439 at Abstract.  FIG. 1 Dongare ’439 at Figure 1.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 2 Dongare ’439 at Figure 2.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 4 Dongare ’439 at Figure 4.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 5 Dongare ’439 at Figure 5.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 6 Dongare ’439 at Figure 6.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 7 Dongare ’439 at Figure 7. “In a first aspect, the present disclosure provides a method for queue ordering in a multi-engine processing system having a plurality of processing engines to process a plurality of input streams and to provide a corresponding plurality of output streams. The method comprises: receiving the plurality of input streams, each input stream having a plurality of input segments, the plurality of input segments ordered relative to other input segments within the input stream; appending an identity tag to the plurality of input segments in each input stream to produce a plurality of tagged input segments, the identity tag including information indicating the input stream and the relative order within the input stream of each of the plurality of input segments; iteratively assigning each of the plurality of tagged input segments to any one of the plurality of processing engines based on availability of the processing engines in the relative order of the tagged input segments within the respective input stream; processing each of the plurality of tagged input segments in its respective assigned processing engine to produce a corresponding plurality of tagged output segments; storing each of the plurality of tagged output segments in a dedicated output buffer associated with its respective assigned processing engine; and, reordering the plurality of tagged output segments to produce the plurality of output streams, each output stream having the same relative order as in the corresponding input stream.” Dongare ’439 at col. 1, ll. 36–60.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	"In another aspect, the present disclosure provides a multi-engine processing system comprising: an input queue handler configured to receive a plurality of input streams, each input stream having a plurality of input segments, the plurality of input segments ordered relative to other input segments within the input stream; a tagging module configured to append an identity tag to a plurality of input segments in each input stream to produce a plurality of tagged input segments, the identity tag including information indicating the input stream and the relative order within the input stream of each of the plurality of input segments; a scheduler configured to iteratively assign each of the plurality of tagged input segments to any one of a plurality of processing engines based on availability of the processing engines in the relative order of the tagged input segments within the respective input stream, each of the plurality of tagged input segments being processed in its respective assigned processing engine to produce a corresponding plurality of tagged output segments; a plurality of dedicated output buffers associated with the plurality of processing engines, the plurality of dedicated output buffers configured to store each of the plurality of tagged output segments; and, a reordering module configured to reorder the plurality of tagged output segments to produce a plurality of output streams, each output stream having the same relative order of output segments as in the corresponding input stream." Dongare '439 at col. 1, l. 61 – col. 2, l. 19. "Generally, the present disclosure provides a method and apparatus for queue ordering in a multi-processing system. The method and apparatus can be implemented in multi-engine, multi-queue, and/or multi-flow environments, such as in multi-engine processor systems or in a network domain. In a multi-engine processor or a multi-threading system, tasks from same program can be executed concomitantly on different processors to achieve maximum throughput. However, the completed task has to be provided in the order it was posted to the multi-processor system. The method and apparatus of the present disclosure provide for reordering completed tasks efficiently." Dongare '439 at col. 4, ll. 44–55. "FIG. 1 illustrates a multi-engine processing system 100 capable of handling a plurality of input queues or input streams 110. The input queues are sequentially ordered commands or segments to be processed by the plurality of engines 130. The processed commands are written out to the plurality of output queues 150. In the example shown in FIG. 1, the plurality of input queues are received and stored in input queue buffer-A 112, input queue buffer-B 114, input queue buffer-C 116, input queue buffer-D 118, and input queue buffer-N 120. The plurality of processing engines 130 includes Engine-0132, Engine-1134, and Engine-2136. The plurality of output queues 150 are written out and stored in output queue buffer-K 152, output queue buffer-L 154, output queue buffer-M 156, and output queue buffer-N 158. Thus, command streams A, B, C, D through N

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	can be received and stored in the respective input queue buffers and the processed commands can be written out and stored in output queue buffers. As shown in FIG. 1, the input queue buffer-A 112 has commands A-1, A-2, . . . A-N, input queue buffer-B 114 has commands B-1, B-2, . . . B-N, input queue buffer-C 116 has commands C-1, C-2, . . . C-N, input queue buffer-D 118 has commands D-1, D-2, . . . D-N, and input queue buffer-N 120 has commands N-1, N-2, . . . N-N." Dongare '439 at col. 5, ll. 3–26. "Commands read out from the input queues can be directly scheduled or assigned to any one of the plurality of the engines for processing based on the availability of the processing engine. In other example embodiments, read out commands can be stored in a shared buffer for pre-processing or filtering before scheduling it to a free engine for processing. When commands are stored in shared buffer, they can be provided to an idle engine immediately without fetching." Dongare '439 at col. 5, l. 62 – col. 6, l. 2. "In the example shown in FIG. 1, any one of Engine-0132, Engine-1134, or Engine-2136 is capable of processing the commands read out from the input queues. In an example embodiment, all engines may be identical in architecture and functionality or, in other example embodiments, each engine may differ in functionality and/or architecture, but capable of processing any command assigned to it. Commands from same input queue can be processed on multiple engines depending on availability of the engines at any given instance in order to use the engines optimally. The commands from different input queues may be interleaved and assigned (or scheduled) to an available engine provided the relative order of the commands within the respective command input queue is maintained. For example, Engine-0 can process commands B-1, C-2, B-3, and C-3, as shown in FIG. 2, provided commands C-1 and B-2 have been assigned for processing by other engines prior to scheduling commands C-2 and B-3 for processing by Engine-0." Dongare '439 at col. 6, ll. 3–20. "Once processed, the processed commands can be written to output queues with a single write channel or multiple write channels. In certain example embodiments, an input queue may also act as an output queue. The processed command may be buffered in one or more output queue buffers and one output queue buffer may hold processed commands from more than one command output queue." Dongare '439 at col. 6, ll. 21–27. "In the example embodiment shown in FIG. 2, the three engines are identical and are capable of processing any command assigned to them. However, depending on the nature of a command being processed, for example, the byte count and functionality of the command, a command provided earlier to one engine may take more time to process than another command assigned to other engine. Thus, processed commands would

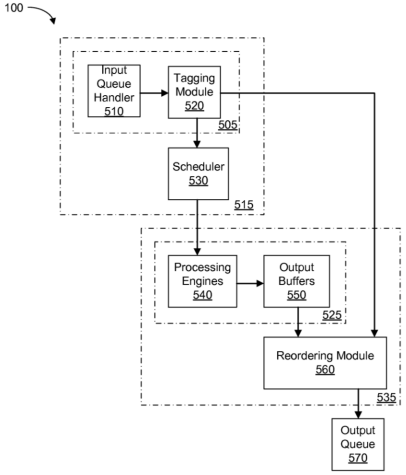
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	be ready to write out at different time points. Also, the sequence of commands from the same input queue would be disturbed at the output side and may be distributed across the dedicated buffers of the respective engines. As shown in FIG. 2, each engine has a dedicated output buffer. For example, Engine-0132 has a dedicated output buffer 162, Engine-1134 has a dedicated output buffer 164, and Engine-2136 has a dedicated output buffer 166." Dongare '439 at col. 6, ll. 28–42. "The scheduling of commands from various command input queues shown in the example embodiment of FIG. 2 with respect to each engine will now be described. Engine-0132 was assigned and has processed commands from command input queues B and C. Commands B-1 and B-3 belong to command input queue B and commands C-2 and C-3 belong to command input queue C. Engine-0132 has processed these commands from different command input queues with sequence order: B-1, C-2, B-3, C-3 respectively and has stored the processed commands in its dedicated output buffer 162. Engine-1134 was assigned and has processed commands from command input queues A, B and C. Commands A-1 and A-2 belong to command input queue A, command B-2 belongs to command input queue B, and command C-1 belongs to command input queue C. Engine-1134 has processed these commands from different command input queues with sequence order: B-2, A-1, C-1, A-2 respectively and has stored the processed commands in its dedicated output buffer 164. Engine-2136 was assigned and has processed commands from command input queues A, B and D. Command A-3 belongs to command input queue A, command B-4 belongs to command input queue B, and commands D-1 and D-2 belong to command input queue D. Engine-2136 has processed these commands from different command input queues with sequence order: D-1, A-3, B-4, D-2 respectively and stored the processed commands in its dedicated output buffer 166." Dongare '439 at col. 7, ll. 6–33. "In an example embodiment, the method further comprises storing the plurality of tagged input segments in a shared buffer for pre-processing or filtering prior to assigning each of the plurality of tagged input segments to any one of the plurality of processing engines." Dongare '439 at col. 8, ll. 41–45. "FIG. 4 illustrates the required queue ordering while writing out the processed commands to the output queues, each output queue having the same relative sequence as in the corresponding input queue. As described earlier, the plurality of output queues 150 are written out and stored in output queue buffer-K 152, output queue buffer-L 154, output queue buffer-M 156, and output queue buffer-N 158. The various command input queues A-N are iteratively scheduled or assigned to be processed on any one of the plurality of processing engines based on the availability of the processing engines in the relative order if the commands

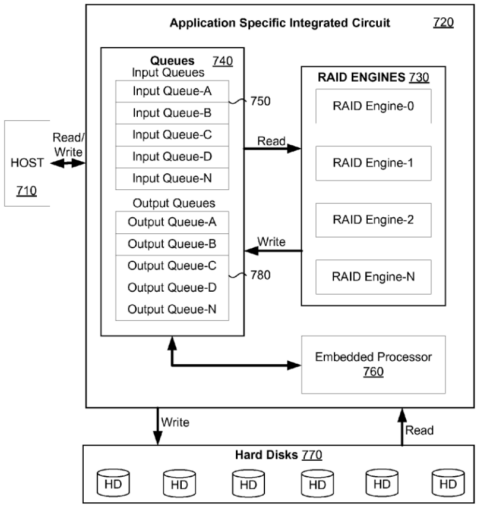
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	within their respective command input queue as discussed earlier with respect to the example embodiment of FIG. 2." Dongare '439 at col. 8, ll. 51–63. "In order to maintain the original order or sequence of the commands within a given command input queue, the processed commands must be written out to the command output queues in the same sequence as it was read out from the respective command input queue. It follows that A-1 should be written before A-2, A-2 should be written before A-3 and so on. As shown in FIG. 4, commands are written in the same relative order as in the command input queue: B-1, B-2, D-1, A-1, C-1, C-2, A-2, B-3, A-3, C-3, B-4, and D-2, i.e., the relative order of commands within a command output queue is maintained in the write out sequence irrespective of the processing engine that processed each command. In the example embodiment of FIG. 4, processed commands from command input queue A are written out as: A-1, A-2 and A-3, wherein commands A-1 and A-2 commands were processed by Engine-1 and command A-3 was processed by Engine-2. In the example embodiment, command A-3 was processed earlier to command A-2 but it was not written out since command A-2 was not processed. Processed command A-3 was stored in the dedicated output buffer 166 of Engine-2136 until command A-2 was processed and ready for write out. Similarly, processed commands from command input queue B are written out as: B-1, B-2, B-3 and B-4, wherein commands B-1 and B-3 were processed by Engine-0, command B-2 was processed by Engine-1 and command B4 was processed by Engine-2. In the example embodiment commands B-1 and B-2 were processed at the same time by different engines and hence could be written out sequentially. However, command B-4 was processed earlier by Engine-2 but it was not written out as command B-3 was not processed. Processed command B-4 was stored in dedicated output buffer 166 of Engine-2136 until command B-3 was processed and ready for write out. Processed commands from command input queue C are written as: C-1, C-2, C-3; wherein command C-1 was processed by Engine-1, commands C-2 and C-3 were processed by Engine-0. In the example embodiment command C-2 was processed earlier than command C-1, but it was not written out until the processing of command C-1 was completed. Processed command C-2 was stored in dedicated output buffer 162 of Engine-0136 until command C-1 was processed and ready for write out. Processed commands from command input queue D are written as: D-1, D-2; wherein commands D-1 and D-2 were processed by same engine sequentially and hence could be written out in that order without any waiting period." Dongare '439 at col. 9, ll. 5–50. "In an aspect, the present disclosure provides a multi-engine processing system comprising: an input queue handler configured to receive a plurality of input streams, each input stream having a plurality of input segments, the plurality of input segments ordered relative to other input segments within the input stream; a

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	tagging module configured to append an identity tag to a plurality of input segments in each input stream to produce a plurality of tagged input segments, the identity tag including information indicating the input stream and the relative order within the input stream of each of the plurality of input segments; a scheduler configured to iteratively assign each of the plurality of tagged input segments to any one of a plurality of processing engines based on availability of the processing engines in the relative order of the tagged input segments within the respective input stream, each of the plurality of tagged input segments being processed in its respective assigned processing engine to produce a corresponding plurality of tagged output segments; a plurality of dedicated output buffers associated with the plurality of processing engines, the plurality of dedicated output buffers configured to store each of the plurality of tagged output segments; and, a reordering module configured to reorder the plurality of tagged output segments to produce a plurality of output streams, each output stream having the same relative order of output segments as in the corresponding input stream." Dongare '439 at col. 9, l. 59 – col. 10, l. 17. "FIG. 5 is a schematic representation of a multi-engine processing system in accordance with this disclosure. The multi-engine processing system 100 comprises an input queue handler 510; a tagging module 520; a scheduler 530; a plurality of processing engines 540; a plurality of output buffers 550; a reordering module 560; and an output queue 570. The input queue handler 510 is configured to receive a plurality of input streams. Each input stream has a plurality of input segments and the plurality of input segments ordered relative to other input segments within the input stream. The tagging module 520 is configured to append an identity tag to a plurality of input segments in each input stream to produce a plurality of tagged input segments. The identity tag includes information indicating the input stream and the relative order within the input stream of each of the plurality of input segments. The scheduler 530 is configured to iteratively assign each of the plurality of tagged input segments to any one of a plurality of processing engines 540 based on availability of the processing engines 540 in the relative order of the tagged input segments within the respective input stream. Each of the plurality of tagged input segments is processed in its respective assigned processing engine 540 to produce a corresponding plurality of tagged output segments. Each of the plurality of processing engines 540 has a corresponding plurality of dedicated output buffers 550. The plurality of dedicated output buffers 550 are configured to store each of the plurality of tagged output segments. The reordering module 560 is configured to reorder the plurality of tagged output segments to produce a plurality of output queues or streams 570. Each output stream has the same relative order of output segments as in the corresponding input stream." Dongare '439 at col. 10, l. 50 – col. 11, l. 13. "FIG. 6 illustrates an example embodiment of an apparatus 200 for queue ordering in a multi-engine processing environment. Commands in the various command input queues are tagged by appending unique

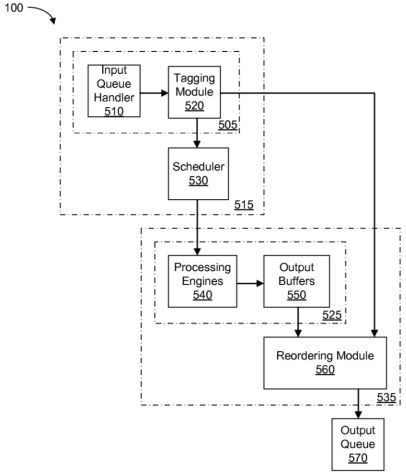
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	information or identity tags indicating the command input queue identification number (queue number) and its position (sequence number) within the respective command input queue. The identity tags may be appended by tagging module 520 while posting commands in the input queues. In other embodiments, commands may be tagged by the tagging mechanism after reading it from input queues and prior to scheduling the commands for processing by the various engines. Commands from command input queues are processed by an engine of the multi-engine processing system in any order resulting in out-of-order processed commands as described with respect to FIG. 4. Processed commands are stored in dedicated output command buffers of the respective engine that processed the command. Since commands from the same command input queue can be processed by multiple engines in an out-of-order sequence and due to the variable time taken by the engines to process assigned/scheduled commands, the processed commands may be distributed across multiple output buffers." Dongare '439 at col. 11, ll. 31–53. "FIG. 7 provides an illustration of an application of the apparatus of FIG. 5 in a multi-engine processing system. In applications such as RAID controllers, the Host computer 710 is connected to a RAID ASIC 720 via a PCIe protocol bus. RAID controller ASIC 720 has multiple RAID engines 730 to process data and commands to meet required throughput. The Host 710 and RAID controller 720 communicate to each other by using a set of queues 740 on the RAID controller 720. The queues 740 in the controller are classified as input queues 750 from the Host, and output queues from the RAID engines 730. The Host 710 writes commands for the RAID controller into the input queues 750. The input queues 750 are read by the embedded processor 760 and commands are created for RAID engines 730 to process data. Newly generated commands are loaded to input queues 750 dedicated to RAID engines 730. RAID engines 730 reads commands from input queues 750 and after processing them (e.g., read/write operations from Hard Disks 770), writes the processed commands back to output queues 780. The embedded processor 760 monitors output queues 780 for completion of RAID commands in the same order as was loaded in input queues 750. After completing the Host command, the embedded processor 760 will signal the Host 710 that the command is complete by writing into the outbound queue 780. A write to the outbound queue can generate an interrupt to the Host 710." Dongare '439 at col. 14, ll. 31–55.
[7e] a reservation register implemented in hardware storing a value indicative of available space in said input queue, said	

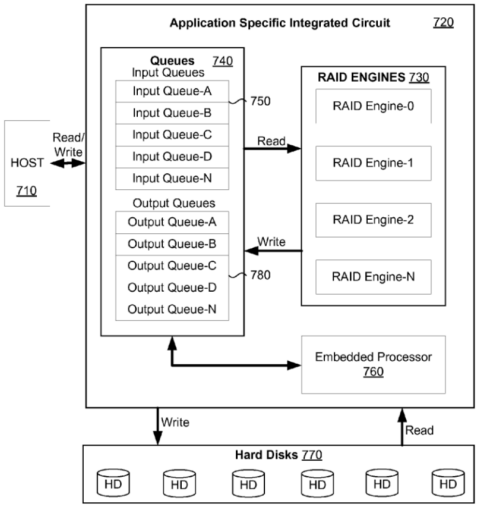
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
reservation register accessible by both said first processing element and said at least one second processing element;	
[7f] computer storage storing instructions, which when executed by said at least one second processing element:	Dongare ’439 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].
[7g] receives a command from said first processing element;	Dongare ’439 teaches “receives a command from said first processing element,” including at least the following exemplary citations. For example, Dongare ’439 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, an embedded processor that receives a RAID command from a host processor via the input queue. The embedded processor, the second processing element receives a command from the first processing element. *See, e.g.:*

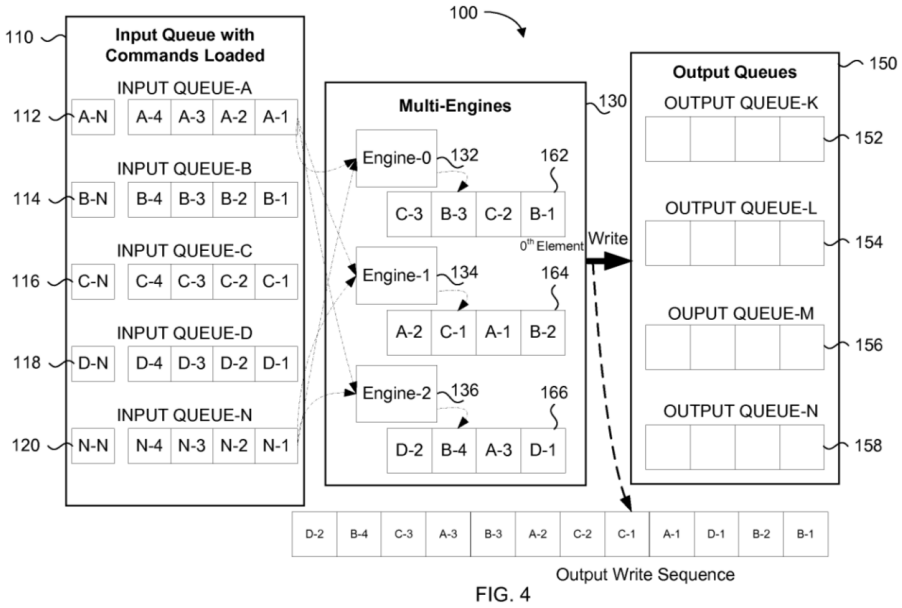
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 5 Dongare ’439 at Figure 5.

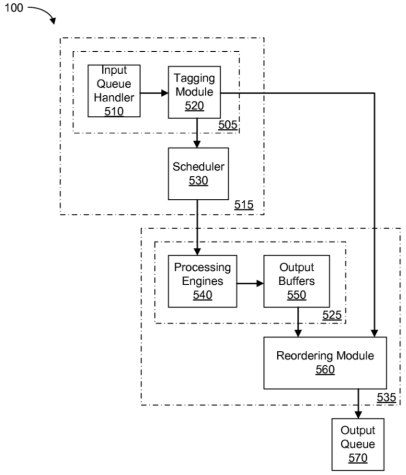
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 7 Dongare ’439 at Figure 7. “FIG. 7 provides an illustration of an application of the apparatus of FIG. 5 in a multi-engine processing system. In applications such as RAID controllers, the Host computer 710 is connected to a RAID ASIC 720 via a PCIe protocol bus. RAID controller ASIC 720 has multiple RAID engines 730 to process data and commands to meet required throughput. The Host 710 and RAID controller 720 communicate to each other by using a set of queues 740 on the RAID controller 720. The queues 740 in the controller are classified as input queues 750 from the Host, and output queues from the RAID engines 730. The Host 710 writes commands for the RAID controller into the input queues 750. The input queues 750 are read by the embedded processor 760 and commands are created for RAID engines 730 to process data. Newly generated commands are loaded to input queues 750 dedicated to RAID engines 730. RAID engines 730 reads commands from input queues 750 and after processing them (e.g., read/write operations from Hard Disks 770), writes the processed commands back to output queues 780. The embedded processor 760 monitors output queues 780 for completion of RAID commands in the same order as was loaded in input queues 750. After completing the Host command, the embedded processor 760 will signal the Host 710 that the command is complete by writing into the outbound queue 780. A write to the outbound queue can generate an interrupt to the Host 710.” Dongare ’439 at col. 14, ll. 31–55.

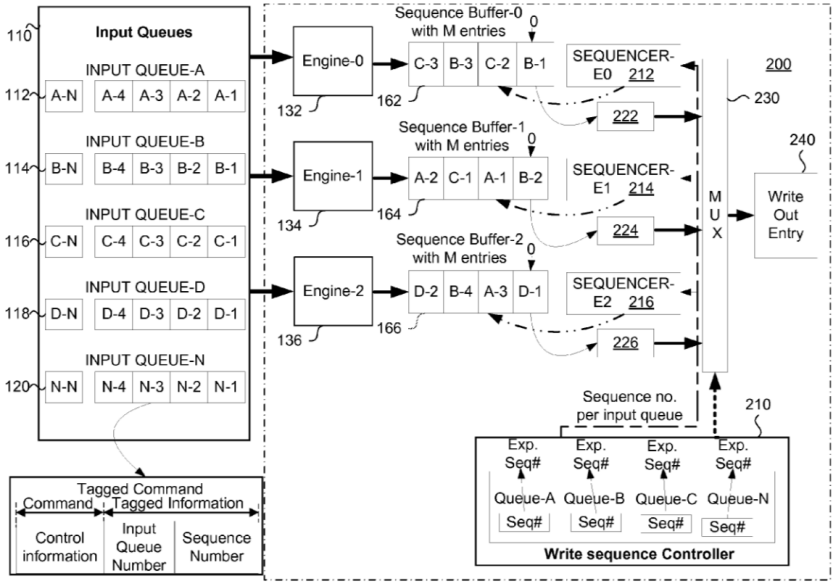
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
[7h] accesses said reservation register and reads said stored value;	
[7i] determines when said read value indicates available space in said input queue to issue said command;	
[7j] issues said command to said input queue on behalf of said first processing element; and	Dongare ’439 teaches “issues said command to said input queue on behalf of said first processing element,” including at least the following exemplary citations. For example, Dongare ’439 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, an embedded processor that divides the host command into portions, tags the portions, and issues the command including all the portions to the input queue. The embedded processor issues the command to the input queue on behalf of the first processing element. *See, e.g.:*

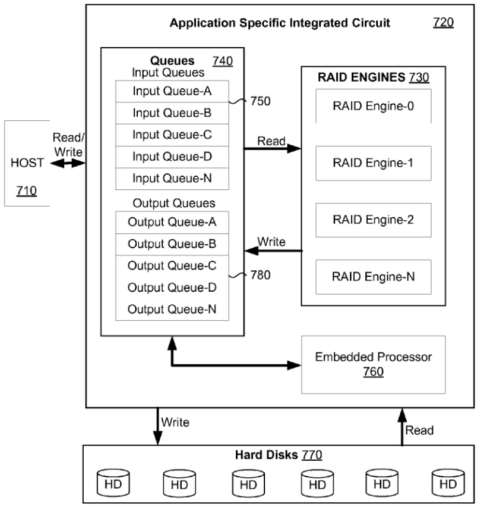
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 5 Dongare ’439 at Figure 5.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 7 Dongare ’439 at Figure 7. “FIG. 7 provides an illustration of an application of the apparatus of FIG. 5 in a multi-engine processing system. In applications such as RAID controllers, the Host computer 710 is connected to a RAID ASIC 720 via a PCIe protocol bus. RAID controller ASIC 720 has multiple RAID engines 730 to process data and commands to meet required throughput. The Host 710 and RAID controller 720 communicate to each other by using a set of queues 740 on the RAID controller 720. The queues 740 in the controller are classified as input queues 750 from the Host, and output queues from the RAID engines 730. The Host 710 writes commands for the RAID controller into the input queues 750. The input queues 750 are read by the embedded processor 760 and commands are created for RAID engines 730 to process data. Newly generated commands are loaded to input queues 750 dedicated to RAID engines 730. RAID engines 730 reads commands from input queues 750 and after processing them (e.g., read/write operations from Hard Disks 770), writes the processed commands back to output queues 780. The embedded processor 760 monitors output queues 780 for completion of RAID commands in the same order as was loaded in input queues 750. After completing the Host command, the embedded processor 760 will signal the Host 710 that the command is complete by writing into the outbound queue 780. A write to the outbound queue can generate an interrupt to the Host 710.” Dongare ’439 at col. 14, ll. 31–55.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
[7k] wherein said first processing element receives a response corresponding to said command from said output queue.	Dongare ’439 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, Dongare ’439 teaches wherein said first processing element receives a response corresponding to said command from said output queue, by disclosing, as non–limiting examples, receiving a response to the RAID command from the output queue after the RAID command has been processed. The host processor receives a response from the output queue corresponding to the command. *See, e.g.:*  FIG. 4 Dongare ’439 at Figure 4.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 5 Dongare ’439 at Figure 5.

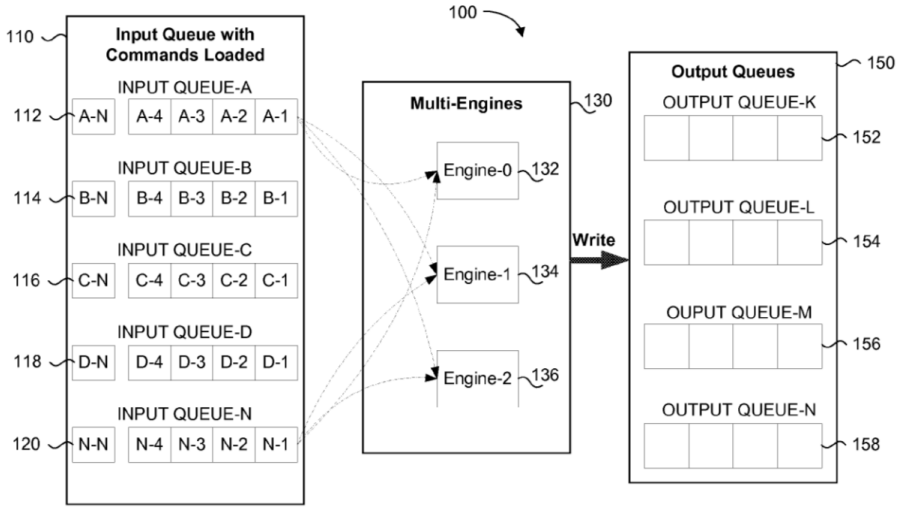
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 6 Dongare ’439 at Figure 6.

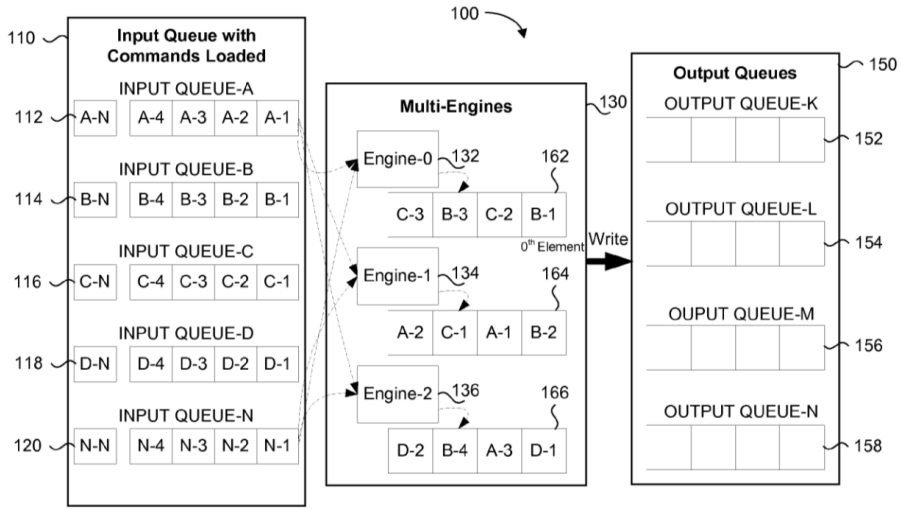
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 7 Dongare ’439 at Figure 7. “Once processed, the processed commands can be written to output queues with a single write channel or multiple write channels. In certain example embodiments, an input queue may also act as an output queue. The processed command may be buffered in one or more output queue buffers and one output queue buffer may hold processed commands from more than one command output queue.” Dongare ’439 at col. 6, ll. 21–27. “In order to maintain the original order or sequence of the commands within a given command input queue, the processed commands must be written out to the command output queues in the same sequence as it was read out from the respective command input queue. It follows that A-1 should be written before A-2, A-2 should be written before A-3 and so on. As shown in FIG. 4, commands are written in the same relative order as in the command input queue: B-1, B-2, D-1, A-1, C-1, C-2, A-2, B-3, A-3, C-3, B-4, and D-2, i.e., the relative order of commands within a command output queue is maintained in the write out sequence irrespective of the processing engine that processed each command. In the example embodiment of FIG. 4, processed commands from command input queue A are written out as: A-1, A-2 and A-3, wherein commands A-1 and A-2 commands were processed by Engine-1 and command A-

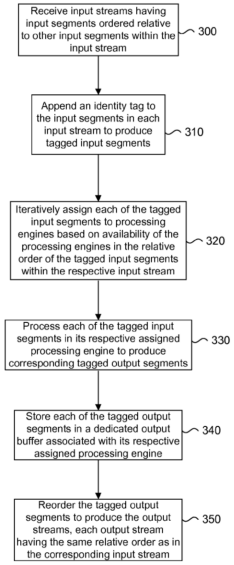
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	3 was processed by Engine-2. In the example embodiment, command A-3 was processed earlier to command A-2 but it was not written out since command A-2 was not processed. Processed command A-3 was stored in the dedicated output buffer 166 of Engine-2136 until command A-2 was processed and ready for write out. Similarly, processed commands from command input queue B are written out as: B-1, B-2, B-3 and B-4, wherein commands B-1 and B-3 were processed by Engine-0, command B-2 was processed by Engine-1 and command B4 was processed by Engine-2. In the example embodiment commands B-1 and B-2 were processed at the same time by different engines and hence could be written out sequentially. However, command B-4 was processed earlier by Engine-2 but it was not written out as command B-3 was not processed. Processed command B-4 was stored in dedicated output buffer 166 of Engine-2136 until command B-3 was processed and ready for write out. Processed commands from command input queue C are written as: C-1, C-2, C-3; wherein command C-1 was processed by Engine-1, commands C-2 and C-3 were processed by Engine-0. In the example embodiment command C-2 was processed earlier than command C-1, but it was not written out until the processing of command C-1 was completed. Processed command C-2 was stored in dedicated output buffer 162 of Engine-0136 until command C-1 was processed and ready for write out. Processed commands from command input queue D are written as: D-1, D-2; wherein commands D-1 and D-2 were processed by same engine sequentially and hence could be written out in that order without any waiting period." Dongare '439 at col. 9, ll. 5–50. "In an aspect, the present disclosure provides a multi-engine processing system comprising: an input queue handler configured to receive a plurality of input streams, each input stream having a plurality of input segments, the plurality of input segments ordered relative to other input segments within the input stream; a tagging module configured to append an identity tag to a plurality of input segments in each input stream to produce a plurality of tagged input segments, the identity tag including information indicating the input stream and the relative order within the input stream of each of the plurality of input segments; a scheduler configured to iteratively assign each of the plurality of tagged input segments to any one of a plurality of processing engines based on availability of the processing engines in the relative order of the tagged input segments within the respective input stream, each of the plurality of tagged input segments being processed in its respective assigned processing engine to produce a corresponding plurality of tagged output segments; a plurality of dedicated output buffers associated with the plurality of processing engines, the plurality of dedicated output buffers configured to store each of the plurality of tagged output segments; and, a reordering module configured to reorder the plurality of tagged output segments to produce a plurality of output streams,

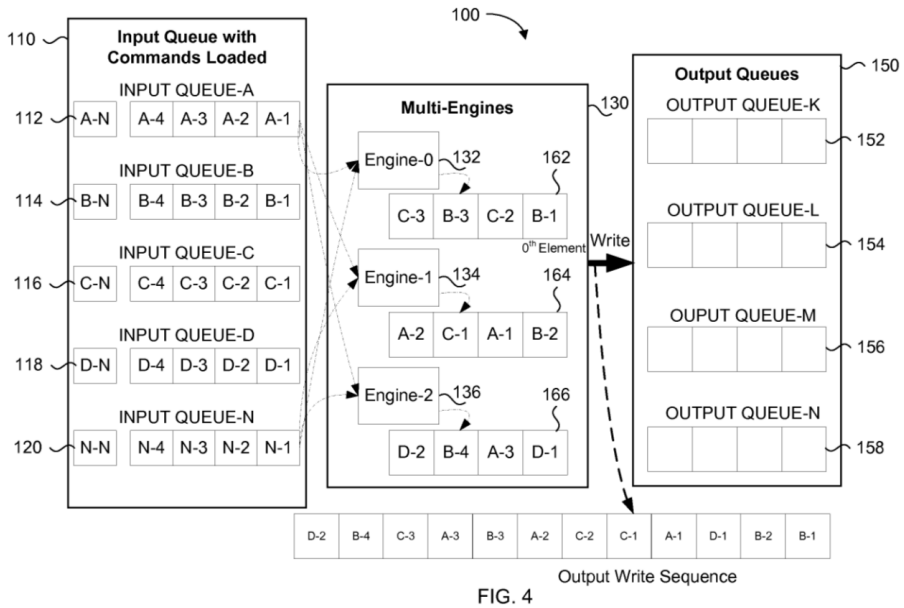
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	each output stream having the same relative order of output segments as in the corresponding input stream." Dongare '439 at col. 9, l. 59 – col. 10, l. 17. "FIG. 7 provides an illustration of an application of the apparatus of FIG. 5 in a multi-engine processing system. In applications such as RAID controllers, the Host computer 710 is connected to a RAID ASIC 720 via a PCIe protocol bus. RAID controller ASIC 720 has multiple RAID engines 730 to process data and commands to meet required throughput. The Host 710 and RAID controller 720 communicate to each other by using a set of queues 740 on the RAID controller 720. The queues 740 in the controller are classified as input queues 750 from the Host, and output queues from the RAID engines 730. The Host 710 writes commands for the RAID controller into the input queues 750. The input queues 750 are read by the embedded processor 760 and commands are created for RAID engines 730 to process data. Newly generated commands are loaded to input queues 750 dedicated to RAID engines 730. RAID engines 730 reads commands from input queues 750 and after processing them (e.g., read/write operations from Hard Disks 770), writes the processed commands back to output queues 780. The embedded processor 760 monitors output queues 780 for completion of RAID commands in the same order as was loaded in input queues 750. After completing the Host command, the embedded processor 760 will signal the Host 710 that the command is complete by writing into the outbound queue 780. A write to the outbound queue can generate an interrupt to the Host 710." Dongare '439 at col. 14, ll. 31–55.
8. The system of claim 7, wherein said input queue is a first-in-first-out (FIFO) buffer.	Dongare '439 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.	Dongare '439 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.	Dongare '439 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.

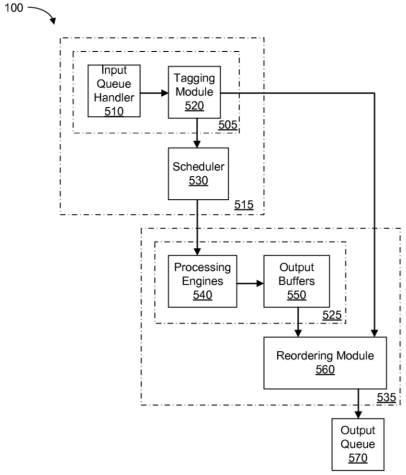
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
11. The system of claim 7, wherein said stored value is decremented by a predefined value prior to returning said stored value to said at least one second processing element.	Dongare ’439 teaches “[t]he system of claim 7, wherein said 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.
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.	Dongare ’439 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:	Dongare ’439 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;	Dongare ’439 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;	Dongare ’439 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;	Dongare ’439 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].

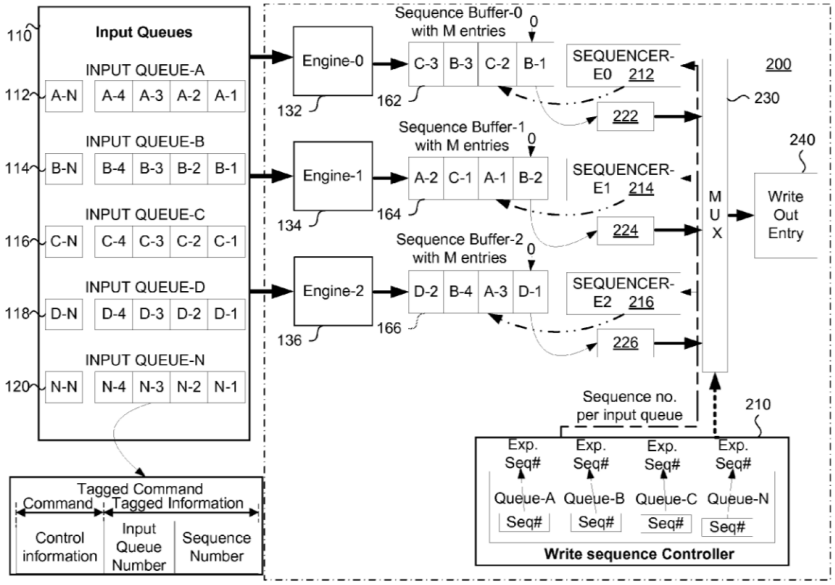
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
[13d] at least one second processing element in communication with said input queue and said output queue;	Dongare ’439 teaches “at least one second processing element in communication with said input queue and said output queue,” including at least the following exemplary citations. For example, Dongare ’439 teaches at least one second processing element in communication with said input queue and said output queue, by disclosing, as non–limiting examples, an embedded processor that communicates with the input queue and output queue. The embedded processor, the second processing element is in communication with the input and output queue. *See, e.g.:* “A method and apparatus for queue-ordering commands in multi-engines, multi-queues and/or multi-flows environment is provided. Commands from single/multiple queues and multi-flows are processed by multi-engines with different processing time and/or out of order, which breaks sequential order of commands from same input queue and commands are distributed across multiple engines' output buffer after processing. Processed commands are stored in dedicated command output buffer associated with each engine temporarily. The processed commands are re-ordered while writing out. Also commands can be scheduled to idle engines to achieve maximum throughput, thus utilizing the engines in an optimal manner.” Dongare ’439 at Abstract.  FIG. 1 Dongare ’439 at Figure 1.

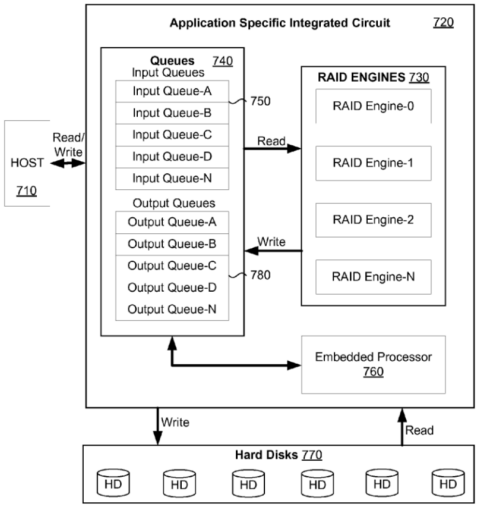
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 2 Dongare ’439 at Figure 2.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 3 Dongare ’439 at Figure 3.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 4 Dongare ’439 at Figure 4.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 5 Dongare ’439 at Figure 5.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 6 Dongare ’439 at Figure 6.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 7 Dongare ’439 at Figure 7. “In a first aspect, the present disclosure provides a method for queue ordering in a multi-engine processing system having a plurality of processing engines to process a plurality of input streams and to provide a corresponding plurality of output streams. The method comprises: receiving the plurality of input streams, each input stream having a plurality of input segments, the plurality of input segments ordered relative to other input segments within the input stream; appending an identity tag to the plurality of input segments in each input stream to produce a plurality of tagged input segments, the identity tag including information indicating the input stream and the relative order within the input stream of each of the plurality of input segments; iteratively assigning each of the plurality of tagged input segments to any one of the plurality of processing engines based on availability of the processing engines in the relative order of the tagged input segments within the respective input stream; processing each of the plurality of tagged input segments in its respective assigned processing engine to produce a corresponding plurality of tagged output segments; storing each of the plurality of tagged output segments in a dedicated output buffer associated with its respective assigned processing engine; and, reordering the plurality of tagged output segments to produce the plurality of output streams, each output stream having the same relative order as in the corresponding input stream.” Dongare ’439 at col. 1, ll. 36–60.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	"In another aspect, the present disclosure provides a multi-engine processing system comprising: an input queue handler configured to receive a plurality of input streams, each input stream having a plurality of input segments, the plurality of input segments ordered relative to other input segments within the input stream; a tagging module configured to append an identity tag to a plurality of input segments in each input stream to produce a plurality of tagged input segments, the identity tag including information indicating the input stream and the relative order within the input stream of each of the plurality of input segments; a scheduler configured to iteratively assign each of the plurality of tagged input segments to any one of a plurality of processing engines based on availability of the processing engines in the relative order of the tagged input segments within the respective input stream, each of the plurality of tagged input segments being processed in its respective assigned processing engine to produce a corresponding plurality of tagged output segments; a plurality of dedicated output buffers associated with the plurality of processing engines, the plurality of dedicated output buffers configured to store each of the plurality of tagged output segments; and, a reordering module configured to reorder the plurality of tagged output segments to produce a plurality of output streams, each output stream having the same relative order of output segments as in the corresponding input stream." Dongare '439 at col. 1, l. 61 – col. 2, l. 19. "Generally, the present disclosure provides a method and apparatus for queue ordering in a multi-processing system. The method and apparatus can be implemented in multi-engine, multi-queue, and/or multi-flow environments, such as in multi-engine processor systems or in a network domain. In a multi-engine processor or a multi-threading system, tasks from same program can be executed concomitantly on different processors to achieve maximum throughput. However, the completed task has to be provided in the order it was posted to the multi-processor system. The method and apparatus of the present disclosure provide for reordering completed tasks efficiently." Dongare '439 at col. 4, ll. 44–55. "FIG. 1 illustrates a multi-engine processing system 100 capable of handling a plurality of input queues or input streams 110. The input queues are sequentially ordered commands or segments to be processed by the plurality of engines 130. The processed commands are written out to the plurality of output queues 150. In the example shown in FIG. 1, the plurality of input queues are received and stored in input queue buffer-A 112, input queue buffer-B 114, input queue buffer-C 116, input queue buffer-D 118, and input queue buffer-N 120. The plurality of processing engines 130 includes Engine-0132, Engine-1134, and Engine-2136. The plurality of output queues 150 are written out and stored in output queue buffer-K 152, output queue buffer-L 154, output queue buffer-M 156, and output queue buffer-N 158. Thus, command streams A, B, C, D through N

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	can be received and stored in the respective input queue buffers and the processed commands can be written out and stored in output queue buffers. As shown in FIG. 1, the input queue buffer-A 112 has commands A-1, A-2, . . . A-N, input queue buffer-B 114 has commands B-1, B-2, . . . B-N, input queue buffer-C 116 has commands C-1, C-2, . . . C-N, input queue buffer-D 118 has commands D-1, D-2, . . . D-N, and input queue buffer-N 120 has commands N-1, N-2, . . . N-N." Dongare '439 at col. 5, ll. 3–26. "Commands read out from the input queues can be directly scheduled or assigned to any one of the plurality of the engines for processing based on the availability of the processing engine. In other example embodiments, read out commands can be stored in a shared buffer for pre-processing or filtering before scheduling it to a free engine for processing. When commands are stored in shared buffer, they can be provided to an idle engine immediately without fetching." Dongare '439 at col. 5, l. 62 – col. 6, l. 2. "In the example shown in FIG. 1, any one of Engine-0132, Engine-1134, or Engine-2136 is capable of processing the commands read out from the input queues. In an example embodiment, all engines may be identical in architecture and functionality or, in other example embodiments, each engine may differ in functionality and/or architecture, but capable of processing any command assigned to it. Commands from same input queue can be processed on multiple engines depending on availability of the engines at any given instance in order to use the engines optimally. The commands from different input queues may be interleaved and assigned (or scheduled) to an available engine provided the relative order of the commands within the respective command input queue is maintained. For example, Engine-0 can process commands B-1, C-2, B-3, and C-3, as shown in FIG. 2, provided commands C-1 and B-2 have been assigned for processing by other engines prior to scheduling commands C-2 and B-3 for processing by Engine-0." Dongare '439 at col. 6, ll. 3–20. "Once processed, the processed commands can be written to output queues with a single write channel or multiple write channels. In certain example embodiments, an input queue may also act as an output queue. The processed command may be buffered in one or more output queue buffers and one output queue buffer may hold processed commands from more than one command output queue." Dongare '439 at col. 6, ll. 21–27. "In the example embodiment shown in FIG. 2, the three engines are identical and are capable of processing any command assigned to them. However, depending on the nature of a command being processed, for example, the byte count and functionality of the command, a command provided earlier to one engine may take more time to process than another command assigned to other engine. Thus, processed commands would

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	be ready to write out at different time points. Also, the sequence of commands from the same input queue would be disturbed at the output side and may be distributed across the dedicated buffers of the respective engines. As shown in FIG. 2, each engine has a dedicated output buffer. For example, Engine-0132 has a dedicated output buffer 162, Engine-1134 has a dedicated output buffer 164, and Engine-2136 has a dedicated output buffer 166." Dongare '439 at col. 6, ll. 28–42. "The scheduling of commands from various command input queues shown in the example embodiment of FIG. 2 with respect to each engine will now be described. Engine-0132 was assigned and has processed commands from command input queues B and C. Commands B-1 and B-3 belong to command input queue B and commands C-2 and C-3 belong to command input queue C. Engine-0132 has processed these commands from different command input queues with sequence order: B-1, C-2, B-3, C-3 respectively and has stored the processed commands in its dedicated output buffer 162. Engine-1134 was assigned and has processed commands from command input queues A, B and C. Commands A-1 and A-2 belong to command input queue A, command B-2 belongs to command input queue B, and command C-1 belongs to command input queue C. Engine-1134 has processed these commands from different command input queues with sequence order: B-2, A-1, C-1, A-2 respectively and has stored the processed commands in its dedicated output buffer 164. Engine-2136 was assigned and has processed commands from command input queues A, B and D. Command A-3 belongs to command input queue A, command B-4 belongs to command input queue B, and commands D-1 and D-2 belong to command input queue D. Engine-2136 has processed these commands from different command input queues with sequence order: D-1, A-3, B-4, D-2 respectively and stored the processed commands in its dedicated output buffer 166." Dongare '439 at col. 7, ll. 6–33. "In an example embodiment, the method further comprises storing the plurality of tagged input segments in a shared buffer for pre-processing or filtering prior to assigning each of the plurality of tagged input segments to any one of the plurality of processing engines." Dongare '439 at col. 8, ll. 41–45. "FIG. 4 illustrates the required queue ordering while writing out the processed commands to the output queues, each output queue having the same relative sequence as in the corresponding input queue. As described earlier, the plurality of output queues 150 are written out and stored in output queue buffer-K 152, output queue buffer-L 154, output queue buffer-M 156, and output queue buffer-N 158. The various command input queues A-N are iteratively scheduled or assigned to be processed on any one of the plurality of processing engines based on the availability of the processing engines in the relative order if the commands

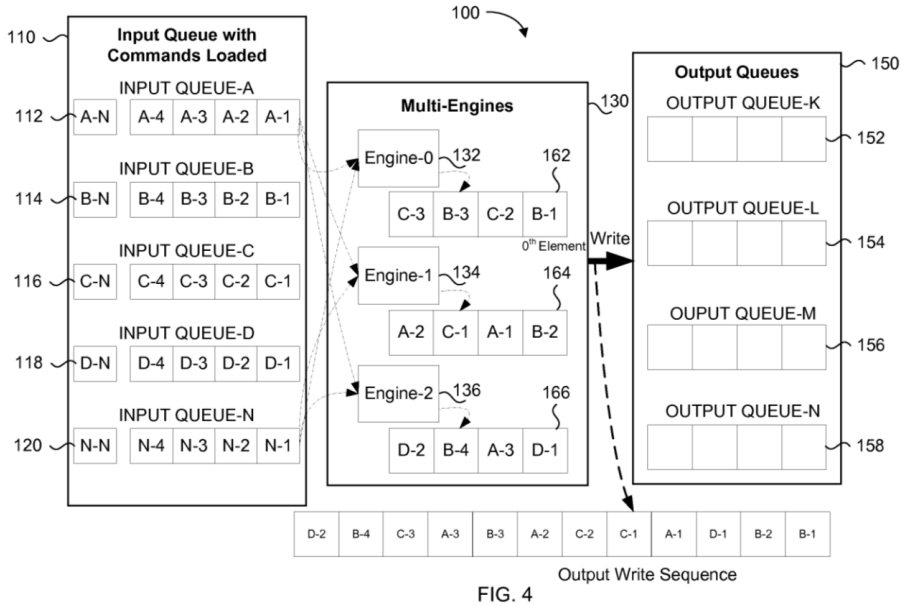
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	within their respective command input queue as discussed earlier with respect to the example embodiment of FIG. 2." Dongare '439 at col. 8, ll. 51–63. "In order to maintain the original order or sequence of the commands within a given command input queue, the processed commands must be written out to the command output queues in the same sequence as it was read out from the respective command input queue. It follows that A-1 should be written before A-2, A-2 should be written before A-3 and so on. As shown in FIG. 4, commands are written in the same relative order as in the command input queue: B-1, B-2, D-1, A-1, C-1, C-2, A-2, B-3, A-3, C-3, B-4, and D-2, i.e., the relative order of commands within a command output queue is maintained in the write out sequence irrespective of the processing engine that processed each command. In the example embodiment of FIG. 4, processed commands from command input queue A are written out as: A-1, A-2 and A-3, wherein commands A-1 and A-2 commands were processed by Engine-1 and command A-3 was processed by Engine-2. In the example embodiment, command A-3 was processed earlier to command A-2 but it was not written out since command A-2 was not processed. Processed command A-3 was stored in the dedicated output buffer 166 of Engine-2136 until command A-2 was processed and ready for write out. Similarly, processed commands from command input queue B are written out as: B-1, B-2, B-3 and B-4, wherein commands B-1 and B-3 were processed by Engine-0, command B-2 was processed by Engine-1 and command B4 was processed by Engine-2. In the example embodiment commands B-1 and B-2 were processed at the same time by different engines and hence could be written out sequentially. However, command B-4 was processed earlier by Engine-2 but it was not written out as command B-3 was not processed. Processed command B-4 was stored in dedicated output buffer 166 of Engine-2136 until command B-3 was processed and ready for write out. Processed commands from command input queue C are written as: C-1, C-2, C-3; wherein command C-1 was processed by Engine-1, commands C-2 and C-3 were processed by Engine-0. In the example embodiment command C-2 was processed earlier than command C-1, but it was not written out until the processing of command C-1 was completed. Processed command C-2 was stored in dedicated output buffer 162 of Engine-0136 until command C-1 was processed and ready for write out. Processed commands from command input queue D are written as: D-1, D-2; wherein commands D-1 and D-2 were processed by same engine sequentially and hence could be written out in that order without any waiting period." Dongare '439 at col. 9, ll. 5–50. "In an aspect, the present disclosure provides a multi-engine processing system comprising: an input queue handler configured to receive a plurality of input streams, each input stream having a plurality of input segments, the plurality of input segments ordered relative to other input segments within the input stream; a

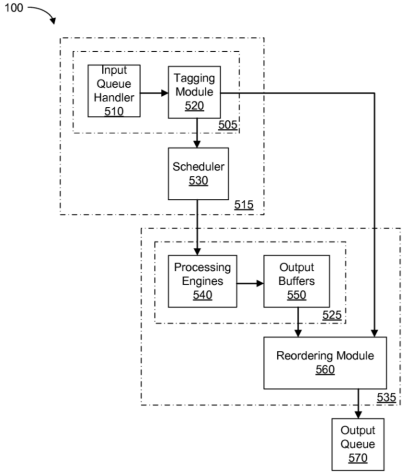
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	tagging module configured to append an identity tag to a plurality of input segments in each input stream to produce a plurality of tagged input segments, the identity tag including information indicating the input stream and the relative order within the input stream of each of the plurality of input segments; a scheduler configured to iteratively assign each of the plurality of tagged input segments to any one of a plurality of processing engines based on availability of the processing engines in the relative order of the tagged input segments within the respective input stream, each of the plurality of tagged input segments being processed in its respective assigned processing engine to produce a corresponding plurality of tagged output segments; a plurality of dedicated output buffers associated with the plurality of processing engines, the plurality of dedicated output buffers configured to store each of the plurality of tagged output segments; and, a reordering module configured to reorder the plurality of tagged output segments to produce a plurality of output streams, each output stream having the same relative order of output segments as in the corresponding input stream." Dongare '439 at col. 9, l. 59 – col. 10, l. 17. "FIG. 5 is a schematic representation of a multi-engine processing system in accordance with this disclosure. The multi-engine processing system 100 comprises an input queue handler 510; a tagging module 520; a scheduler 530; a plurality of processing engines 540; a plurality of output buffers 550; a reordering module 560; and an output queue 570. The input queue handler 510 is configured to receive a plurality of input streams. Each input stream has a plurality of input segments and the plurality of input segments ordered relative to other input segments within the input stream. The tagging module 520 is configured to append an identity tag to a plurality of input segments in each input stream to produce a plurality of tagged input segments. The identity tag includes information indicating the input stream and the relative order within the input stream of each of the plurality of input segments. The scheduler 530 is configured to iteratively assign each of the plurality of tagged input segments to any one of a plurality of processing engines 540 based on availability of the processing engines 540 in the relative order of the tagged input segments within the respective input stream. Each of the plurality of tagged input segments is processed in its respective assigned processing engine 540 to produce a corresponding plurality of tagged output segments. Each of the plurality of processing engines 540 has a corresponding plurality of dedicated output buffers 550. The plurality of dedicated output buffers 550 are configured to store each of the plurality of tagged output segments. The reordering module 560 is configured to reorder the plurality of tagged output segments to produce a plurality of output queues or streams 570. Each output stream has the same relative order of output segments as in the corresponding input stream." Dongare '439 at col. 10, l. 50 – col. 11, l. 13. "FIG. 6 illustrates an example embodiment of an apparatus 200 for queue ordering in a multi-engine processing environment. Commands in the various command input queues are tagged by appending unique

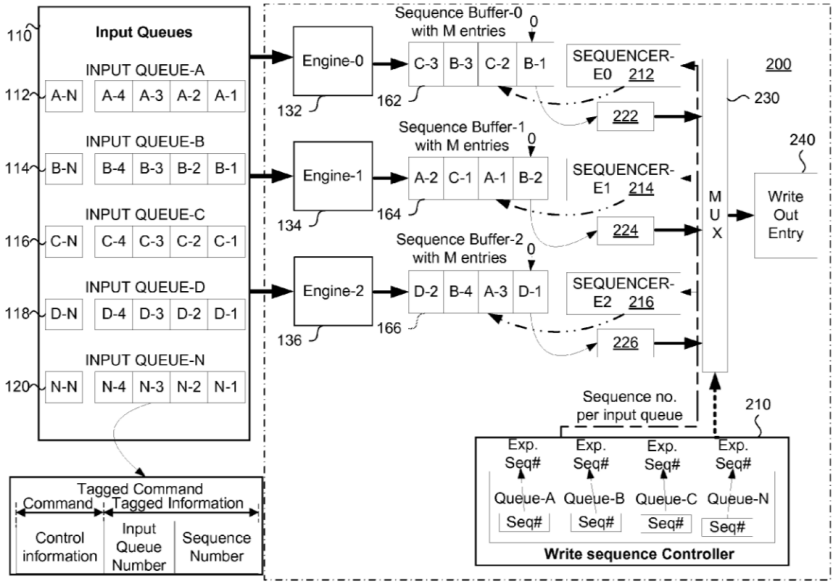
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	information or identity tags indicating the command input queue identification number (queue number) and its position (sequence number) within the respective command input queue. The identity tags may be appended by tagging module 520 while posting commands in the input queues. In other embodiments, commands may be tagged by the tagging mechanism after reading it from input queues and prior to scheduling the commands for processing by the various engines. Commands from command input queues are processed by an engine of the multi-engine processing system in any order resulting in out-of-order processed commands as described with respect to FIG. 4. Processed commands are stored in dedicated output command buffers of the respective engine that processed the command. Since commands from the same command input queue can be processed by multiple engines in an out-of-order sequence and due to the variable time taken by the engines to process assigned/scheduled commands, the processed commands may be distributed across multiple output buffers." Dongare '439 at col. 11, ll. 31–53. "Command Output Buffer: After processing of commands, each engine writes the processed commands in its dedicated command output buffer. Processed commands (along with their identity tags, also referred to as tagged output segments) are written to the command output buffer in the sequence they are processed by engines. The depth of the command output buffers can be chosen based on the bandwidth required by the system. The depth of a command output buffer refers to the number of processed commands or segments that can be stored in the command output buffer. By having a dedicated command buffer for each engine, each engine can directly access its command output buffer to write out processed commands without any waiting time. Furthermore, the engine can be freed to process the next command in the pipeline thereby increasing processing throughput. Thus, engine utilization can be optimized and engine idle times can be reduced. The processed commands in the command output buffers wait for write out until it is selected by the sequencer. Thus, processed commands from the command output buffers are read out in proper queue order without the requirement for additional sorting means." Dongare '439 at col. 12, ll. 14–36. "FIG. 7 provides an illustration of an application of the apparatus of FIG. 5 in a multi-engine processing system. In applications such as RAID controllers, the Host computer 710 is connected to a RAID ASIC 720 via a PCIe protocol bus. RAID controller ASIC 720 has multiple RAID engines 730 to process data and commands to meet required throughput. The Host 710 and RAID controller 720 communicate to each other by using a set of queues 740 on the RAID controller 720. The queues 740 in the controller are classified as input queues 750 from the Host, and output queues from the RAID engines 730. The Host 710 writes

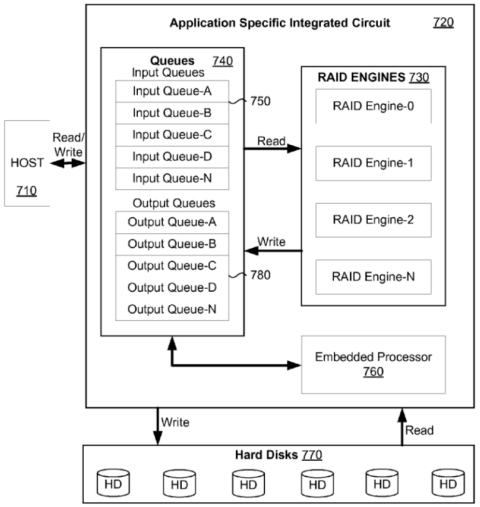
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	commands for the RAID controller into the input queues 750. The input queues 750 are read by the embedded processor 760 and commands are created for RAID engines 730 to process data. Newly generated commands are loaded to input queues 750 dedicated to RAID engines 730. RAID engines 730 reads commands from input queues 750 and after processing them (e.g., read/write operations from Hard Disks 770), writes the processed commands back to output queues 780. The embedded processor 760 monitors output queues 780 for completion of RAID commands in the same order as was loaded in input queues 750. After completing the Host command, the embedded processor 760 will signal the Host 710 that the command is complete by writing into the outbound queue 780. A write to the outbound queue can generate an interrupt to the Host 710." Dongare '439 at col. 14, ll. 31–55.
[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;	
[13f] computer storage storing instructions, which when executed by said at least one second processing element:	Dongare '439 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].
[13g] receives a command from said first processing element;	Dongare '439 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;	

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
[13i] determines when said read value indicates available space in said input queue to issue said command;	
[13j] issues said command to said input queue on behalf of said first processing element; and	Dongare ’439 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.	Dongare ’439 teaches “receives a response corresponding to said command from said output queue,” including at least the following exemplary citations. For example, Dongare ’439 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, receiving a response to the RAID command from the output queue after the RAID command has been processed. The embedded processor receives a corresponding response from the output queue. *See, e.g.:*

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 4 Dongare ’439 at Figure 4.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 5 Dongare ’439 at Figure 5.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 6 Dongare ’439 at Figure 6.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 FIG. 7 Dongare ’439 at Figure 7. “Once processed, the processed commands can be written to output queues with a single write channel or multiple write channels. In certain example embodiments, an input queue may also act as an output queue. The processed command may be buffered in one or more output queue buffers and one output queue buffer may hold processed commands from more than one command output queue.” Dongare ’439 at col. 6, ll. 21–27. “In order to maintain the original order or sequence of the commands within a given command input queue, the processed commands must be written out to the command output queues in the same sequence as it was read out from the respective command input queue. It follows that A-1 should be written before A-2, A-2 should be written before A-3 and so on. As shown in FIG. 4, commands are written in the same relative order as in the command input queue: B-1, B-2, D-1, A-1, C-1, C-2, A-2, B-3, A-3, C-3, B-4, and D-2, i.e., the relative order of commands within a command output queue is maintained in the write out sequence irrespective of the processing engine that processed each command. In the example embodiment of FIG. 4, processed commands from command input queue A are written out as: A-1, A-2 and A-3, wherein commands A-1 and A-2 commands were processed by Engine-1 and command A-

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	3 was processed by Engine-2. In the example embodiment, command A-3 was processed earlier to command A-2 but it was not written out since command A-2 was not processed. Processed command A-3 was stored in the dedicated output buffer 166 of Engine-2136 until command A-2 was processed and ready for write out. Similarly, processed commands from command input queue B are written out as: B-1, B-2, B-3 and B-4, wherein commands B-1 and B-3 were processed by Engine-0, command B-2 was processed by Engine-1 and command B4 was processed by Engine-2. In the example embodiment commands B-1 and B-2 were processed at the same time by different engines and hence could be written out sequentially. However, command B-4 was processed earlier by Engine-2 but it was not written out as command B-3 was not processed. Processed command B-4 was stored in dedicated output buffer 166 of Engine-2136 until command B-3 was processed and ready for write out. Processed commands from command input queue C are written as: C-1, C-2, C-3; wherein command C-1 was processed by Engine-1, commands C-2 and C-3 were processed by Engine-0. In the example embodiment command C-2 was processed earlier than command C-1, but it was not written out until the processing of command C-1 was completed. Processed command C-2 was stored in dedicated output buffer 162 of Engine-0136 until command C-1 was processed and ready for write out. Processed commands from command input queue D are written as: D-1, D-2; wherein commands D-1 and D-2 were processed by same engine sequentially and hence could be written out in that order without any waiting period." Dongare '439 at col. 9, ll. 5–50. "In an aspect, the present disclosure provides a multi-engine processing system comprising: an input queue handler configured to receive a plurality of input streams, each input stream having a plurality of input segments, the plurality of input segments ordered relative to other input segments within the input stream; a tagging module configured to append an identity tag to a plurality of input segments in each input stream to produce a plurality of tagged input segments, the identity tag including information indicating the input stream and the relative order within the input stream of each of the plurality of input segments; a scheduler configured to iteratively assign each of the plurality of tagged input segments to any one of a plurality of processing engines based on availability of the processing engines in the relative order of the tagged input segments within the respective input stream, each of the plurality of tagged input segments being processed in its respective assigned processing engine to produce a corresponding plurality of tagged output segments; a plurality of dedicated output buffers associated with the plurality of processing engines, the plurality of dedicated output buffers configured to store each of the plurality of tagged output segments; and, a reordering module configured to reorder the plurality of tagged output segments to produce a plurality of output streams,

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	each output stream having the same relative order of output segments as in the corresponding input stream." Dongare '439 at col. 9, l. 59 – col. 10, l. 17. "FIG. 7 provides an illustration of an application of the apparatus of FIG. 5 in a multi-engine processing system. In applications such as RAID controllers, the Host computer 710 is connected to a RAID ASIC 720 via a PCIe protocol bus. RAID controller ASIC 720 has multiple RAID engines 730 to process data and commands to meet required throughput. The Host 710 and RAID controller 720 communicate to each other by using a set of queues 740 on the RAID controller 720. The queues 740 in the controller are classified as input queues 750 from the Host, and output queues from the RAID engines 730. The Host 710 writes commands for the RAID controller into the input queues 750. The input queues 750 are read by the embedded processor 760 and commands are created for RAID engines 730 to process data. Newly generated commands are loaded to input queues 750 dedicated to RAID engines 730. RAID engines 730 reads commands from input queues 750 and after processing them (e.g., read/write operations from Hard Disks 770), writes the processed commands back to output queues 780. The embedded processor 760 monitors output queues 780 for completion of RAID commands in the same order as was loaded in input queues 750. After completing the Host command, the embedded processor 760 will signal the Host 710 that the command is complete by writing into the outbound queue 780. A write to the outbound queue can generate an interrupt to the Host 710." Dongare '439 at col. 14, ll. 31–55.
14. The system of claim 13, wherein said input queue is a first-in-first-out (FIFO) buffer.	Dongare '439 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.	Dongare '439 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.	Dongare '439 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.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
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.	
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.	Dongare ’439 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.