

Invalidity of the '596 Patent by Rosenthal '930 and Other Prior Art

The asserted claims of U.S. Patent No. 8,924,596 are anticipated and/or rendered obvious in view of US5805930A ("Rosenthal '930") 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 Rosenthal '930. 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, Rosenthal '930 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, Rosenthal '930 teaches a system for dividing and synchronizing processing tasks by disclosing, as non-limiting examples, a digital system with multiple parallel FIFO buffers for application programs that divide and synchronize a processing task. <i>See, e.g.:</i> "A digital system which uses an arrangement of one or more parallel FIFO buffers in which each FIFO buffer handles data from only one application program at any time. In order to assure that no data written to a FIFO buffer by an application program will overflow the FIFO buffer, each FIFO buffer includes a flow control register which must be read by the processing unit running the application before writing data to an input/output device. The register stores a value which indicates the amount of space available in the FIFO buffer to which data may be written. Reading this register tells the application program how much data may be written without running the risk of overflowing the data storage area which the input/output device has available." Rosenthal '930 at Abstract.

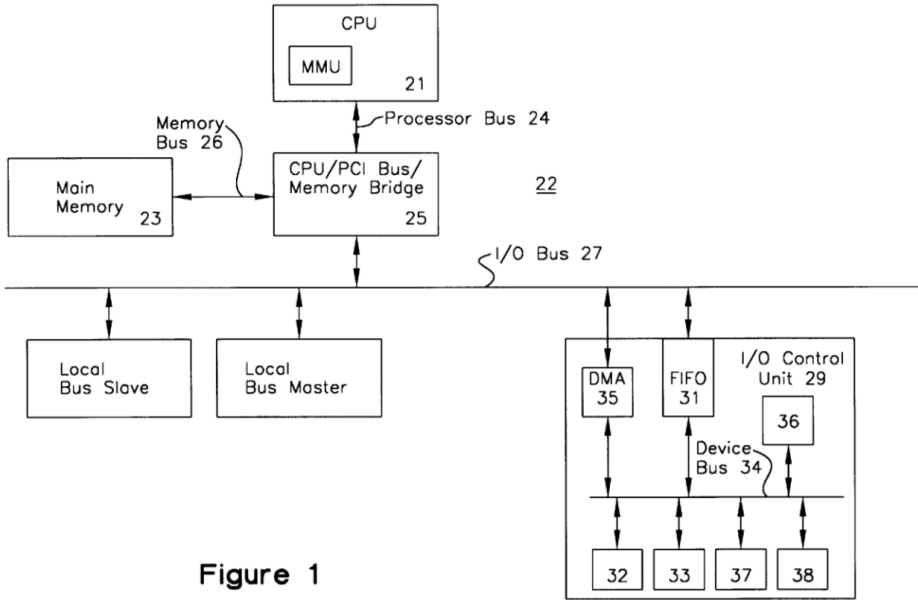
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	Figure 1 Rosenthal '930 at Figure 1. “Since any practical input/output device will have limited FIFO buffer storage for holding data written to it over the input/output bus, any architecture for input/output devices must include some technique for controlling the flow of data so that this storage is not exhausted. It is desirable to provide a means for providing flow control for a computer system or similar system utilizing FIFO buffers to receive data so that the operation may proceed as rapidly as possible without loss of data.” Rosenthal '930 at col. 1, l. 66 – col. 2, l. 7. “It is another object of the present invention to provide a flow control system for an architecture which utilizes FIFO buffers for buffering write operations. These and other objects of the present invention are realized in a system which uses an arrangement of parallel FIFO buffers in which each FIFO buffer handles data from only one application program at any time. In order to assure that no data written to a FIFO buffer by an application program will overflow the FIFO buffer, each FIFO buffer includes a flow control register which must be read by the central processing unit running the application before writing data to an input/output device. The register stores a value which indicates the amount of space available in the FIFO buffer to which data may be written. Reading this register

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	tells the application program how much data may be written without running the risk of overflowing the data storage area which the input/output device has available. Provided that no application program ever writes more than the data for which it has permission, no data will be lost.” Rosenthal ’930 at col. 2, ll. 15–33. “FIG. 1 is a block diagram of a computer system 22 which has been devised to overcome the problems of the prior art. The system 22 provides a new input/output architecture which cooperates with other components of present systems based on the PDP11 architecture, runs legacy code for those systems, yet is able to drastically increase the speed of input/output operations for new application programs. In order to accomplish this, the new architecture of the system allows read and write operations by application programs to be made directly to the input/output devices. This eliminates the cumbersome multi-step software processes invoked by prior art systems using the operating system and trusted code for every input/output access. In order to accomplish the process safely, the input/output architecture of the system 22 utilizes an input/output control unit 29 which provides its own virtual name-to-physical-device and context translation for all of the input/output devices associated with the new control unit 29 on its own device bus 34. By enforcing this translation, application programs can write directly to input/output devices without affecting assets of other application programs. Once this translation from virtual names furnished by the application programs to physical input/output devices on the device bus is accomplished and context has been furnished to the input/output devices, translation of addresses of input/output devices on the input/output bus into physical addresses on the device bus 34 is accomplished directly by hardware at the input/output control unit 29. This hardware also checks permissions; and, when an operation is known to be safe, it is performed by hardware. When a translation operation fails, the operating system software is invoked. Thus, rather than trapping every input/output operation to determine whether it is safe as is done in prior art computer systems based on the PDP11 architecture, the system 22 traps and sends to operating system software only unsafe operations allowing hardware to accomplish most translations and greatly speeding the access of input/output devices.” Rosenthal ’930 at col. 3, ll. 29–65. “Although the input/output architecture of system 22 may be used with systems utilizing a single input/output bus for all operations, the preferred embodiment of system 22 functions as well in a system utilizing a local bus 27 such as the Peripheral Component Interconnect (PCI) bus or the Video Electronics Standards Association (VESA) local bus which may be associated with other input/output buses. While the discussion of this specification will assume that bus 27 is a local bus, the local bus 27 is also referred to in this specification as the input/output bus 27 in order to emphasize its use. In arrangements utilizing local buses, the central processing unit 21 and main memory 23 are typically arranged on a processor bus 24 and a memory bus 26, respectively, and are joined to a bridge unit 25. The central processing unit 21 typically

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	includes a memory management unit. The bridge unit 25 provides write buffering for operations between the central processing unit 21 and the input/output bus 27, between the central processing unit 21 and main memory 23 on the processor bus 24 and the memory bus 26, and between the input/output bus 27 and main memory 23.” Rosenthal ’930 at col. 4, ll. 28–48. “Those skilled in the art will recognize that the use of 128 individual FIFO buffers 39 to receive commands written by the central processing unit is very expensive and probably not economically feasible for use with personal computers. In addition, it is expensive of system resources for an operating system to context switch the central processing unit from one application program to another. Thus, the operating system will try to ensure that once an application is given the central processing unit the application will retain control long enough to amortize the cost of the context switch. This means that typically in a one processor system at most one FIFO buffer will be active at a time. In a multi-processor system, typically at most as many FIFO buffers will be active as there are processors.” Rosenthal ’930 at col. 21, ll. 39–52. “One embodiment of the new architecture provides a means for attaining almost the same speed as is possible with 128 FIFO buffers but much more economically. Instead of 128 individual FIFO buffers 39, the described embodiment utilizes a smaller number of FIFO buffers 39 than the number of areas into which the input/output address space may be divided and thus the number of application programs which may be run. Utilizing a smaller number of FIFO buffers causes each FIFO buffer to function as a cache for commands which may be sent by any of the application programs. In the preferred embodiment, one FIFO buffer 39 is used for each processor which may be writing to the input/output control unit 29. For example, if the computer utilizes two processors arranged as symmetrical multi-processors, then two individual FIFO buffers are utilized. If, as is usually the case, only a single central processing unit is used, then a single FIFO buffer 39 may in the limit act as a cache for commands from all application programs which may be running on the central processing unit. Such an arrangement is illustrated in FIG. 6. Of course, in either case more FIFO buffers might be used.” Rosenthal ’930 at col. 21, l. 53 – col. 22, l. 6.
[1a] an input queue implemented in hardware;	Rosenthal ’930 teaches “an input queue implemented in hardware,” including at least the following exemplary citations. For example, Rosenthal ’930 teaches an input queue implemented in hardware, by disclosing, as non-limiting examples, FIFO queue for storing operations. Defendants have proposed a construction that “queue implemented in hardware” means “queue implemented in physical circuits without software intervention.” The input queue in Rosenthal ’930 meets Defendants’

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	construction because, for example, Rosenthal '930 teaches a hardware implementation of its teachings, which includes its teachings of queues. <i>See, e.g.</i>, Rosenthal '930 at 5:24–27 (“The control unit 29 includes a hardware FIFO unit 31 for receiving incoming commands addressed to the input/output devices on a device bus 34.”). <i>See, e.g.</i>: “A digital system which uses an arrangement of one or more parallel FIFO buffers in which each FIFO buffer handles data from only one application program at any time. In order to assure that no data written to a FIFO buffer by an application program will overflow the FIFO buffer, each FIFO buffer includes a flow control register which must be read by the processing unit running the application before writing data to an input/output device. The register stores a value which indicates the amount of space available in the FIFO buffer to which data may be written. Reading this register tells the application program how much data may be written without running the risk of overflowing the data storage area which the input/output device has available.” Rosenthal '930 at Abstract.  Figure 1 Rosenthal '930 at Figure 1.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	The diagram illustrates a system architecture. At the top, a horizontal line represents Bus 27. Below it, a large rectangular block represents the Input/Output Control Unit 29. Inside this unit, there is a 'Decode' block 30 connected to Bus 27. Below 'Decode' is a 'Pusher' block 53. To the left of the Pusher is a memory structure with a vertical stack of cells (36) and a horizontal row of cells (3, 11, 32). Above this memory structure is a block labeled 'MMU 50' containing '52', and below it is a 'DMA 35'. To the right of the memory structure is a 'Resource Manager' block (dashed outline) and a 'Translation Table' 38. Below the Translation Table is a block labeled '40'. At the bottom of the unit, there is a 'Sound Generator' 37. Below the unit, a horizontal line represents Bus 34. Connected to Bus 34 are a 'Disk Controller' 32, a 'Graphics Output Controller Memory' 33, and the 'Sound Generator' 37. Various numbered blocks (7, 31, 42, 45) and arrows indicate data flow and control signals between the components. Figure 3 Rosenthal '930 at Figure 3.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	The diagram illustrates a system architecture centered around an Input/Output Control Unit 29. This unit is connected to a Decode block 30 via Bus 27. Inside the Input/Output Control Unit 29, there is a Pusher 53, a DMA 35, and an MMU 50 containing a sub-block 52. A central processing area includes a 3x3 grid of blocks labeled 3, 11, and 32, with a vertical stack of blocks labeled 7, 36, and 42. A Resource Manager block is connected to the central area via a dashed line. A Translation Table 38 is connected to the central area via a dashed line. A bus labeled 40 connects the central area to a Translation Table 38. Below the Input/Output Control Unit 29, a Disk Controller 32, a Graphics Output Controller Memory 33, and a Sound Generator 37 are connected to Bus 34. Bus 34 also connects to the Translation Table 38. Figure 6 Rosenthal '930 at Figure 6.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	Figure 7 Rosenthal '930 at Figure 7. “It is another object of the present invention to provide a flow control system for an architecture which utilizes FIFO buffers for buffering write operations. These and other objects of the present invention are realized in a system which uses an arrangement of parallel FIFO buffers in which each FIFO buffer handles data from only one application program at any time. In order to assure that no data written to a FIFO buffer by an application program will overflow the FIFO buffer, each FIFO buffer includes a flow control register which must be read by the central processing unit running the application before writing data to an input/output device. The register stores a value which indicates the amount of space available in the FIFO buffer to which data may be written. Reading this register tells the application program how much data may be written without running the risk of overflowing the data storage area which the input/output device has available. Provided that no application program ever writes more than the data for which it has permission, no data will be lost.” Rosenthal '930 at col. 2, ll. 15–33. “The architecture of the system 22 has been designed so that it eliminates almost all read operations of input/output devices by the central processing unit. In order to accomplish this, the input/output control unit 29 includes a first-in first-out (FIFO) unit 31 for storing write operations directed to the input/output control

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	unit. The FIFO unit 31 queues incoming write operations. Unlike FIFO units in prior art systems, it stores both addresses and data. This allows the write operations to the input/output control unit 29 to occur asynchronously so that both the central processing unit 21 and the input/output control unit 29 may be functioning independently of one another and neither need wait for operations of the other.” Rosenthal ’930 at col. 3, l. 66 – col. 4, l. 11. “In the architecture of system 22, an input/output control unit 29 is shown joined to the input/output bus 27. The control unit 29 includes a hardware FIFO unit 31 for receiving incoming commands addressed to the input/output devices on a device bus 34.” Rosenthal ’930 at col. 4, ll. 60–64. “In one embodiment, the FIFO unit 31 includes a plurality of FIFO buffers 39 (see FIG. 3). When an application program desires to write to an input/output device on the device bus 34, it addresses that device. Decoding circuitry decodes the address by reviewing a number of the highest order bits decoded by the chip sufficient to indicate a unique portion of the input/output address space assigned to an application program and places the command in the appropriate FIFO buffer 39 for that application program. Each FIFO buffer 39 handles commands only from the application program to which the address area has been mapped. An arrangement using a smaller number of FIFO buffers 39 than one buffer for each application program (as few as a single buffer 39) could also be utilized as long as the address of the FIFO buffer is mapped to only one application program at a time.” Rosenthal ’930 at col. 5, ll. 51–66. “The unit 29 receives physical addresses furnished by the memory management unit and virtual names furnished by application programs for operations to be performed which have been sent to the FIFO unit 31 and controls the translation of those virtual names for all input/output devices. The hardware unit 29 includes the device bus 34 to which the individual input/output devices such as a disk controller 32, a graphics output controller 33, and a sound generator 37 are shown joined. The unit 29 also includes a DMA unit 35 which is adapted to transfer data between the individual input/output devices and main memory for use by the central processing unit or other components of the system.” Rosenthal ’930 at col. 5, l. 67 – col. 6, l. 10. “Because the amount of space in each of the FIFO buffers is limited to thirty-two entries in the embodiment being described (and will be limited to some finite number in any embodiment), it is possible for a buffer 39 to fill with data and be unable to take more commands directed to the particular area. As discussed previously, this will occur when the central processing unit is writing a FIFO buffer 39 faster than the input/output control unit 29 is able to handle the commands placed in the FIFO unit 31. In such a case, the input/output control unit 29 issues a hold command on the bus 27 to stop additional commands from being sent by the application program. During the holdoff period, the puller circuit 40 may empty some of the

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	commands of the application program from the FIFO buffer 39 so that space will be available. Alternatively, when the bus holdoff expires, commands from the program may still fill the FIFO buffer 39; and the new commands may have to be otherwise dealt with or the data will be lost to the input/output control unit 29.” Rosenthal ’930 at col. 16, ll. 14–32. “In one embodiment of the new architecture, a flow control register 45 is included for each FIFO buffer 39 of the input/output control unit 29. The register 45 stores an indication of the number of available spaces for commands in the associated FIFO buffer 39. The use of this flow control register 45 allows a requirement to be placed on an application program attempting an access of the input/output control unit 29 that it first determine whether space is available in the FIFO unit 31 before it writes any command. Before writing any data, the application program must determine the amount of "free" space to which it may write. This it does by having the central processing unit read the value in the flow control register 45 associated with the FIFO buffer 39 being utilized. Once this value has been read, the application program may send up to that amount of data before it need test again to determine whether free space remains to which it may send more data.” Rosenthal ’930 at col. 16, ll. 33–49. “In one embodiment of the new architecture the individual buffers 39 are implemented utilizing an array of random access memory (RAM). In such an arrangement, the data comprising the commands is placed in the RAM and pointers are utilized to define the positions of the beginning and end of the individual FIFO buffers 39 and the beginning and end of the data. In such an embodiment, a simple subtraction of a pointer designating the last data placed in a particular RAM buffer from the pointer designating the end of the buffer provides the value to be held by the flow control register. In such a case, the flow control register itself may be a logical register whose contents are computed whenever it is read.” Rosenthal ’930 at col. 16, l. 58 – col. 17, l. 3. “The use of the flow control register 45 allows an application program presently using the resources of the input/output control unit 29 to proceed without causing the overflow of the FIFO unit 31. Whenever the application has written commands totaling the amount previously read from the register 45, it must read the register again. This is especially useful for application programs which may wish to transfer a series of commands in sequence, possibly using the burst transfer mode of operation provided by many of the modern buses. By ascertaining the space available in the FIFO buffer 39 before transferring a sequence of commands, the application can know that sufficient space will be available for the operation. Provided that no application program ever writes more than the data for which it has permission, no data will be lost.” Rosenthal ’930 at col. 17, ll. 4–18.

Exhibit 596-16

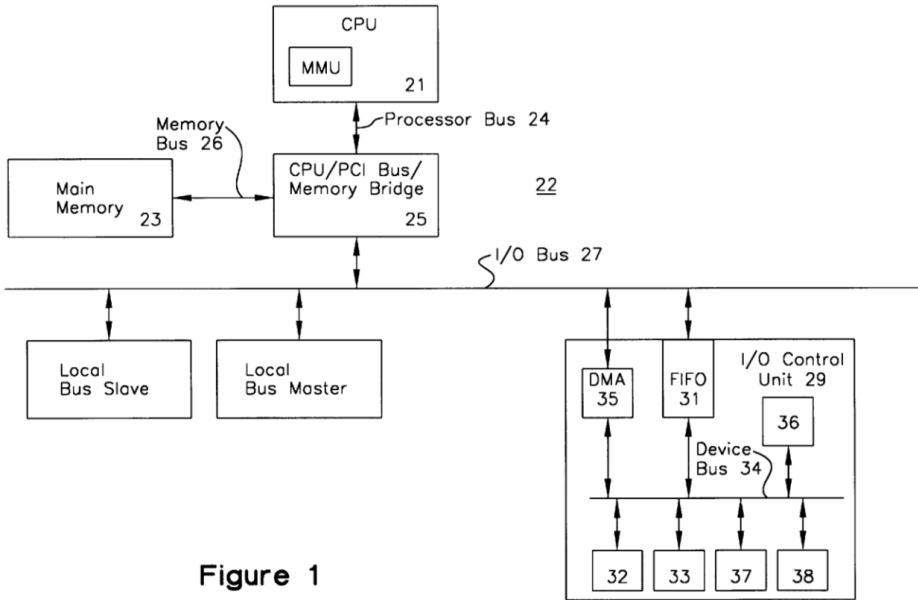
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	“Various operating conditions can cause the flow control register 45 associated with a buffer 39 to store different values. If the central processing unit is writing at a rate faster than the input/output device can handle the commands, then a FIFO buffer 39 which is initially empty and provides a maximum free count value when read by the central processing unit will provide a lower free count number after the amount of data first indicated by the free count register has been sent. This will occur because the FIFO buffer 39 will empty more slowly than it is filled. In such a case, the next value read by the application program after sending the amount of data designated by the free count register will be a smaller number. The application program may again send this amount of data and be sure that the FIFO buffer will not overflow. On the other hand, if the central processing unit is writing at a rate slower than the input/output device can handle the commands, then a FIFO buffer which is initially empty and provides a maximum free count value when read by the central processing unit, will provide the same free count number after the amount of data first indicated by the flow control register has been sent. This will occur because the FIFO will empty more rapidly than it is filled and will always be empty when the register 45 is read by the central processing unit.” Rosenthal '930 at col. 17, ll. 19–42. “Those skilled in the art will recognize that the use of 128 individual FIFO buffers 39 to receive commands written by the central processing unit is very expensive and probably not economically feasible for use with personal computers. In addition, it is expensive of system resources for an operating system to context switch the central processing unit from one application program to another. Thus, the operating system will try to ensure that once an application is given the central processing unit the application will retain control long enough to amortize the cost of the context switch. This means that typically in a one processor system at most one FIFO buffer will be active at a time. In a multi-processor system, typically at most as many FIFO buffers will be active as there are processors.” Rosenthal '930 at col. 21, ll. 39–52. “One embodiment of the new architecture provides a means for attaining almost the same speed as is possible with 128 FIFO buffers but much more economically. Instead of 128 individual FIFO buffers 39, the described embodiment utilizes a smaller number of FIFO buffers 39 than the number of areas into which the input/output address space may be divided and thus the number of application programs which may be run. Utilizing a smaller number of FIFO buffers causes each FIFO buffer to function as a cache for commands which may be sent by any of the application programs. In the preferred embodiment, one FIFO buffer 39 is used for each processor which may be writing to the input/output control unit 29. For example, if the computer utilizes two processors arranged as symmetrical multi-processors, then two individual FIFO buffers are utilized. If, as is usually the case, only a single central processing unit is used, then a single FIFO buffer 39 may in the limit act as a cache for commands from all application programs which may be running on the

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	central processing unit. Such an arrangement is illustrated in FIG. 6. Of course, in either case more FIFO buffers might be used.” Rosenthal ’930 at col. 21, l. 53 – col. 22, l. 6. “As with the embodiments which use one FIFO buffer for each application area, a free count register 45 and a puller are included with each FIFO buffer 39 as a part of the input/output control unit 29. The register 45 stores an indication of the number of available entries in each FIFO buffer 39 of the FIFO unit 31 and allows a requirement to be placed on an application program attempting to access the input/output control unit 29 that it first determine whether space is available in the FIFO unit 31. It also allows an application presently using the resources of the input/output control unit 29 to proceed without overflowing the FIFO unit 31. The application program may obtain a boundary value indicating the number of free entries in the FIFO buffer by reading from a designated offset in any of the sub-areas of its mapped area. The application may write up to the amount of data designated by this boundary value without further testing and be certain that overflow of the FIFO unit will not occur. As with the multiple FIFO buffer embodiments, the FIFO buffer arrangements of the described embodiment make use of a memory runout area; such a memory runout area assures that a FIFO buffer may be used to cache commands from a plurality of application programs. The use of a memory runout area also allows the system to make use of lying in order to enhance the speed of transfer by reducing the number of read operations by the central processing unit of the free count register 45.” Rosenthal ’930 at col. 22, ll. 27–52. “9. A method for controlling the flow of data in a data transfer path of a computer having a first-in first-out (FIFO) buffer with a plurality of stages positioned between a computer input-output bus and input/output devices, each stage providing storage for both data and a virtual address which are translated to provide a physical address for commands decoupled from application programs originating the commands, comprising the steps of: detecting a value indicating an amount of space available for data in the FIFO buffer, transferring to the FIFO buffer an amount of data up to an amount determine from the value detected, and detecting another value indicating an amount of space available for data in the FIFO buffer before transferring additional data to the FIFO buffer.” Rosenthal ’930 at col. 24, ll. 51–65.
[1b] an output queue implemented in hardware;	Rosenthal ’930 teaches “an output queue implemented in hardware,” including at least the following exemplary citations.

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	For example, Rosenthal '930 teaches an output queue implemented in hardware, by disclosing, as non-limiting examples, a DMA device. Defendants have proposed a construction that “queue implemented in hardware” means “queue implemented in physical circuits without software intervention.” The output queue in Rosenthal '930 meets Defendants’ construction because, for example, Rosenthal '930 teaches a hardware implementation of its teachings, which includes its teachings of queues. <i>See, e.g.</i>, Rosenthal '930 at 4:54–57 (“The DMA device 35 includes its own memory management unit which allows writes from input/output devices to the virtual memory space of an application without involving the operating system in the translation process.”). <i>See, e.g.</i>: “To help maintain this asynchronous operating arrangement and to eliminate read operations to the extent possible, the input/output control unit also includes an advanced direct memory access (DMA) device 35 which provides direct memory access for operations conducted involving input/output devices. The DMA device 35 allows the results of input/output operations to be written from input/output devices to main memory 23 rather than requiring read operations by the central processing unit 21 to obtain these results. This eliminates almost all need for the central processing unit 21 to read input/output devices and drastically increases the overall speed of input/output operations. The DMA device 35 includes its own memory management unit which allows writes from input/output devices to the virtual memory space of an application without involving the operating system in the translation process.” Rosenthal '930 at col. 4, ll. 12–27.
[1c] a first processing element having access to said input queue and said output queue;	Rosenthal '930 teaches “a first processing element having access to said input queue and said output queue,” including at least the following exemplary citations. For example, Rosenthal '930 teaches a first processing element having access to said input queue and said output queue, by disclosing, as non-limiting examples, a CPU. The CPU has access to both the input queue and output queue. <i>See, e.g.</i>: “A digital system which uses an arrangement of one or more parallel FIFO buffers in which each FIFO buffer handles data from only one application program at any time. In order to assure that no data written to a FIFO buffer by an application program will overflow the FIFO buffer, each FIFO buffer includes a flow control register which must be read by the processing unit running the application before writing data to an input/output device. The register stores a value which indicates the amount of space available in the FIFO buffer to which data may be written. Reading this register tells the application program how much data may

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	be written without running the risk of overflowing the data storage area which the input/output device has available.” Rosenthal '930 at Abstract.  Figure 1 Rosenthal '930 at Figure 1.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	The diagram illustrates a system architecture. At the top, a horizontal line represents Bus 27. Below it, a box labeled 'Decode 30' is connected to Bus 27. To the right, a box labeled 'Input/Output Control Unit 29' is connected to Bus 27. Below Bus 27, a large box contains several components: 'MMU 50' with a sub-box '52' and 'DMA 35' are on the left. In the center is a 'Pusher 53' box with a grid of cells containing numbers 7, 3, 11, 32, and 39. To the right of the grid is a box labeled '45'. Below the grid is a box labeled '36'. To the right of the grid is a box labeled '31'. Below the grid is a box labeled '40'. To the right of the grid is a box labeled '38' labeled 'Translation Table'. A dashed box labeled 'Resource Manager' is to the right of the grid. Below the large box is a horizontal line representing Bus 34. Below Bus 34 are three boxes: 'Disk Controller 32', 'Graphics Output Controller Memory 33', and 'Sound Generator 37'. Arrows indicate data flow between components and buses. Figure 3 Rosenthal '930 at Figure 3.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	The diagram illustrates a system architecture centered around an Input/Output Control Unit 29. This unit is connected to a Decode block 30 via Bus 27. Inside the Input/Output Control Unit 29, there is a Pusher block 53, an MMU block 50 containing a sub-block 52, and a DMA block 35. A central processing area contains a grid of blocks labeled 3, 11, and 32, with a vertical stack of blocks labeled 7 and 36. A Resource Manager block is connected to the central area via a dashed line. A Translation Table 38 is connected to the central area via a dashed line. A block labeled 40 is connected to the central area via a dashed line. A block labeled 45 is connected to the central area via a dashed line. A block labeled 49 is connected to the central area via a dashed line. The Input/Output Control Unit 29 is connected to a Disk Controller 32, a Graphics Output Controller Memory 33, and a Sound Generator 37 via Bus 34. Figure 6 Rosenthal '930 at Figure 6.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	Figure 7 Rosenthal '930 at Figure 7. “It is another object of the present invention to provide a flow control system for an architecture which utilizes FIFO buffers for buffering write operations. These and other objects of the present invention are realized in a system which uses an arrangement of parallel FIFO buffers in which each FIFO buffer handles data from only one application program at any time. In order to assure that no data written to a FIFO buffer by an application program will overflow the FIFO buffer, each FIFO buffer includes a flow control register which must be read by the central processing unit running the application before writing data to an input/output device. The register stores a value which indicates the amount of space available in the FIFO buffer to which data may be written. Reading this register tells the application program how much data may be written without running the risk of overflowing the data storage area which the input/output device has available. Provided that no application program ever writes more than the data for which it has permission, no data will be lost.” Rosenthal '930 at col. 2, ll. 15–33. “FIG. 1 is a block diagram of a computer system 22 which has been devised to overcome the problems of the prior art. The system 22 provides a new input/output architecture which cooperates with other components of present systems based on the PDP11 architecture, runs legacy code for those systems, yet is able to drastically

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	increase the speed of input/output operations for new application programs. In order to accomplish this, the new architecture of the system allows read and write operations by application programs to be made directly to the input/output devices. This eliminates the cumbersome multi-step software processes invoked by prior art systems using the operating system and trusted code for every input/output access. In order to accomplish the process safely, the input/output architecture of the system 22 utilizes an input/output control unit 29 which provides its own virtual name-to-physical-device and context translation for all of the input/output devices associated with the new control unit 29 on its own device bus 34. By enforcing this translation, application programs can write directly to input/output devices without affecting assets of other application programs. Once this translation from virtual names furnished by the application programs to physical input/output devices on the device bus is accomplished and context has been furnished to the input/output devices, translation of addresses of input/output devices on the input/output bus into physical addresses on the device bus 34 is accomplished directly by hardware at the input/output control unit 29. This hardware also checks permissions; and, when an operation is known to be safe, it is performed by hardware. When a translation operation fails, the operating system software is invoked. Thus, rather than trapping every input/output operation to determine whether it is safe as is done in prior art computer systems based on the PDP11 architecture, the system 22 traps and sends to operating system software only unsafe operations allowing hardware to accomplish most translations and greatly speeding the access of input/output devices.” Rosenthal ’930 at col. 3, ll. 29–65. “The architecture of the system 22 has been designed so that it eliminates almost all read operations of input/output devices by the central processing unit. In order to accomplish this, the input/output control unit 29 includes a first-in first-out (FIFO) unit 31 for storing write operations directed to the input/output control unit. The FIFO unit 31 queues incoming write operations. Unlike FIFO units in prior art systems, it stores both addresses and data. This allows the write operations to the input/output control unit 29 to occur asynchronously so that both the central processing unit 21 and the input/output control unit 29 may be functioning independently of one another and neither need wait for operations of the other.” Rosenthal ’930 at col. 3, l. 66 – col. 4, l. 11. “Although the input/output architecture of system 22 may be used with systems utilizing a single input/output bus for all operations, the preferred embodiment of system 22 functions as well in a system utilizing a local bus 27 such as the Peripheral Component Interconnect (PCI) bus or the Video Electronics Standards Association (VESA) local bus which may be associated with other input/output buses. While the discussion of this specification will assume that bus 27 is a local bus, the local bus 27 is also referred to in this specification as the input/output bus 27 in order to emphasize its use. In arrangements utilizing local buses,

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	the central processing unit 21 and main memory 23 are typically arranged on a processor bus 24 and a memory bus 26, respectively, and are joined to a bridge unit 25. The central processing unit 21 typically includes a memory management unit. The bridge unit 25 provides write buffering for operations between the central processing unit 21 and the input/output bus 27, between the central processing unit 21 and main memory 23 on the processor bus 24 and the memory bus 26, and between the input/output bus 27 and main memory 23.” Rosenthal ’930 at col. 4, ll. 28–48. “In the architecture of system 22, an input/output control unit 29 is shown joined to the input/output bus 27. The control unit 29 includes a hardware FIFO unit 31 for receiving incoming commands addressed to the input/output devices on a device bus 34.” Rosenthal ’930 at col. 4, ll. 60–64. “In one embodiment, the FIFO unit 31 includes a plurality of FIFO buffers 39 (see FIG. 3). When an application program desires to write to an input/output device on the device bus 34, it addresses that device. Decoding circuitry decodes the address by reviewing a number of the highest order bits decoded by the chip sufficient to indicate a unique portion of the input/output address space assigned to an application program and places the command in the appropriate FIFO buffer 39 for that application program. Each FIFO buffer 39 handles commands only from the application program to which the address area has been mapped. An arrangement using a smaller number of FIFO buffers 39 than one buffer for each application program (as few as a single buffer 39) could also be utilized as long as the address of the FIFO buffer is mapped to only one application program at a time.” Rosenthal ’930 at col. 5, ll. 51–66. “The unit 29 receives physical addresses furnished by the memory management unit and virtual names furnished by application programs for operations to be performed which have been sent to the FIFO unit 31 and controls the translation of those virtual names for all input/output devices. The hardware unit 29 includes the device bus 34 to which the individual input/output devices such as a disk controller 32, a graphics output controller 33, and a sound generator 37 are shown joined. The unit 29 also includes a DMA unit 35 which is adapted to transfer data between the individual input/output devices and main memory for use by the central processing unit or other components of the system.” Rosenthal ’930 at col. 5, l. 67 – col. 6, l. 10. “FIG. 3 is a block diagram illustrating details of the input/output control unit 29 including the device bus 34 and the input/output devices arranged on that bus. As described above, the input/output control unit 29 includes a decode unit 30 which receives commands directly from the input/output bus 27 and transfers the commands to a pusher circuit 53 which transfers the command to one of the FIFO buffers 39 of the FIFO unit 31. The FIFO unit 31 stores the data along with the addresses for each of the commands being transferred to all of the input/output devices associated with the input/output control unit 29. The buffers 39 of the FIFO

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	unit replace the individual data registers used by the input/output devices of the prior art. However, in contrast to the registers used by the prior art for communication on the input/output bus 27, the FIFO unit 31 allows a very much larger number of commands to be processed much more rapidly and facilitates the asynchronous operations of the input/output devices and the central processing unit. In one embodiment, the FIFO unit 31 includes 128 individual FIFO buffers 39 each of which has thirty-two stages. The use of 128 individual FIFO buffers 39 allows a FIFO buffer to be assigned to each of 128 different application programs which may be running on the central processing unit for the use of that application program alone. The thirty-two stages of each buffer 39 allow any FIFO buffer to hold thirty-two individual serially ordered commands at any time. Although each of the stages of the FIFO unit 31 holds (along with the address bits) the data held by an individual register of a typical prior art input/output device, a particular system has the ability to store commands for over 16 thousand 32 bit registers for each of 128 different application programs which may map different addresses decoded by the input/output control unit 29.” Rosenthal ’930 at col. 8, ll. 21–53. “The use of the flow control register 45 allows an application program presently using the resources of the input/output control unit 29 to proceed without causing the overflow of the FIFO unit 31. Whenever the application has written commands totaling the amount previously read from the register 45, it must read the register again. This is especially useful for application programs which may wish to transfer a series of commands in sequence, possibly using the burst transfer mode of operation provided by many of the modern buses. By ascertaining the space available in the FIFO buffer 39 before transferring a sequence of commands, the application can know that sufficient space will be available for the operation. Provided that no application program ever writes more than the data for which it has permission, no data will be lost.” Rosenthal ’930 at col. 17, ll. 4–18. “Various operating conditions can cause the flow control register 45 associated with a buffer 39 to store different values. If the central processing unit is writing at a rate faster than the input/output device can handle the commands, then a FIFO buffer 39 which is initially empty and provides a maximum free count value when read by the central processing unit will provide a lower free count number after the amount of data first indicated by the free count register has been sent. This will occur because the FIFO buffer 39 will empty more slowly than it is filled. In such a case, the next value read by the application program after sending the amount of data designated by the free count register will be a smaller number. The application program may again send this amount of data and be sure that the FIFO buffer will not overflow. On the other hand, if the central processing unit is writing at a rate slower than the input/output device can handle the commands, then a FIFO buffer which is initially empty and provides a maximum free count value when read by the central processing unit, will provide the same free count number after the amount of data first indicated by the flow

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	control register has been sent. This will occur because the FIFO will empty more rapidly than it is filled and will always be empty when the register 45 is read by the central processing unit.” Rosenthal ’930 at col. 17, ll. 19–42. “Those skilled in the art will recognize that the use of 128 individual FIFO buffers 39 to receive commands written by the central processing unit is very expensive and probably not economically feasible for use with personal computers. In addition, it is expensive of system resources for an operating system to context switch the central processing unit from one application program to another. Thus, the operating system will try to ensure that once an application is given the central processing unit the application will retain control long enough to amortize the cost of the context switch. This means that typically in a one processor system at most one FIFO buffer will be active at a time. In a multi-processor system, typically at most as many FIFO buffers will be active as there are processors.” Rosenthal ’930 at col. 21, ll. 39–52. “One embodiment of the new architecture provides a means for attaining almost the same speed as is possible with 128 FIFO buffers but much more economically. Instead of 128 individual FIFO buffers 39, the described embodiment utilizes a smaller number of FIFO buffers 39 than the number of areas into which the input/output address space may be divided and thus the number of application programs which may be run. Utilizing a smaller number of FIFO buffers causes each FIFO buffer to function as a cache for commands which may be sent by any of the application programs. In the preferred embodiment, one FIFO buffer 39 is used for each processor which may be writing to the input/output control unit 29. For example, if the computer utilizes two processors arranged as symmetrical multi-processors, then two individual FIFO buffers are utilized. If, as is usually the case, only a single central processing unit is used, then a single FIFO buffer 39 may in the limit act as a cache for commands from all application programs which may be running on the central processing unit. Such an arrangement is illustrated in FIG. 6. Of course, in either case more FIFO buffers might be used.” Rosenthal ’930 at col. 21, l. 53 – col. 22, l. 6. “As with the embodiments which use one FIFO buffer for each application area, a free count register 45 and a puller are included with each FIFO buffer 39 as a part of the input/output control unit 29. The register 45 stores an indication of the number of available entries in each FIFO buffer 39 of the FIFO unit 31 and allows a requirement to be placed on an application program attempting to access the input/output control unit 29 that it first determine whether space is available in the FIFO unit 31. It also allows an application presently using the resources of the input/output control unit 29 to proceed without overflowing the FIFO unit 31. The application program may obtain a boundary value indicating the number of free entries in the FIFO buffer by reading from a designated offset in any of the sub-areas of its mapped area. The application may write up to

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	the amount of data designated by this boundary value without further testing and be certain that overflow of the FIFO unit will not occur. As with the multiple FIFO buffer embodiments, the FIFO buffer arrangements of the described embodiment make use of a memory runout area; such a memory runout area assures that a FIFO buffer may be used to cache commands from a plurality of application programs. The use of a memory runout area also allows the system to make use of lying in order to enhance the speed of transfer by reducing the number of read operations by the central processing unit of the free count register 45.” Rosenthal ’930 at col. 22, ll. 27–52. “9. A method for controlling the flow of data in a data transfer path of a computer having a first-in first-out (FIFO) buffer with a plurality of stages positioned between a computer input-output bus and input/output devices, each stage providing storage for both data and a virtual address which are translated to provide a physical address for commands decoupled from application programs originating the commands, comprising the steps of: detecting a value indicating an amount of space available for data in the FIFO buffer, transferring to the FIFO buffer an amount of data up to an amount determine from the value detected, and detecting another value indicating an amount of space available for data in the FIFO buffer before transferring additional data to the FIFO buffer.” Rosenthal ’930 at col. 24, ll. 51–65.
[1d] at least one second processing element in communication with said first processing element;	Rosenthal ’930 teaches “at least one second processing element in communication with said first processing element,” including at least the following exemplary citations. For example, Rosenthal ’930 teaches at least one second processing element in communication with said first processing element, by disclosing, as non-limiting examples, an I/O control unit. The I/O control unit, the second processing element is in communication with the first processing element. <i>See, e.g.:</i>

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	Figure 1 Rosenthal '930 at Figure 1.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	<pre>graph TD; AP[Application Program] --> S[System]; S --> ID1[I/O Driver #1]; ID1 --> IOCU[Input/Output Control Unit 29]; AP --> IOCU; S --> ID2[I/O Driver #2]; ID2 --> RM[I/O Driver #2 Resource Manager]; RM --> IOCU;</pre> The diagram illustrates a system architecture. At the top is a box labeled 'Application Program'. An arrow points down from it to a box labeled 'System'. From 'System', an arrow points down to a box labeled 'I/O Driver #1'. From 'I/O Driver #1', an arrow points down to a wide box at the bottom labeled 'Input/Output Control Unit 29'. A second arrow points down from 'Application Program' to the 'Input/Output Control Unit 29'. From 'System', an arrow points down to a box labeled 'I/O Driver #2'. From 'I/O Driver #2', an arrow points down to a box labeled 'I/O Driver #2 Resource Manager'. From 'I/O Driver #2 Resource Manager', an arrow points down to the 'Input/Output Control Unit 29'. Figure 2 Rosenthal '930 at Figure 2.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	The diagram illustrates a system architecture. At the top, a horizontal line represents Bus 27. Below it, a large rectangular block represents the Input/Output Control Unit 29. Inside this unit, at the top, is a 'Decode' block 30. Below 'Decode' 30 is a 'Pusher' block 53. To the left of the 'Pusher' 53 is a block containing 'MMU 50' and 'DMA 35'. Below the 'Pusher' 53 is a large grid-like structure with various numbered cells (7, 3, 11, 32, 39, 40, 45, etc.). To the right of this grid is a 'Resource Manager' block (dashed outline) and a 'Translation Table' 38. Below the grid is a 'Disk Controller' 32, a 'Graphics Output Controller Memory' 33, and a 'Sound Generator' 37. A horizontal line at the bottom represents Bus 34. Arrows indicate data flow between these components and the buses. Figure 3 Rosenthal '930 at Figure 3.

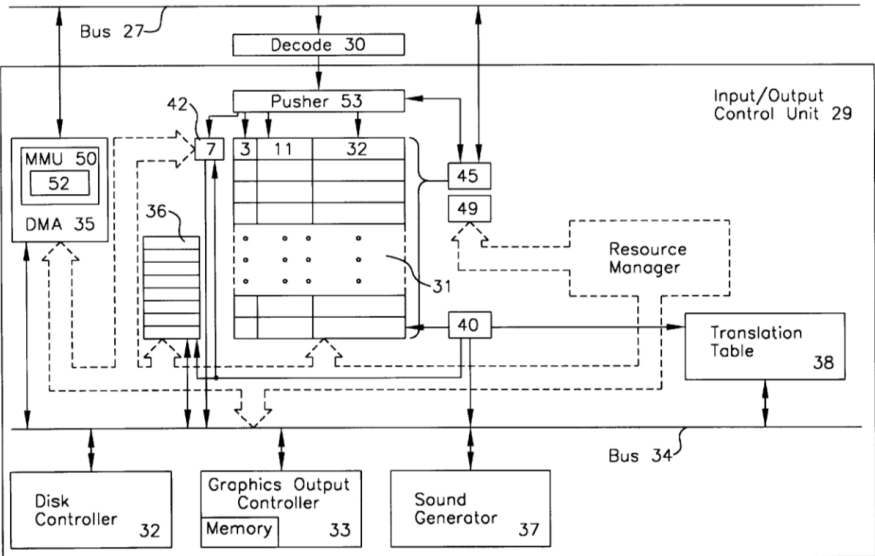
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Figure 6 Rosenthal '930 at Figure 6. “It is another object of the present invention to provide a flow control system for an architecture which utilizes FIFO buffers for buffering write operations. These and other objects of the present invention are realized in a system which uses an arrangement of parallel FIFO buffers in which each FIFO buffer handles data from only one application program at any time. In order to assure that no data written to a FIFO buffer by an application program will overflow the FIFO buffer, each FIFO buffer includes a flow control register which must be read by the central processing unit running the application before writing data to an input/output device. The register stores a value which indicates the amount of space available in the FIFO buffer to which data may be written. Reading this register tells the application program how much data may be written without running the risk of overflowing the data storage area which the input/output device has available. Provided that no application program ever writes more than the data for which it has permission, no data will be lost.” Rosenthal '930 at col. 2, ll. 15–33. “FIG. 1 is a block diagram of a computer system 22 which has been devised to overcome the problems of the prior art. The system 22 provides a new input/output architecture which cooperates with other components of present systems based on the PDP11 architecture, runs legacy code for those systems, yet is able to drastically

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	increase the speed of input/output operations for new application programs. In order to accomplish this, the new architecture of the system allows read and write operations by application programs to be made directly to the input/output devices. This eliminates the cumbersome multi-step software processes invoked by prior art systems using the operating system and trusted code for every input/output access. In order to accomplish the process safely, the input/output architecture of the system 22 utilizes an input/output control unit 29 which provides its own virtual name-to-physical-device and context translation for all of the input/output devices associated with the new control unit 29 on its own device bus 34. By enforcing this translation, application programs can write directly to input/output devices without affecting assets of other application programs. Once this translation from virtual names furnished by the application programs to physical input/output devices on the device bus is accomplished and context has been furnished to the input/output devices, translation of addresses of input/output devices on the input/output bus into physical addresses on the device bus 34 is accomplished directly by hardware at the input/output control unit 29. This hardware also checks permissions; and, when an operation is known to be safe, it is performed by hardware. When a translation operation fails, the operating system software is invoked. Thus, rather than trapping every input/output operation to determine whether it is safe as is done in prior art computer systems based on the PDP11 architecture, the system 22 traps and sends to operating system software only unsafe operations allowing hardware to accomplish most translations and greatly speeding the access of input/output devices.” Rosenthal ’930 at col. 3, ll. 29–65. “Although the input/output architecture of system 22 may be used with systems utilizing a single input/output bus for all operations, the preferred embodiment of system 22 functions as well in a system utilizing a local bus 27 such as the Peripheral Component Interconnect (PCI) bus or the Video Electronics Standards Association (VESA) local bus which may be associated with other input/output buses. While the discussion of this specification will assume that bus 27 is a local bus, the local bus 27 is also referred to in this specification as the input/output bus 27 in order to emphasize its use. In arrangements utilizing local buses, the central processing unit 21 and main memory 23 are typically arranged on a processor bus 24 and a memory bus 26, respectively, and are joined to a bridge unit 25. The central processing unit 21 typically includes a memory management unit. The bridge unit 25 provides write buffering for operations between the central processing unit 21 and the input/output bus 27, between the central processing unit 21 and main memory 23 on the processor bus 24 and the memory bus 26, and between the input/output bus 27 and main memory 23.” Rosenthal ’930 at col. 4, ll. 28–48.

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	“In the architecture of system 22, an input/output control unit 29 is shown joined to the input/output bus 27. The control unit 29 includes a hardware FIFO unit 31 for receiving incoming commands addressed to the input/output devices on a device bus 34.” Rosenthal ’930 at col. 4, ll. 60–64. “In one embodiment, the FIFO unit 31 includes a plurality of FIFO buffers 39 (see FIG. 3). When an application program desires to write to an input/output device on the device bus 34, it addresses that device. Decoding circuitry decodes the address by reviewing a number of the highest order bits decoded by the chip sufficient to indicate a unique portion of the input/output address space assigned to an application program and places the command in the appropriate FIFO buffer 39 for that application program. Each FIFO buffer 39 handles commands only from the application program to which the address area has been mapped. An arrangement using a smaller number of FIFO buffers 39 than one buffer for each application program (as few as a single buffer 39) could also be utilized as long as the address of the FIFO buffer is mapped to only one application program at a time.” Rosenthal ’930 at col. 5, ll. 51–66. “The unit 29 receives physical addresses furnished by the memory management unit and virtual names furnished by application programs for operations to be performed which have been sent to the FIFO unit 31 and controls the translation of those virtual names for all input/output devices. The hardware unit 29 includes the device bus 34 to which the individual input/output devices such as a disk controller 32, a graphics output controller 33, and a sound generator 37 are shown joined. The unit 29 also includes a DMA unit 35 which is adapted to transfer data between the individual input/output devices and main memory for use by the central processing unit or other components of the system.” Rosenthal ’930 at col. 5, l. 67 – col. 6, l. 10. “FIG. 3 is a block diagram illustrating details of the input/output control unit 29 including the device bus 34 and the input/output devices arranged on that bus. As described above, the input/output control unit 29 includes a decode unit 30 which receives commands directly from the input/output bus 27 and transfers the commands to a pusher circuit 53 which transfers the command to one of the FIFO buffers 39 of the FIFO unit 31. The FIFO unit 31 stores the data along with the addresses for each of the commands being transferred to all of the input/output devices associated with the input/output control unit 29. The buffers 39 of the FIFO unit replace the individual data registers used by the input/output devices of the prior art. However, in contrast to the registers used by the prior art for communication on the input/output bus 27, the FIFO unit 31 allows a very much larger number of commands to be processed much more rapidly and facilitates the asynchronous operations of the input/output devices and the central processing unit. In one embodiment, the FIFO unit 31 includes 128 individual FIFO buffers 39 each of which has thirty-two stages. The use of 128 individual FIFO buffers 39 allows a FIFO buffer to be assigned to each of 128 different application programs which may be

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	running on the central processing unit for the use of that application program alone. The thirty-two stages of each buffer 39 allow any FIFO buffer to hold thirty-two individual serially ordered commands at any time. Although each of the stages of the FIFO unit 31 holds (along with the address bits) the data held by an individual register of a typical prior art input/output device, a particular system has the ability to store commands for over 16 thousand 32 bit registers for each of 128 different application programs which may map different addresses decoded by the input/output control unit 29.” Rosenthal ’930 at col. 8, ll. 21–53. “Those skilled in the art will recognize that the use of 128 individual FIFO buffers 39 to receive commands written by the central processing unit is very expensive and probably not economically feasible for use with personal computers. In addition, it is expensive of system resources for an operating system to context switch the central processing unit from one application program to another. Thus, the operating system will try to ensure that once an application is given the central processing unit the application will retain control long enough to amortize the cost of the context switch. This means that typically in a one processor system at most one FIFO buffer will be active at a time. In a multi-processor system, typically at most as many FIFO buffers will be active as there are processors.” Rosenthal ’930 at col. 21, ll. 39–52. “One embodiment of the new architecture provides a means for attaining almost the same speed as is possible with 128 FIFO buffers but much more economically. Instead of 128 individual FIFO buffers 39, the described embodiment utilizes a smaller number of FIFO buffers 39 than the number of areas into which the input/output address space may be divided and thus the number of application programs which may be run. Utilizing a smaller number of FIFO buffers causes each FIFO buffer to function as a cache for commands which may be sent by any of the application programs. In the preferred embodiment, one FIFO buffer 39 is used for each processor which may be writing to the input/output control unit 29. For example, if the computer utilizes two processors arranged as symmetrical multi-processors, then two individual FIFO buffers are utilized. If, as is usually the case, only a single central processing unit is used, then a single FIFO buffer 39 may in the limit act as a cache for commands from all application programs which may be running on the central processing unit. Such an arrangement is illustrated in FIG. 6. Of course, in either case more FIFO buffers might be used.” Rosenthal ’930 at col. 21, l. 53 – col. 22, l. 6. “9. A method for controlling the flow of data in a data transfer path of a computer having a first-in first-out (FIFO) buffer with a plurality of stages positioned between a computer input-output bus and input/output devices, each stage providing storage for both data and a virtual address which are translated to provide a physical address for commands decoupled from application programs originating the commands, comprising the steps of:

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	detecting a value indicating an amount of space available for data in the FIFO buffer, transferring to the FIFO buffer an amount of data up to an amount determine from the value detected, and detecting another value indicating an amount of space available for data in the FIFO buffer before transferring additional data to the FIFO buffer.” Rosenthal ’930 at col. 24, ll. 51–65.
[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;	Rosenthal ’930 teaches “a reservation register implemented in hardware storing a value indicative of available space in said input queue, said reservation register accessible by both said first processing element and said at least one second processing element,” including at least the following exemplary citations. For example, Rosenthal ’930 teaches a reservation register implemented in hardware storing a value indicative of available space in said input queue, said reservation register accessible by both said first processing element and said at least one second processing element, by disclosing, as non-limiting examples, a flow control register. The flow control register stores a value of indicative space and is accessible by both the first and second processing elements discussed above. Defendants have proposed a construction that “register” means “set of flip–flops that store a set of related bits.” Rosenthal ’930 teaches a reservation register that is a set of flip–flops that store a set of related bits. <i>See, e.g.</i>, Rosenthal ’930 at 16:58–62 (“In one embodiment of the new architecture, a flow control register 45 is included for each FIFO buffer 39 of the input/output control unit 29. The register 45 stores an indication of the number of available spaces for commands in the associated FIFO buffer 39.”). For example, the flow control register identified above is explained and described as a register and the description of its behavior is consistent with a set of flip–flops that store a set of related bits. <i>See id.</i> Rosenthal ’930 is a multiprocessor system that uses registers. A POSITA reading Rosenthal ’930 would have understood the description of these registers to be a set of flip–flops that store a set of related bits in view of the relevant disclosures, and further because that is how hardware systems and registers in particular had been designed and implemented for decades. <i>See, e.g.</i>: “A digital system which uses an arrangement of one or more parallel FIFO buffers in which each FIFO buffer handles data from only one application program at any time. In order to assure that no data written to a FIFO buffer by an application program will overflow the FIFO buffer, each FIFO buffer includes a flow control register which must be read by the processing unit running the application before writing data to an input/output device. The register stores a value which indicates the amount of space available in the FIFO buffer to which data may be written. Reading this register tells the application program how much data may

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	be written without running the risk of overflowing the data storage area which the input/output device has available.” Rosenthal ’930 at Abstract. “It is another object of the present invention to provide a flow control system for an architecture which utilizes FIFO buffers for buffering write operations. These and other objects of the present invention are realized in a system which uses an arrangement of parallel FIFO buffers in which each FIFO buffer handles data from only one application program at any time. In order to assure that no data written to a FIFO buffer by an application program will overflow the FIFO buffer, each FIFO buffer includes a flow control register which must be read by the central processing unit running the application before writing data to an input/output device. The register stores a value which indicates the amount of space available in the FIFO buffer to which data may be written. Reading this register tells the application program how much data may be written without running the risk of overflowing the data storage area which the input/output device has available. Provided that no application program ever writes more than the data for which it has permission, no data will be lost.” Rosenthal ’930 at col. 2, ll. 15–33. “In one embodiment of the new architecture, a flow control register 45 is included for each FIFO buffer 39 of the input/output control unit 29. The register 45 stores an indication of the number of available spaces for commands in the associated FIFO buffer 39. The use of this flow control register 45 allows a requirement to be placed on an application program attempting an access of the input/output control unit 29 that it first determine whether space is available in the FIFO unit 31 before it writes any command. Before writing any data, the application program must determine the amount of "free" space to which it may write. This it does by having the central processing unit read the value in the flow control register 45 associated with the FIFO buffer 39 being utilized. Once this value has been read, the application program may send up to that amount of data before it need test again to determine whether free space remains to which it may send more data.” Rosenthal ’930 at col. 16, ll. 33–49. “In one embodiment of the new architecture the individual buffers 39 are implemented utilizing an array of random access memory (RAM). In such an arrangement, the data comprising the commands is placed in the RAM and pointers are utilized to define the positions of the beginning and end of the individual FIFO buffers 39 and the beginning and end of the data. In such an embodiment, a simple subtraction of a pointer designating the last data placed in a particular RAM buffer from the pointer designating the end of the buffer provides the value to be held by the flow control register. In such a case, the flow control register itself may be a logical register whose contents are computed whenever it is read.” Rosenthal ’930 at col. 16, l. 58 – col. 17, l. 3.

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	“The use of the flow control register 45 allows an application program presently using the resources of the input/output control unit 29 to proceed without causing the overflow of the FIFO unit 31. Whenever the application has written commands totaling the amount previously read from the register 45, it must read the register again. This is especially useful for application programs which may wish to transfer a series of commands in sequence, possibly using the burst transfer mode of operation provided by many of the modern buses. By ascertaining the space available in the FIFO buffer 39 before transferring a sequence of commands, the application can know that sufficient space will be available for the operation. Provided that no application program ever writes more than the data for which it has permission, no data will be lost.” Rosenthal '930 at col. 17, ll. 4–18. “Various operating conditions can cause the flow control register 45 associated with a buffer 39 to store different values. If the central processing unit is writing at a rate faster than the input/output device can handle the commands, then a FIFO buffer 39 which is initially empty and provides a maximum free count value when read by the central processing unit will provide a lower free count number after the amount of data first indicated by the free count register has been sent. This will occur because the FIFO buffer 39 will empty more slowly than it is filled. In such a case, the next value read by the application program after sending the amount of data designated by the free count register will be a smaller number. The application program may again send this amount of data and be sure that the FIFO buffer will not overflow. On the other hand, if the central processing unit is writing at a rate slower than the input/output device can handle the commands, then a FIFO buffer which is initially empty and provides a maximum free count value when read by the central processing unit, will provide the same free count number after the amount of data first indicated by the flow control register has been sent. This will occur because the FIFO will empty more rapidly than it is filled and will always be empty when the register 45 is read by the central processing unit.” Rosenthal '930 at col. 17, ll. 19–42. “In utilizing the free count value, the application program should deduct the value of any data which it has already written before each new write operation commences. In this manner, the application program knows at any time the amount of data which it may still write to the input/output device before another read of the free count register will be required. This allows a number of write operations to occur before a read of the register 45 (which significantly slows operation of the overall system) need occur. This also assures that any writes which might be presently in the write channel on the input/output bus are accounted for in determining the actual value of the free count left. It should be noted that this read of the free count register is the only read necessary in dealing with the input/output control unit 29. It would be possible either to interrupt the central processing unit if a FIFO buffer 39 filled or to use the DMA unit 35 to transfer the value in the free

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	count register to the central processing unit instead of reading the value, but in the particular embodiment reading the free count register appears to be the fastest operation.” Rosenthal ’930 at col. 17, l. 52 – col. 18, l. 4. “In one embodiment, the application program may obtain a free count value indicating the number of free entries in a FIFO buffer 39 by reading from a designated method offset in any of the sub-areas of the area mapped to the application program. The value read is that stored in the flow control register 45 associated with the particular FIFO buffer 39. This value is then placed on the bus when the central processing unit provides the appropriate control signals to implement the read operation. The application may write up to the amount of data designated by this free count value without further testing and be certain that overflow of the FIFO buffer 39 will not occur.” Rosenthal ’930 at col. 18, ll. 16–27. “As with the embodiments which use one FIFO buffer for each application area, a free count register 45 and a puller are included with each FIFO buffer 39 as a part of the input/output control unit 29. The register 45 stores an indication of the number of available entries in each FIFO buffer 39 of the FIFO unit 31 and allows a requirement to be placed on an application program attempting to access the input/output control unit 29 that it first determine whether space is available in the FIFO unit 31. It also allows an application presently using the resources of the input/output control unit 29 to proceed without overflowing the FIFO unit 31. The application program may obtain a boundary value indicating the number of free entries in the FIFO buffer by reading from a designated offset in any of the sub-areas of its mapped area. The application may write up to the amount of data designated by this boundary value without further testing and be certain that overflow of the FIFO unit will not occur. As with the multiple FIFO buffer embodiments, the FIFO buffer arrangements of the described embodiment make use of a memory runout area; such a memory runout area assures that a FIFO buffer may be used to cache commands from a plurality of application programs. The use of a memory runout area also allows the system to make use of lying in order to enhance the speed of transfer by reducing the number of read operations by the central processing unit of the free count register 45.” Rosenthal ’930 at col. 22, ll. 27–52. “means for informing a source of data of stages of the FIFO circuit available to store data including a register for storing a value indicating the number of stages of the FIFO circuit which are available. 2. A flow control circuit as claimed in claim 1 in which the means for informing a source of data of stages of the FIFO circuit available to store data includes means for furnishing the value in the register to an application program.” Rosenthal ’930 at col. 24, ll. 12–19.

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	“9. A method for controlling the flow of data in a data transfer path of a computer having a first-in first-out (FIFO) buffer with a plurality of stages positioned between a computer input-output bus and input/output devices, each stage providing storage for both data and a virtual address which are translated to provide a physical address for commands decoupled from application programs originating the commands, comprising the steps of: detecting a value indicating an amount of space available for data in the FIFO buffer, transferring to the FIFO buffer an amount of data up to an amount determine from the value detected, and detecting another value indicating an amount of space available for data in the FIFO buffer before transferring additional data to the FIFO buffer.” Rosenthal ’930 at col. 24, ll. 51–65.
[1f] computer storage storing instructions, which when executed by said at least one second processing element:	Rosenthal ’930 teaches computer storage storing instructions executed by said at least one second processing element, including at least the following exemplary citations. For example, Rosenthal ’930 teaches computer storage storing instructions that are executed by said at least one second processing element, by disclosing, as non-limiting examples, application programs, e.g. resource manager that are executed from computer storage. The processors execute those instructions stored in the computer storage. <i>See, e.g.:</i> “A digital system which uses an arrangement of one or more parallel FIFO buffers in which each FIFO buffer handles data from only one application program at any time. In order to assure that no data written to a FIFO buffer by an application program will overflow the FIFO buffer, each FIFO buffer includes a flow control register which must be read by the processing unit running the application before writing data to an input/output device. The register stores a value which indicates the amount of space available in the FIFO buffer to which data may be written. Reading this register tells the application program how much data may be written without running the risk of overflowing the data storage area which the input/output device has available.” Rosenthal ’930 at Abstract. “It is another object of the present invention to provide a flow control system for an architecture which utilizes FIFO buffers for buffering write operations. These and other objects of the present invention are realized in a system which uses an arrangement of parallel FIFO buffers in which each FIFO buffer handles data from only one application program at any time. In order to assure that no data written to a FIFO buffer by an application program will overflow the FIFO

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	buffer, each FIFO buffer includes a flow control register which must be read by the central processing unit running the application before writing data to an input/output device. The register stores a value which indicates the amount of space available in the FIFO buffer to which data may be written. Reading this register tells the application program how much data may be written without running the risk of overflowing the data storage area which the input/output device has available. Provided that no application program ever writes more than the data for which it has permission, no data will be lost.” Rosenthal ’930 at col. 2, ll. 15–33. “Thus, as may be seen, the resource manager is a piece of software which is associated with the input/output control unit 29 and determines that the input/output control unit 29 functions correctly. It maintains a database of data structures which represent the various input/output devices and the context that those devices require to function correctly. It fills the translation table 38, does the necessary context switching for initializing the physical devices, provides routines for less used input/output operations which input/output devices may invoke through interrupts, and does other things required to run the input/output control unit 29. The resource manager may be thought of as part of the operating system and takes the place of the device driver used in a conventional input/output system. The resource manager maps in a part of the physical hardware of the input/output control unit 29 called the privileged address space. This space is distinct from the FIFO unit. Unlike the application programs operating with input/output devices, the resource manager both reads and writes this address space to perform its various tasks such as context switching. Unlike all of the device drivers of the prior art, the resource manager accomplishes its functions after the hardware of the input/output control unit 29 has been directly addressed by an application program rather than before. Moreover, in the overall operation of the input/output control unit 29, the resource manager is used infrequently compared to the hardware portions of the input/output control unit 29 since the resource manager attends only to creation operations, the various software implementations of methods, and unsafe operations.” Rosenthal ’930 at col. 15, ll. 35–64. “The use of the flow control register 45 allows an application program presently using the resources of the input/output control unit 29 to proceed without causing the overflow of the FIFO unit 31. Whenever the application has written commands totaling the amount previously read from the register 45, it must read the register again. This is especially useful for application programs which may wish to transfer a series of commands in sequence, possibly using the burst transfer mode of operation provided by many of the modern buses. By ascertaining the space available in the FIFO buffer 39 before transferring a sequence of commands, the application can know that sufficient space will be available for the operation. Provided that no application

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	program ever writes more than the data for which it has permission, no data will be lost.” Rosenthal ’930 at col. 17, ll. 4–18. “Various operating conditions can cause the flow control register 45 associated with a buffer 39 to store different values. If the central processing unit is writing at a rate faster than the input/output device can handle the commands, then a FIFO buffer 39 which is initially empty and provides a maximum free count value when read by the central processing unit will provide a lower free count number after the amount of data first indicated by the free count register has been sent. This will occur because the FIFO buffer 39 will empty more slowly than it is filled. In such a case, the next value read by the application program after sending the amount of data designated by the free count register will be a smaller number. The application program may again send this amount of data and be sure that the FIFO buffer will not overflow. On the other hand, if the central processing unit is writing at a rate slower than the input/output device can handle the commands, then a FIFO buffer which is initially empty and provides a maximum free count value when read by the central processing unit, will provide the same free count number after the amount of data first indicated by the flow control register has been sent. This will occur because the FIFO will empty more rapidly than it is filled and will always be empty when the register 45 is read by the central processing unit.” Rosenthal ’930 at col. 17, ll. 19–42. “In utilizing the free count value, the application program should deduct the value of any data which it has already written before each new write operation commences. In this manner, the application program knows at any time the amount of data which it may still write to the input/output device before another read of the free count register will be required. This allows a number of write operations to occur before a read of the register 45 (which significantly slows operation of the overall system) need occur. This also assures that any writes which might be presently in the write channel on the input/output bus are accounted for in determining the actual value of the free count left. It should be noted that this read of the free count register is the only read necessary in dealing with the input/output control unit 29. It would be possible either to interrupt the central processing unit if a FIFO buffer 39 filled or to use the DMA unit 35 to transfer the value in the free count register to the central processing unit instead of reading the value, but in the particular embodiment reading the free count register appears to be the fastest operation.” Rosenthal ’930 at col. 17, l. 52 – col. 18, l. 4. “In one embodiment, the application program may obtain a free count value indicating the number of free entries in a FIFO buffer 39 by reading from a designated method offset in any of the sub-areas of the area mapped to the application program. The value read is that stored in the flow control register 45 associated

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	with the particular FIFO buffer 39. This value is then placed on the bus when the central processing unit provides the appropriate control signals to implement the read operation. The application may write up to the amount of data designated by this free count value without further testing and be certain that overflow of the FIFO buffer 39 will not occur.” Rosenthal ’930 at col. 18, ll. 16–27. “As with the embodiments which use one FIFO buffer for each application area, a free count register 45 and a puller are included with each FIFO buffer 39 as a part of the input/output control unit 29. The register 45 stores an indication of the number of available entries in each FIFO buffer 39 of the FIFO unit 31 and allows a requirement to be placed on an application program attempting to access the input/output control unit 29 that it first determine whether space is available in the FIFO unit 31. It also allows an application presently using the resources of the input/output control unit 29 to proceed without overflowing the FIFO unit 31. The application program may obtain a boundary value indicating the number of free entries in the FIFO buffer by reading from a designated offset in any of the sub-areas of its mapped area. The application may write up to the amount of data designated by this boundary value without further testing and be certain that overflow of the FIFO unit will not occur. As with the multiple FIFO buffer embodiments, the FIFO buffer arrangements of the described embodiment make use of a memory runout area; such a memory runout area assures that a FIFO buffer may be used to cache commands from a plurality of application programs. The use of a memory runout area also allows the system to make use of lying in order to enhance the speed of transfer by reducing the number of read operations by the central processing unit of the free count register 45.” Rosenthal ’930 at col. 22, ll. 27–52. “9. A method for controlling the flow of data in a data transfer path of a computer having a first-in first-out (FIFO) buffer with a plurality of stages positioned between a computer input-output bus and input/output devices, each stage providing storage for both data and a virtual address which are translated to provide a physical address for commands decoupled from application programs originating the commands, comprising the steps of: detecting a value indicating an amount of space available for data in the FIFO buffer, transferring to the FIFO buffer an amount of data up to an amount determine from the value detected, and detecting another value indicating an amount of space available for data in the FIFO buffer before transferring additional data to the FIFO buffer.” Rosenthal ’930 at col. 24, ll. 51–65.

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
[1g] accesses said reservation register and reads said stored value;	Rosenthal '930 teaches "accesses said reservation register and reads said stored value," including at least the following exemplary citations. For example, Rosenthal '930 teaches that the second processing element executes instructions that causes it to access said reservation register and reads said stored value, by disclosing, as non-limiting examples, accessing the flow control register to determine if there is sufficient space. The processor that is running the application program accesses and reads the stored value in the reservation register. <i>See, e.g.:</i> "A digital system which uses an arrangement of one or more parallel FIFO buffers in which each FIFO buffer handles data from only one application program at any time. In order to assure that no data written to a FIFO buffer by an application program will overflow the FIFO buffer, each FIFO buffer includes a flow control register which must be read by the processing unit running the application before writing data to an input/output device. The register stores a value which indicates the amount of space available in the FIFO buffer to which data may be written. Reading this register tells the application program how much data may be written without running the risk of overflowing the data storage area which the input/output device has available." Rosenthal '930 at Abstract. "It is another object of the present invention to provide a flow control system for an architecture which utilizes FIFO buffers for buffering write operations. These and other objects of the present invention are realized in a system which uses an arrangement of parallel FIFO buffers in which each FIFO buffer handles data from only one application program at any time. In order to assure that no data written to a FIFO buffer by an application program will overflow the FIFO buffer, each FIFO buffer includes a flow control register which must be read by the central processing unit running the application before writing data to an input/output device. The register stores a value which indicates the amount of space available in the FIFO buffer to which data may be written. Reading this register tells the application program how much data may be written without running the risk of overflowing the data storage area which the input/output device has available. Provided that no application program ever writes more than the data for which it has permission, no data will be lost." Rosenthal '930 at col. 2, ll. 15–33. "In one embodiment of the new architecture, a flow control register 45 is included for each FIFO buffer 39 of the input/output control unit 29. The register 45 stores an indication of the number of available spaces for commands in the associated FIFO buffer 39. The use of this flow control register 45 allows a requirement to be placed on an application program attempting an access of the input/output control unit 29 that it first determine whether space is available in the FIFO unit 31 before it writes any command. Before writing any data, the application program must determine the amount of "free" space to which it may write. This it does

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	by having the central processing unit read the value in the flow control register 45 associated with the FIFO buffer 39 being utilized. Once this value has been read, the application program may send up to that amount of data before it need test again to determine whether free space remains to which it may send more data.” Rosenthal ’930 at col. 16, ll. 33–49. “The use of the flow control register 45 allows an application program presently using the resources of the input/output control unit 29 to proceed without causing the overflow of the FIFO unit 31. Whenever the application has written commands totaling the amount previously read from the register 45, it must read the register again. This is especially useful for application programs which may wish to transfer a series of commands in sequence, possibly using the burst transfer mode of operation provided by many of the modern buses. By ascertaining the space available in the FIFO buffer 39 before transferring a sequence of commands, the application can know that sufficient space will be available for the operation. Provided that no application program ever writes more than the data for which it has permission, no data will be lost.” Rosenthal ’930 at col. 17, ll. 4–18. “Various operating conditions can cause the flow control register 45 associated with a buffer 39 to store different values. If the central processing unit is writing at a rate faster than the input/output device can handle the commands, then a FIFO buffer 39 which is initially empty and provides a maximum free count value when read by the central processing unit will provide a lower free count number after the amount of data first indicated by the free count register has been sent. This will occur because the FIFO buffer 39 will empty more slowly than it is filled. In such a case, the next value read by the application program after sending the amount of data designated by the free count register will be a smaller number. The application program may again send this amount of data and be sure that the FIFO buffer will not overflow. On the other hand, if the central processing unit is writing at a rate slower than the input/output device can handle the commands, then a FIFO buffer which is initially empty and provides a maximum free count value when read by the central processing unit, will provide the same free count number after the amount of data first indicated by the flow control register has been sent. This will occur because the FIFO will empty more rapidly than it is filled and will always be empty when the register 45 is read by the central processing unit.” Rosenthal ’930 at col. 17, ll. 19–42. “In utilizing the free count value, the application program should deduct the value of any data which it has already written before each new write operation commences. In this manner, the application program knows at any time the amount of data which it may still write to the input/output device before another read of the free count register will be required. This allows a number of write operations to occur before a read of the

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	register 45 (which significantly slows operation of the overall system) need occur. This also assures that any writes which might be presently in the write channel on the input/output bus are accounted for in determining the actual value of the free count left. It should be noted that this read of the free count register is the only read necessary in dealing with the input/output control unit 29. It would be possible either to interrupt the central processing unit if a FIFO buffer 39 filled or to use the DMA unit 35 to transfer the value in the free count register to the central processing unit instead of reading the value, but in the particular embodiment reading the free count register appears to be the fastest operation.” Rosenthal ’930 at col. 17, l. 52 – col. 18, l. 4. “In one embodiment, the application program may obtain a free count value indicating the number of free entries in a FIFO buffer 39 by reading from a designated method offset in any of the sub-areas of the area mapped to the application program. The value read is that stored in the flow control register 45 associated with the particular FIFO buffer 39. This value is then placed on the bus when the central processing unit provides the appropriate control signals to implement the read operation. The application may write up to the amount of data designated by this free count value without further testing and be certain that overflow of the FIFO buffer 39 will not occur.” Rosenthal ’930 at col. 18, ll. 16–27. “As with the embodiments which use one FIFO buffer for each application area, a free count register 45 and a puller are included with each FIFO buffer 39 as a part of the input/output control unit 29. The register 45 stores an indication of the number of available entries in each FIFO buffer 39 of the FIFO unit 31 and allows a requirement to be placed on an application program attempting to access the input/output control unit 29 that it first determine whether space is available in the FIFO unit 31. It also allows an application presently using the resources of the input/output control unit 29 to proceed without overflowing the FIFO unit 31. The application program may obtain a boundary value indicating the number of free entries in the FIFO buffer by reading from a designated offset in any of the sub-areas of its mapped area. The application may write up to the amount of data designated by this boundary value without further testing and be certain that overflow of the FIFO unit will not occur. As with the multiple FIFO buffer embodiments, the FIFO buffer arrangements of the described embodiment make use of a memory runout area; such a memory runout area assures that a FIFO buffer may be used to cache commands from a plurality of application programs. The use of a memory runout area also allows the system to make use of lying in order to enhance the speed of transfer by reducing the number of read operations by the central processing unit of the free count register 45.” Rosenthal ’930 at col. 22, ll. 27–52.

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	“means for informing a source of data of stages of the FIFO circuit available to store data including a register for storing a value indicating the number of stages of the FIFO circuit which are available. 2. A flow control circuit as claimed in claim 1 in which the means for informing a source of data of stages of the FIFO circuit available to store data includes means for furnishing the value in the register to an application program.” Rosenthal ’930 at col. 24, ll. 12–19. “9. A method for controlling the flow of data in a data transfer path of a computer having a first-in first-out (FIFO) buffer with a plurality of stages positioned between a computer input-output bus and input/output devices, each stage providing storage for both data and a virtual address which are translated to provide a physical address for commands decoupled from application programs originating the commands, comprising the steps of: detecting a value indicating an amount of space available for data in the FIFO buffer, transferring to the FIFO buffer an amount of data up to an amount determine from the value detected, and detecting another value indicating an amount of space available for data in the FIFO buffer before transferring additional data to the FIFO buffer.” Rosenthal ’930 at col. 24, ll. 51–65.
[1h] determines when said read value indicates available space in said input queue for said first processing element to issue a command;	Rosenthal ’930 teaches “determines when said read value indicates available space in said input queue for said first processing element to issue a command,” including at least the following exemplary citations. For example, Rosenthal ’930 teaches that the second processing element executes instructions that causes it to determine when said read value indicates available space in said input queue for said first processing element to issue a command, by disclosing, as non-limiting examples, application programs that read the flow control register to determine if there is sufficient space before writing. The processor that is running the application program determines if the stored value indicates available space in the input queue for the first processing element to issue a command. <i>See, e.g.:</i> “A digital system which uses an arrangement of one or more parallel FIFO buffers in which each FIFO buffer handles data from only one application program at any time. In order to assure that no data written to a FIFO buffer by an application program will overflow the FIFO buffer, each FIFO buffer includes a flow control register which must be read by the processing unit running the application before writing data to an input/output device. The register stores a value which indicates the amount of space available in the FIFO buffer to which data may be written. Reading this register tells the application program how much data may

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	be written without running the risk of overflowing the data storage area which the input/output device has available.” Rosenthal ’930 at Abstract. “It is another object of the present invention to provide a flow control system for an architecture which utilizes FIFO buffers for buffering write operations. These and other objects of the present invention are realized in a system which uses an arrangement of parallel FIFO buffers in which each FIFO buffer handles data from only one application program at any time. In order to assure that no data written to a FIFO buffer by an application program will overflow the FIFO buffer, each FIFO buffer includes a flow control register which must be read by the central processing unit running the application before writing data to an input/output device. The register stores a value which indicates the amount of space available in the FIFO buffer to which data may be written. Reading this register tells the application program how much data may be written without running the risk of overflowing the data storage area which the input/output device has available. Provided that no application program ever writes more than the data for which it has permission, no data will be lost.” Rosenthal ’930 at col. 2, ll. 15–33. “In one embodiment of the new architecture, a flow control register 45 is included for each FIFO buffer 39 of the input/output control unit 29. The register 45 stores an indication of the number of available spaces for commands in the associated FIFO buffer 39. The use of this flow control register 45 allows a requirement to be placed on an application program attempting an access of the input/output control unit 29 that it first determine whether space is available in the FIFO unit 31 before it writes any command. Before writing any data, the application program must determine the amount of "free" space to which it may write. This it does by having the central processing unit read the value in the flow control register 45 associated with the FIFO buffer 39 being utilized. Once this value has been read, the application program may send up to that amount of data before it need test again to determine whether free space remains to which it may send more data.” Rosenthal ’930 at col. 16, ll. 33–49. “The use of the flow control register 45 allows an application program presently using the resources of the input/output control unit 29 to proceed without causing the overflow of the FIFO unit 31. Whenever the application has written commands totaling the amount previously read from the register 45, it must read the register again. This is especially useful for application programs which may wish to transfer a series of commands in sequence, possibly using the burst transfer mode of operation provided by many of the modern buses. By ascertaining the space available in the FIFO buffer 39 before transferring a sequence of commands, the application can know that sufficient space will be available for the operation. Provided that no application

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	program ever writes more than the data for which it has permission, no data will be lost.” Rosenthal ’930 at col. 17, ll. 4–18. “Various operating conditions can cause the flow control register 45 associated with a buffer 39 to store different values. If the central processing unit is writing at a rate faster than the input/output device can handle the commands, then a FIFO buffer 39 which is initially empty and provides a maximum free count value when read by the central processing unit will provide a lower free count number after the amount of data first indicated by the free count register has been sent. This will occur because the FIFO buffer 39 will empty more slowly than it is filled. In such a case, the next value read by the application program after sending the amount of data designated by the free count register will be a smaller number. The application program may again send this amount of data and be sure that the FIFO buffer will not overflow. On the other hand, if the central processing unit is writing at a rate slower than the input/output device can handle the commands, then a FIFO buffer which is initially empty and provides a maximum free count value when read by the central processing unit, will provide the same free count number after the amount of data first indicated by the flow control register has been sent. This will occur because the FIFO will empty more rapidly than it is filled and will always be empty when the register 45 is read by the central processing unit.” Rosenthal ’930 at col. 17, ll. 19–42. “In utilizing the free count value, the application program should deduct the value of any data which it has already written before each new write operation commences. In this manner, the application program knows at any time the amount of data which it may still write to the input/output device before another read of the free count register will be required. This allows a number of write operations to occur before a read of the register 45 (which significantly slows operation of the overall system) need occur. This also assures that any writes which might be presently in the write channel on the input/output bus are accounted for in determining the actual value of the free count left. It should be noted that this read of the free count register is the only read necessary in dealing with the input/output control unit 29. It would be possible either to interrupt the central processing unit if a FIFO buffer 39 filled or to use the DMA unit 35 to transfer the value in the free count register to the central processing unit instead of reading the value, but in the particular embodiment reading the free count register appears to be the fastest operation.” Rosenthal ’930 at col. 17, l. 52 – col. 18, l. 4. “In one embodiment, the application program may obtain a free count value indicating the number of free entries in a FIFO buffer 39 by reading from a designated method offset in any of the sub-areas of the area mapped to the application program. The value read is that stored in the flow control register 45 associated

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	with the particular FIFO buffer 39. This value is then placed on the bus when the central processing unit provides the appropriate control signals to implement the read operation. The application may write up to the amount of data designated by this free count value without further testing and be certain that overflow of the FIFO buffer 39 will not occur.” Rosenthal ’930 at col. 18, ll. 16–27. “As with the embodiments which use one FIFO buffer for each application area, a free count register 45 and a puller are included with each FIFO buffer 39 as a part of the input/output control unit 29. The register 45 stores an indication of the number of available entries in each FIFO buffer 39 of the FIFO unit 31 and allows a requirement to be placed on an application program attempting to access the input/output control unit 29 that it first determine whether space is available in the FIFO unit 31. It also allows an application presently using the resources of the input/output control unit 29 to proceed without overflowing the FIFO unit 31. The application program may obtain a boundary value indicating the number of free entries in the FIFO buffer by reading from a designated offset in any of the sub-areas of its mapped area. The application may write up to the amount of data designated by this boundary value without further testing and be certain that overflow of the FIFO unit will not occur. As with the multiple FIFO buffer embodiments, the FIFO buffer arrangements of the described embodiment make use of a memory runout area; such a memory runout area assures that a FIFO buffer may be used to cache commands from a plurality of application programs. The use of a memory runout area also allows the system to make use of lying in order to enhance the speed of transfer by reducing the number of read operations by the central processing unit of the free count register 45.” Rosenthal ’930 at col. 22, ll. 27–52. “9. A method for controlling the flow of data in a data transfer path of a computer having a first-in first-out (FIFO) buffer with a plurality of stages positioned between a computer input-output bus and input/output devices, each stage providing storage for both data and a virtual address which are translated to provide a physical address for commands decoupled from application programs originating the commands, comprising the steps of: detecting a value indicating an amount of space available for data in the FIFO buffer, transferring to the FIFO buffer an amount of data up to an amount determine from the value detected, and detecting another value indicating an amount of space available for data in the FIFO buffer before transferring additional data to the FIFO buffer.” Rosenthal ’930 at col. 24, ll. 51–65.

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
[1i] notifies said first processing element to issue said command to said input queue; and	Rosenthal '930 teaches “notifies said first processing element to issue said command to said input queue,” including at least the following exemplary citations. For example, Rosenthal '930 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, reading the flow control register and determining if there is enough space. The processor running the application program notifies the first processing element to issue the command to the input queue. <i>See, e.g.:</i> “A digital system which uses an arrangement of one or more parallel FIFO buffers in which each FIFO buffer handles data from only one application program at any time. In order to assure that no data written to a FIFO buffer by an application program will overflow the FIFO buffer, each FIFO buffer includes a flow control register which must be read by the processing unit running the application before writing data to an input/output device. The register stores a value which indicates the amount of space available in the FIFO buffer to which data may be written. Reading this register tells the application program how much data may be written without running the risk of overflowing the data storage area which the input/output device has available.” Rosenthal '930 at Abstract. “It is another object of the present invention to provide a flow control system for an architecture which utilizes FIFO buffers for buffering write operations. These and other objects of the present invention are realized in a system which uses an arrangement of parallel FIFO buffers in which each FIFO buffer handles data from only one application program at any time. In order to assure that no data written to a FIFO buffer by an application program will overflow the FIFO buffer, each FIFO buffer includes a flow control register which must be read by the central processing unit running the application before writing data to an input/output device. The register stores a value which indicates the amount of space available in the FIFO buffer to which data may be written. Reading this register tells the application program how much data may be written without running the risk of overflowing the data storage area which the input/output device has available. Provided that no application program ever writes more than the data for which it has permission, no data will be lost.” Rosenthal '930 at col. 2, ll. 15–33. “In one embodiment of the new architecture, a flow control register 45 is included for each FIFO buffer 39 of the input/output control unit 29. The register 45 stores an indication of the number of available spaces for commands in the associated FIFO buffer 39. The use of this flow control register 45 allows a requirement to be placed on an application program attempting an access of the input/output control unit 29 that it first determine whether space is available in the FIFO unit 31 before it writes any command. Before writing any

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	data, the application program must determine the amount of "free" space to which it may write. This it does by having the central processing unit read the value in the flow control register 45 associated with the FIFO buffer 39 being utilized. Once this value has been read, the application program may send up to that amount of data before it need test again to determine whether free space remains to which it may send more data.” Rosenthal ’930 at col. 16, ll. 33–49. “The use of the flow control register 45 allows an application program presently using the resources of the input/output control unit 29 to proceed without causing the overflow of the FIFO unit 31. Whenever the application has written commands totaling the amount previously read from the register 45, it must read the register again. This is especially useful for application programs which may wish to transfer a series of commands in sequence, possibly using the burst transfer mode of operation provided by many of the modern buses. By ascertaining the space available in the FIFO buffer 39 before transferring a sequence of commands, the application can know that sufficient space will be available for the operation. Provided that no application program ever writes more than the data for which it has permission, no data will be lost.” Rosenthal ’930 at col. 17, ll. 4–18. “Various operating conditions can cause the flow control register 45 associated with a buffer 39 to store different values. If the central processing unit is writing at a rate faster than the input/output device can handle the commands, then a FIFO buffer 39 which is initially empty and provides a maximum free count value when read by the central processing unit will provide a lower free count number after the amount of data first indicated by the free count register has been sent. This will occur because the FIFO buffer 39 will empty more slowly than it is filled. In such a case, the next value read by the application program after sending the amount of data designated by the free count register will be a smaller number. The application program may again send this amount of data and be sure that the FIFO buffer will not overflow. On the other hand, if the central processing unit is writing at a rate slower than the input/output device can handle the commands, then a FIFO buffer which is initially empty and provides a maximum free count value when read by the central processing unit, will provide the same free count number after the amount of data first indicated by the flow control register has been sent. This will occur because the FIFO will empty more rapidly than it is filled and will always be empty when the register 45 is read by the central processing unit.” Rosenthal ’930 at col. 17, ll. 19–42. “In one embodiment, the application program may obtain a free count value indicating the number of free entries in a FIFO buffer 39 by reading from a designated method offset in any of the sub-areas of the area mapped to the application program. The value read is that stored in the flow control register 45 associated

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	with the particular FIFO buffer 39. This value is then placed on the bus when the central processing unit provides the appropriate control signals to implement the read operation. The application may write up to the amount of data designated by this free count value without further testing and be certain that overflow of the FIFO buffer 39 will not occur.” Rosenthal ’930 at col. 18, ll. 16–27. “As with the embodiments which use one FIFO buffer for each application area, a free count register 45 and a puller are included with each FIFO buffer 39 as a part of the input/output control unit 29. The register 45 stores an indication of the number of available entries in each FIFO buffer 39 of the FIFO unit 31 and allows a requirement to be placed on an application program attempting to access the input/output control unit 29 that it first determine whether space is available in the FIFO unit 31. It also allows an application presently using the resources of the input/output control unit 29 to proceed without overflowing the FIFO unit 31. The application program may obtain a boundary value indicating the number of free entries in the FIFO buffer by reading from a designated offset in any of the sub-areas of its mapped area. The application may write up to the amount of data designated by this boundary value without further testing and be certain that overflow of the FIFO unit will not occur. As with the multiple FIFO buffer embodiments, the FIFO buffer arrangements of the described embodiment make use of a memory runout area; such a memory runout area assures that a FIFO buffer may be used to cache commands from a plurality of application programs. The use of a memory runout area also allows the system to make use of lying in order to enhance the speed of transfer by reducing the number of read operations by the central processing unit of the free count register 45.” Rosenthal ’930 at col. 22, ll. 27–52. “9. A method for controlling the flow of data in a data transfer path of a computer having a first-in first-out (FIFO) buffer with a plurality of stages positioned between a computer input-output bus and input/output devices, each stage providing storage for both data and a virtual address which are translated to provide a physical address for commands decoupled from application programs originating the commands, comprising the steps of: detecting a value indicating an amount of space available for data in the FIFO buffer, transferring to the FIFO buffer an amount of data up to an amount determine from the value detected, and detecting another value indicating an amount of space available for data in the FIFO buffer before transferring additional data to the FIFO buffer.” Rosenthal ’930 at col. 24, ll. 51–65.

Exhibit 596-16

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.	Rosenthal '930 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, Rosenthal '930 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, that the processor running the application program receives the notification, issues the command, and receives a corresponding response from the output queue. <i>See, e.g.:</i> “A digital system which uses an arrangement of one or more parallel FIFO buffers in which each FIFO buffer handles data from only one application program at any time. In order to assure that no data written to a FIFO buffer by an application program will overflow the FIFO buffer, each FIFO buffer includes a flow control register which must be read by the processing unit running the application before writing data to an input/output device. The register stores a value which indicates the amount of space available in the FIFO buffer to which data may be written. Reading this register tells the application program how much data may be written without running the risk of overflowing the data storage area which the input/output device has available.” Rosenthal '930 at Abstract. “It is another object of the present invention to provide a flow control system for an architecture which utilizes FIFO buffers for buffering write operations. These and other objects of the present invention are realized in a system which uses an arrangement of parallel FIFO buffers in which each FIFO buffer handles data from only one application program at any time. In order to assure that no data written to a FIFO buffer by an application program will overflow the FIFO buffer, each FIFO buffer includes a flow control register which must be read by the central processing unit running the application before writing data to an input/output device. The register stores a value which indicates the amount of space available in the FIFO buffer to which data may be written. Reading this register tells the application program how much data may be written without running the risk of overflowing the data storage area which the input/output device has available. Provided that no application program ever writes more than the data for which it has permission, no data will be lost.” Rosenthal '930 at col. 2, ll. 15–33. “When the operation involves a device directly associated with the control unit 29 on its device bus 34, the commands after the first commands (creating the new data structure, attaching its new virtual name, providing any necessary device context, and creating the address translation) are sent by hardware directly to that device

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	for execution. If the command requires that data be transferred to or from the application, the input/output device performs the transfer using the DMA unit 35. Upon the return of data in response to a command, the DMA unit 35 of the control unit 29 responds by transferring the data to main memory and notifying the central processing unit in a separate DMA operation of the existence of the data so that no local bus read operation by the central processing unit 21 is necessary to ascertain the result of the operation or to receive the data provided.” Rosenthal ’930 at col. 7, ll. 30–45. “In one embodiment of the new architecture, a flow control register 45 is included for each FIFO buffer 39 of the input/output control unit 29. The register 45 stores an indication of the number of available spaces for commands in the associated FIFO buffer 39. The use of this flow control register 45 allows a requirement to be placed on an application program attempting an access of the input/output control unit 29 that it first determine whether space is available in the FIFO unit 31 before it writes any command. Before writing any data, the application program must determine the amount of "free" space to which it may write. This it does by having the central processing unit read the value in the flow control register 45 associated with the FIFO buffer 39 being utilized. Once this value has been read, the application program may send up to that amount of data before it need test again to determine whether free space remains to which it may send more data.” Rosenthal ’930 at col. 16, ll. 33–49. “The use of the flow control register 45 allows an application program presently using the resources of the input/output control unit 29 to proceed without causing the overflow of the FIFO unit 31. Whenever the application has written commands totaling the amount previously read from the register 45, it must read the register again. This is especially useful for application programs which may wish to transfer a series of commands in sequence, possibly using the burst transfer mode of operation provided by many of the modern buses. By ascertaining the space available in the FIFO buffer 39 before transferring a sequence of commands, the application can know that sufficient space will be available for the operation. Provided that no application program ever writes more than the data for which it has permission, no data will be lost.” Rosenthal ’930 at col. 17, ll. 4–18. “Various operating conditions can cause the flow control register 45 associated with a buffer 39 to store different values. If the central processing unit is writing at a rate faster than the input/output device can handle the commands, then a FIFO buffer 39 which is initially empty and provides a maximum free count value when read by the central processing unit will provide a lower free count number after the amount of data first indicated by the free count register has been sent. This will occur because the FIFO buffer 39 will empty more slowly than it is filled. In such a case, the next value read by the application program after sending the

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	amount of data designated by the free count register will be a smaller number. The application program may again send this amount of data and be sure that the FIFO buffer will not overflow. On the other hand, if the central processing unit is writing at a rate slower than the input/output device can handle the commands, then a FIFO buffer which is initially empty and provides a maximum free count value when read by the central processing unit, will provide the same free count number after the amount of data first indicated by the flow control register has been sent. This will occur because the FIFO will empty more rapidly than it is filled and will always be empty when the register 45 is read by the central processing unit.” Rosenthal ’930 at col. 17, ll. 19–42. “In one embodiment, the application program may obtain a free count value indicating the number of free entries in a FIFO buffer 39 by reading from a designated method offset in any of the sub-areas of the area mapped to the application program. The value read is that stored in the flow control register 45 associated with the particular FIFO buffer 39. This value is then placed on the bus when the central processing unit provides the appropriate control signals to implement the read operation. The application may write up to the amount of data designated by this free count value without further testing and be certain that overflow of the FIFO buffer 39 will not occur.” Rosenthal ’930 at col. 18, ll. 16–27. “As with the embodiments which use one FIFO buffer for each application area, a free count register 45 and a puller are included with each FIFO buffer 39 as a part of the input/output control unit 29. The register 45 stores an indication of the number of available entries in each FIFO buffer 39 of the FIFO unit 31 and allows a requirement to be placed on an application program attempting to access the input/output control unit 29 that it first determine whether space is available in the FIFO unit 31. It also allows an application presently using the resources of the input/output control unit 29 to proceed without overflowing the FIFO unit 31. The application program may obtain a boundary value indicating the number of free entries in the FIFO buffer by reading from a designated offset in any of the sub-areas of its mapped area. The application may write up to the amount of data designated by this boundary value without further testing and be certain that overflow of the FIFO unit will not occur. As with the multiple FIFO buffer embodiments, the FIFO buffer arrangements of the described embodiment make use of a memory runout area; such a memory runout area assures that a FIFO buffer may be used to cache commands from a plurality of application programs. The use of a memory runout area also allows the system to make use of lying in order to enhance the speed of transfer by reducing the number of read operations by the central processing unit of the free count register 45.” Rosenthal ’930 at col. 22, ll. 27–52.

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	“9. A method for controlling the flow of data in a data transfer path of a computer having a first-in first-out (FIFO) buffer with a plurality of stages positioned between a computer input-output bus and input/output devices, each stage providing storage for both data and a virtual address which are translated to provide a physical address for commands decoupled from application programs originating the commands, comprising the steps of: detecting a value indicating an amount of space available for data in the FIFO buffer, transferring to the FIFO buffer an amount of data up to an amount determine from the value detected, and detecting another value indicating an amount of space available for data in the FIFO buffer before transferring additional data to the FIFO buffer.” Rosenthal ’930 at col. 24, ll. 51–65.
2. The system of claim 1, wherein said input queue is a first-in-first-out (FIFO) buffer.	Rosenthal ’930 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, Rosenthal ’930 teaches wherein said input queue is a first-in-first-out (FIFO) buffer, by disclosing, as non-limiting examples, a FIFO buffer. <i>See, e.g.:</i> “A digital system which uses an arrangement of one or more parallel FIFO buffers in which each FIFO buffer handles data from only one application program at any time. In order to assure that no data written to a FIFO buffer by an application program will overflow the FIFO buffer, each FIFO buffer includes a flow control register which must be read by the processing unit running the application before writing data to an input/output device. The register stores a value which indicates the amount of space available in the FIFO buffer to which data may be written. Reading this register tells the application program how much data may be written without running the risk of overflowing the data storage area which the input/output device has available.” Rosenthal ’930 at Abstract.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	Figure 1 Rosenthal '930 at Figure 1.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	The diagram illustrates a system architecture. At the top, a horizontal line represents Bus 27. Below it, a large rectangular block represents the Input/Output Control Unit 29. Inside this unit, at the top, is a 'Decode' block 30. Below 'Decode' 30 is a 'Pusher' block 53. To the left of 'Pusher' 53 is a block containing 'MMU 50' and 'DMA 35'. Below 'MMU 50' is a block labeled '52'. To the right of 'Pusher' 53 is a large grid-like structure with rows and columns, containing various numbered blocks: 42, 7, 3, 11, 32, 45, 31, 39, and 40. Below the grid is a 'Resource Manager' block (dashed outline) and a 'Translation Table' block 38. Below the 'Input/Output Control Unit 29' is another horizontal line representing Bus 34. Below Bus 34 are three blocks: 'Disk Controller' 32, 'Graphics Output Controller Memory' 33, and 'Sound Generator' 37. Arrows indicate data flow between these components and the buses. Figure 3 Rosenthal '930 at Figure 3.

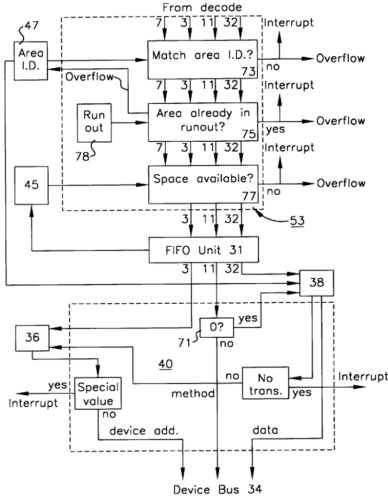
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 The flowchart in Figure 7 illustrates a data processing system. It begins with a 'Device Bus 34' at the bottom, which feeds into a 'FIFO Unit 31'. The FIFO unit has two outputs: one labeled '3' and another labeled '11 32'. The '3' output goes to a decision block '36'. The '11 32' output goes to a decision block '40'. From block '36', a 'yes' path leads to a 'Special value' block, which then leads to an 'Interrupt' signal. A 'no' path from '36' leads to a 'device add.' block. From block '40', a 'yes' path leads to a 'method' block, which then leads to a 'data' block. A 'no' path from '40' leads to a 'No trans.' block, which then leads to an 'Interrupt' signal. The 'device add.' and 'data' blocks both lead to a 'Device Bus 34' at the bottom. The 'method' and 'data' blocks both lead to a 'FIFO Unit 31' at the bottom. The 'FIFO Unit 31' has two inputs: one labeled '3' and another labeled '11 32'. The '3' input goes to a decision block '45'. The '11 32' input goes to a decision block '53'. From block '45', a 'yes' path leads to a 'Space available?' block, which then leads to an 'Interrupt' signal. A 'no' path from '45' leads to a 'Run out' block, which then leads to an 'Overflow' signal. From block '53', a 'yes' path leads to an 'Area already in runout?' block, which then leads to an 'Interrupt' signal. A 'no' path from '53' leads to a 'Match area I.D.?' block, which then leads to an 'Overflow' signal. The 'Match area I.D.?' block has two inputs: one labeled '7' and another labeled '31 11 32'. The '7' input goes to a decision block '47'. The '31 11 32' input goes to a decision block '73'. From block '47', a 'yes' path leads to an 'Area I.D.' block, which then leads to an 'Overflow' signal. A 'no' path from '47' leads to a 'From decode' block, which then leads to an 'Interrupt' signal. From block '73', a 'yes' path leads to an 'Area I.D.' block, which then leads to an 'Overflow' signal. A 'no' path from '73' leads to a 'From decode' block, which then leads to an 'Interrupt' signal. Figure 7 Rosenthal '930 at Figure 7. “Since most central processing unit accesses to input/output devices are write operations, the designers of systems and input/output devices attempt to decouple the central processing unit and input/output devices as far as write operations are concerned by implementing write queues using first-in first-out (FIFO) write buffers. These buffers may appear at various places in a particular implementation: as a part of the central processing unit, as part of a bridge chip, or as part of an input/output device.” Rosenthal '930 at col. 1, ll. 45–53. “One problem raised in systems using FIFO buffers is that an input/output device and the buffers supplying it must accept all information written to them over the input/output bus. Although some input/output buses allow devices to "hold off" writes, that is, delay the completion of the write operation until the device has enough resources available to store the data, there is always a limit to how long a write can be held off. If a write is held off too long the data will be lost. In the limit, the input/output device has no alternative but to store all data written to it. In a system utilizing FIFO buffers for storage of this data at the input/output device, the FIFO buffers must ultimately store the data.” Rosenthal '930 at col. 1, ll. 54–65.

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	“It is another object of the present invention to provide a flow control system for an architecture which utilizes FIFO buffers for buffering write operations. These and other objects of the present invention are realized in a system which uses an arrangement of parallel FIFO buffers in which each FIFO buffer handles data from only one application program at any time. In order to assure that no data written to a FIFO buffer by an application program will overflow the FIFO buffer, each FIFO buffer includes a flow control register which must be read by the central processing unit running the application before writing data to an input/output device. The register stores a value which indicates the amount of space available in the FIFO buffer to which data may be written. Reading this register tells the application program how much data may be written without running the risk of overflowing the data storage area which the input/output device has available. Provided that no application program ever writes more than the data for which it has permission, no data will be lost.” Rosenthal ’930 at col. 2, ll. 15–33. “In the architecture of system 22, an input/output control unit 29 is shown joined to the input/output bus 27. The control unit 29 includes a hardware FIFO unit 31 for receiving incoming commands addressed to the input/output devices on a device bus 34.” Rosenthal ’930 at col. 4, ll. 60–64. “In one embodiment, the FIFO unit 31 includes a plurality of FIFO buffers 39 (see FIG. 3). When an application program desires to write to an input/output device on the device bus 34, it addresses that device. Decoding circuitry decodes the address by reviewing a number of the highest order bits decoded by the chip sufficient to indicate a unique portion of the input/output address space assigned to an application program and places the command in the appropriate FIFO buffer 39 for that application program. Each FIFO buffer 39 handles commands only from the application program to which the address area has been mapped. An arrangement using a smaller number of FIFO buffers 39 than one buffer for each application program (as few as a single buffer 39) could also be utilized as long as the address of the FIFO buffer is mapped to only one application program at a time.” Rosenthal ’930 at col. 5, ll. 51–66. “The unit 29 receives physical addresses furnished by the memory management unit and virtual names furnished by application programs for operations to be performed which have been sent to the FIFO unit 31 and controls the translation of those virtual names for all input/output devices. The hardware unit 29 includes the device bus 34 to which the individual input/output devices such as a disk controller 32, a graphics output controller 33, and a sound generator 37 are shown joined. The unit 29 also includes a DMA unit 35 which is adapted to transfer data between the individual input/output devices and main memory for use by the central processing unit or other components of the system.” Rosenthal ’930 at col. 5, l. 67 – col. 6, l. 10.

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	“FIG. 3 is a block diagram illustrating details of the input/output control unit 29 including the device bus 34 and the input/output devices arranged on that bus. As described above, the input/output control unit 29 includes a decode unit 30 which receives commands directly from the input/output bus 27 and transfers the commands to a pusher circuit 53 which transfers the command to one of the FIFO buffers 39 of the FIFO unit 31. The FIFO unit 31 stores the data along with the addresses for each of the commands being transferred to all of the input/output devices associated with the input/output control unit 29. The buffers 39 of the FIFO unit replace the individual data registers used by the input/output devices of the prior art. However, in contrast to the registers used by the prior art for communication on the input/output bus 27, the FIFO unit 31 allows a very much larger number of commands to be processed much more rapidly and facilitates the asynchronous operations of the input/output devices and the central processing unit. In one embodiment, the FIFO unit 31 includes 128 individual FIFO buffers 39 each of which has thirty-two stages. The use of 128 individual FIFO buffers 39 allows a FIFO buffer to be assigned to each of 128 different application programs which may be running on the central processing unit for the use of that application program alone. The thirty-two stages of each buffer 39 allow any FIFO buffer to hold thirty-two individual serially ordered commands at any time. Although each of the stages of the FIFO unit 31 holds (along with the address bits) the data held by an individual register of a typical prior art input/output device, a particular system has the ability to store commands for over 16 thousand 32 bit registers for each of 128 different application programs which may map different addresses decoded by the input/output control unit 29.” Rosenthal ’930 at col. 8, ll. 21–53. “Thus, as may be seen, the resource manager is a piece of software which is associated with the input/output control unit 29 and determines that the input/output control unit 29 functions correctly. It maintains a database of data structures which represent the various input/output devices and the context that those devices require to function correctly. It fills the translation table 38, does the necessary context switching for initializing the physical devices, provides routines for less used input/output operations which input/output devices may invoke through interrupts, and does other things required to run the input/output control unit 29. The resource manager may be thought of as part of the operating system and takes the place of the device driver used in a conventional input/output system. The resource manager maps in a part of the physical hardware of the input/output control unit 29 called the privileged address space. This space is distinct from the FIFO unit. Unlike the application programs operating with input/output devices, the resource manager both reads and writes this address space to perform its various tasks such as context switching. Unlike all of the device drivers of the prior art, the resource manager accomplishes its functions after the hardware of the input/output control unit 29 has been directly addressed by an application program rather than before. Moreover, in the overall operation of the input/output control unit 29, the resource manager is used infrequently compared to

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	the hardware portions of the input/output control unit 29 since the resource manager attends only to creation operations, the various software implementations of methods, and unsafe operations.” Rosenthal ’930 at col. 15, ll. 35–64. “The use of the flow control register 45 allows an application program presently using the resources of the input/output control unit 29 to proceed without causing the overflow of the FIFO unit 31. Whenever the application has written commands totaling the amount previously read from the register 45, it must read the register again. This is especially useful for application programs which may wish to transfer a series of commands in sequence, possibly using the burst transfer mode of operation provided by many of the modern buses. By ascertaining the space available in the FIFO buffer 39 before transferring a sequence of commands, the application can know that sufficient space will be available for the operation. Provided that no application program ever writes more than the data for which it has permission, no data will be lost.” Rosenthal ’930 at col. 17, ll. 4–18. “One embodiment of the new architecture provides a means for attaining almost the same speed as is possible with 128 FIFO buffers but much more economically. Instead of 128 individual FIFO buffers 39, the described embodiment utilizes a smaller number of FIFO buffers 39 than the number of areas into which the input/output address space may be divided and thus the number of application programs which may be run. Utilizing a smaller number of FIFO buffers causes each FIFO buffer to function as a cache for commands which may be sent by any of the application programs. In the preferred embodiment, one FIFO buffer 39 is used for each processor which may be writing to the input/output control unit 29. For example, if the computer utilizes two processors arranged as symmetrical multi-processors, then two individual FIFO buffers are utilized. If, as is usually the case, only a single central processing unit is used, then a single FIFO buffer 39 may in the limit act as a cache for commands from all application programs which may be running on the central processing unit. Such an arrangement is illustrated in FIG. 6. Of course, in either case more FIFO buffers might be used.” Rosenthal ’930 at col. 21, l. 53 – col. 22, l. 6. “As with the embodiments which use one FIFO buffer for each application area, a free count register 45 and a puller are included with each FIFO buffer 39 as a part of the input/output control unit 29. The register 45 stores an indication of the number of available entries in each FIFO buffer 39 of the FIFO unit 31 and allows a requirement to be placed on an application program attempting to access the input/output control unit 29 that it first determine whether space is available in the FIFO unit 31. It also allows an application presently using the resources of the input/output control unit 29 to proceed without overflowing the FIFO unit 31. The application program may obtain a boundary value indicating the number of free entries in the FIFO buffer by

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	reading from a designated offset in any of the sub-areas of its mapped area. The application may write up to the amount of data designated by this boundary value without further testing and be certain that overflow of the FIFO unit will not occur. As with the multiple FIFO buffer embodiments, the FIFO buffer arrangements of the described embodiment make use of a memory runout area; such a memory runout area assures that a FIFO buffer may be used to cache commands from a plurality of application programs. The use of a memory runout area also allows the system to make use of lying in order to enhance the speed of transfer by reducing the number of read operations by the central processing unit of the free count register 45.” Rosenthal ’930 at col. 22, ll. 27–52. “a first-in first-out (FIFO) circuit positioned between a computer input-output bus and input/output devices in the computer and having a plurality of stages, each stage providing storage for commands sent directly from application programs including both data and a virtual address for the data which together translate to provide a physical address for the command, and” Rosenthal ’930 at col. 24, ll. 5–11. “9. A method for controlling the flow of data in a data transfer path of a computer having a first-in first-out (FIFO) buffer with a plurality of stages positioned between a computer input-output bus and input/output devices, each stage providing storage for both data and a virtual address which are translated to provide a physical address for commands decoupled from application programs originating the commands, comprising the steps of: detecting a value indicating an amount of space available for data in the FIFO buffer, transferring to the FIFO buffer an amount of data up to an amount determine from the value detected, and detecting another value indicating an amount of space available for data in the FIFO buffer before transferring additional data to the FIFO buffer.” Rosenthal ’930 at col. 24, ll. 51–65.
3. The system of claim 1, wherein said system is part of a storage area network (SAN) controller.	
4. The system of claim 1, wherein said system is part of a network attached storage (NAS) controller.	

Exhibit 596-16

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.	Rosenthal '930 teaches “[t]he system of claim 1, wherein said stored value is decremented by a predefined value prior to returning said stored value to said at least one second processing element,” including at least the following exemplary citations. For example, Rosenthal '930 teaches wherein said stored value is decremented by a predefined value prior to returning said stored value to said at least one second processing element by disclosing, as non-limiting examples, the application program deducting the number of writes it has performed before needing to read the flow control register again. The application program decrements the stored value by a predefined value. <i>See, e.g.:</i> “Various operating conditions can cause the flow control register 45 associated with a buffer 39 to store different values. If the central processing unit is writing at a rate faster than the input/output device can handle the commands, then a FIFO buffer 39 which is initially empty and provides a maximum free count value when read by the central processing unit will provide a lower free count number after the amount of data first indicated by the free count register has been sent. This will occur because the FIFO buffer 39 will empty more slowly than it is filled. In such a case, the next value read by the application program after sending the amount of data designated by the free count register will be a smaller number. The application program may again send this amount of data and be sure that the FIFO buffer will not overflow. On the other hand, if the central processing unit is writing at a rate slower than the input/output device can handle the commands, then a FIFO buffer which is initially empty and provides a maximum free count value when read by the central processing unit, will provide the same free count number after the amount of data first indicated by the flow control register has been sent. This will occur because the FIFO will empty more rapidly than it is filled and will always be empty when the register 45 is read by the central processing unit.” Rosenthal '930 at col. 17, ll. 19–42. “In utilizing the free count value, the application program should deduct the value of any data which it has already written before each new write operation commences. In this manner, the application program knows at any time the amount of data which it may still write to the input/output device before another read of the free count register will be required. This allows a number of write operations to occur before a read of the register 45 (which significantly slows operation of the overall system) need occur. This also assures that any writes which might be presently in the write channel on the input/output bus are accounted for in determining the actual value of the free count left. It should be noted that this read of the free count register is the only read necessary in dealing with the input/output control unit 29. It would be possible either to interrupt the central processing unit if a FIFO buffer 39 filled or to use the DMA unit 35 to transfer the value in the free

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	count register to the central processing unit instead of reading the value, but in the particular embodiment reading the free count register appears to be the fastest operation.” Rosenthal ’930 at col. 17, l. 52 – col. 18, l. 4.
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.	Rosenthal ’930 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, Rosenthal ’930 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, a DMA device. The DMA device executes the command and notifies the output queue to issue a response. <i>See, e.g.:</i> “To help maintain this asynchronous operating arrangement and to eliminate read operations to the extent possible, the input/output control unit also includes an advanced direct memory access (DMA) device 35 which provides direct memory access for operations conducted involving input/output devices. The DMA device 35 allows the results of input/output operations to be written from input/output devices to main memory 23 rather than requiring read operations by the central processing unit 21 to obtain these results. This eliminates almost all need for the central processing unit 21 to read input/output devices and drastically increases the overall speed of input/output operations. The DMA device 35 includes its own memory management unit which allows writes from input/output devices to the virtual memory space of an application without involving the operating system in the translation process.” Rosenthal ’930 at col. 4, ll. 12–27. “When the operation involves a device directly associated with the control unit 29 on its device bus 34, the commands after the first commands (creating the new data structure, attaching its new virtual name, providing any necessary device context, and creating the address translation) are sent by hardware directly to that device for execution. If the command requires that data be transferred to or from the application, the input/output device performs the transfer using the DMA unit 35. Upon the return of data in response to a command, the DMA unit 35 of the control unit 29 responds by transferring the data to main memory and notifying the central processing unit in a separate DMA operation of the existence of the data so that no local bus read operation by the central processing unit 21 is necessary to ascertain the result of the operation or to receive the data provided.” Rosenthal ’930 at col. 7, ll. 30–45.
[7pre] A system for dividing and synchronizing a processing task	Rosenthal ’930 teaches “[a] system for dividing and synchronizing a processing task across a plurality of processing elements,” including at least the following exemplary citations.

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
across a plurality of processing elements comprising:	<i>See above at row [1pre].</i>
[7a] an input queue implemented in hardware;	Rosenthal '930 teaches “an input queue implemented in hardware,” including at least the following exemplary citations. <i>See above at row [1a].</i>
[7b] an output queue implemented in hardware;	Rosenthal '930 teaches “an output queue implemented in hardware,” including at least the following exemplary citations. <i>See above at row [1b].</i>
[7c] a first processing element having access to said input queue and said output queue;	Rosenthal '930 teaches “a first processing element having access to said input queue and said output queue,” including at least the following exemplary citations. <i>See above at row [1c].</i>
[7d] at least one second processing element in communication with said input queue;	Rosenthal '930 teaches “at least one second processing element in communication with said input queue,” including at least the following exemplary citations. For example, Rosenthal '930 teaches at least one second processing element in communication with said input queue, by disclosing, as non-limiting examples, an I/O control unit. The I/O control unit, the second processing element is in communication with the input queue. <i>See, e.g.:</i>

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	Figure 1 Rosenthal '930 at Figure 1.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	<pre>graph TD; AP[Application Program] --> S[System]; S --> ID1[I/O Driver #1]; ID1 --> IOCU[Input/Output Control Unit 29]; AP --> IOCU; S --> ID2[I/O Driver #2]; ID2 --> RM[I/O Driver #2 Resource Manager]; RM --> IOCU;</pre> Figure 2 Rosenthal '930 at Figure 2.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	The diagram illustrates a system architecture. At the top, a horizontal line represents Bus 27. Below it, a large rectangular block represents the Input/Output Control Unit 29. Inside this unit, at the top, is a 'Decode' block 30. Below 'Decode' 30 is a 'Pusher' block 53. To the left of 'Pusher' 53 is a block containing 'MMU 50' and 'DMA 35'. Below 'MMU 50' is a block labeled '52'. To the right of 'Pusher' 53 is a large grid-like structure with rows and columns, containing various numbered blocks: 42, 7, 3, 11, 32, 39, 45, 31, and 40. Below the grid is a 'Resource Manager' block (dashed outline) and a 'Translation Table' block 38. Below the 'Input/Output Control Unit 29' is another horizontal line representing Bus 34. Below Bus 34 are three blocks: 'Disk Controller' 32, 'Graphics Output Controller Memory' 33, and 'Sound Generator' 37. Arrows indicate data flow between these components and the buses. Figure 3 Rosenthal '930 at Figure 3.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	Figure 6 Rosenthal '930 at Figure 6.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	Figure 7 Rosenthal '930 at Figure 7. “It is another object of the present invention to provide a flow control system for an architecture which utilizes FIFO buffers for buffering write operations. These and other objects of the present invention are realized in a system which uses an arrangement of parallel FIFO buffers in which each FIFO buffer handles data from only one application program at any time. In order to assure that no data written to a FIFO buffer by an application program will overflow the FIFO buffer, each FIFO buffer includes a flow control register which must be read by the central processing unit running the application before writing data to an input/output device. The register stores a value which indicates the amount of space available in the FIFO buffer to which data may be written. Reading this register tells the application program how much data may be written without running the risk of overflowing the data storage area which the input/output device has available. Provided that no application program ever writes more than the data for which it has permission, no data will be lost.” Rosenthal '930 at col. 2, ll. 15–33. “The architecture of the system 22 has been designed so that it eliminates almost all read operations of input/output devices by the central processing unit. In order to accomplish this, the input/output control unit 29 includes a first-in first-out (FIFO) unit 31 for storing write operations directed to the input/output control

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	unit. The FIFO unit 31 queues incoming write operations. Unlike FIFO units in prior art systems, it stores both addresses and data. This allows the write operations to the input/output control unit 29 to occur asynchronously so that both the central processing unit 21 and the input/output control unit 29 may be functioning independently of one another and neither need wait for operations of the other.” Rosenthal ’930 at col. 3, l. 66 – col. 4, l. 11. “In one embodiment, the FIFO unit 31 includes a plurality of FIFO buffers 39 (see FIG. 3). When an application program desires to write to an input/output device on the device bus 34, it addresses that device. Decoding circuitry decodes the address by reviewing a number of the highest order bits decoded by the chip sufficient to indicate a unique portion of the input/output address space assigned to an application program and places the command in the appropriate FIFO buffer 39 for that application program. Each FIFO buffer 39 handles commands only from the application program to which the address area has been mapped. An arrangement using a smaller number of FIFO buffers 39 than one buffer for each application program (as few as a single buffer 39) could also be utilized as long as the address of the FIFO buffer is mapped to only one application program at a time.” Rosenthal ’930 at col. 5, ll. 51–66. “The unit 29 receives physical addresses furnished by the memory management unit and virtual names furnished by application programs for operations to be performed which have been sent to the FIFO unit 31 and controls the translation of those virtual names for all input/output devices. The hardware unit 29 includes the device bus 34 to which the individual input/output devices such as a disk controller 32, a graphics output controller 33, and a sound generator 37 are shown joined. The unit 29 also includes a DMA unit 35 which is adapted to transfer data between the individual input/output devices and main memory for use by the central processing unit or other components of the system.” Rosenthal ’930 at col. 5, l. 67 – col. 6, l. 10. “FIG. 3 is a block diagram illustrating details of the input/output control unit 29 including the device bus 34 and the input/output devices arranged on that bus. As described above, the input/output control unit 29 includes a decode unit 30 which receives commands directly from the input/output bus 27 and transfers the commands to a pusher circuit 53 which transfers the command to one of the FIFO buffers 39 of the FIFO unit 31. The FIFO unit 31 stores the data along with the addresses for each of the commands being transferred to all of the input/output devices associated with the input/output control unit 29. The buffers 39 of the FIFO unit replace the individual data registers used by the input/output devices of the prior art. However, in contrast to the registers used by the prior art for communication on the input/output bus 27, the FIFO unit 31 allows a very much larger number of commands to be processed much more rapidly and facilitates the asynchronous operations of the input/output devices and the central processing unit. In one embodiment, the FIFO unit 31

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	includes 128 individual FIFO buffers 39 each of which has thirty-two stages. The use of 128 individual FIFO buffers 39 allows a FIFO buffer to be assigned to each of 128 different application programs which may be running on the central processing unit for the use of that application program alone. The thirty-two stages of each buffer 39 allow any FIFO buffer to hold thirty-two individual serially ordered commands at any time. Although each of the stages of the FIFO unit 31 holds (along with the address bits) the data held by an individual register of a typical prior art input/output device, a particular system has the ability to store commands for over 16 thousand 32 bit registers for each of 128 different application programs which may map different addresses decoded by the input/output control unit 29.” Rosenthal ’930 at col. 8, ll. 21–53. “The use of the flow control register 45 allows an application program presently using the resources of the input/output control unit 29 to proceed without causing the overflow of the FIFO unit 31. Whenever the application has written commands totaling the amount previously read from the register 45, it must read the register again. This is especially useful for application programs which may wish to transfer a series of commands in sequence, possibly using the burst transfer mode of operation provided by many of the modern buses. By ascertaining the space available in the FIFO buffer 39 before transferring a sequence of commands, the application can know that sufficient space will be available for the operation. Provided that no application program ever writes more than the data for which it has permission, no data will be lost.” Rosenthal ’930 at col. 17, ll. 4–18. “Various operating conditions can cause the flow control register 45 associated with a buffer 39 to store different values. If the central processing unit is writing at a rate faster than the input/output device can handle the commands, then a FIFO buffer 39 which is initially empty and provides a maximum free count value when read by the central processing unit will provide a lower free count number after the amount of data first indicated by the free count register has been sent. This will occur because the FIFO buffer 39 will empty more slowly than it is filled. In such a case, the next value read by the application program after sending the amount of data designated by the free count register will be a smaller number. The application program may again send this amount of data and be sure that the FIFO buffer will not overflow. On the other hand, if the central processing unit is writing at a rate slower than the input/output device can handle the commands, then a FIFO buffer which is initially empty and provides a maximum free count value when read by the central processing unit, will provide the same free count number after the amount of data first indicated by the flow control register has been sent. This will occur because the FIFO will empty more rapidly than it is filled and will always be empty when the register 45 is read by the central processing unit.” Rosenthal ’930 at col. 17, ll. 19–42.

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	“Those skilled in the art will recognize that the use of 128 individual FIFO buffers 39 to receive commands written by the central processing unit is very expensive and probably not economically feasible for use with personal computers. In addition, it is expensive of system resources for an operating system to context switch the central processing unit from one application program to another. Thus, the operating system will try to ensure that once an application is given the central processing unit the application will retain control long enough to amortize the cost of the context switch. This means that typically in a one processor system at most one FIFO buffer will be active at a time. In a multi-processor system, typically at most as many FIFO buffers will be active as there are processors.” Rosenthal ’930 at col. 21, ll. 39–52. “One embodiment of the new architecture provides a means for attaining almost the same speed as is possible with 128 FIFO buffers but much more economically. Instead of 128 individual FIFO buffers 39, the described embodiment utilizes a smaller number of FIFO buffers 39 than the number of areas into which the input/output address space may be divided and thus the number of application programs which may be run. Utilizing a smaller number of FIFO buffers causes each FIFO buffer to function as a cache for commands which may be sent by any of the application programs. In the preferred embodiment, one FIFO buffer 39 is used for each processor which may be writing to the input/output control unit 29. For example, if the computer utilizes two processors arranged as symmetrical multi-processors, then two individual FIFO buffers are utilized. If, as is usually the case, only a single central processing unit is used, then a single FIFO buffer 39 may in the limit act as a cache for commands from all application programs which may be running on the central processing unit. Such an arrangement is illustrated in FIG. 6. Of course, in either case more FIFO buffers might be used.” Rosenthal ’930 at col. 21, l. 53 – col. 22, l. 6. “9. A method for controlling the flow of data in a data transfer path of a computer having a first-in first-out (FIFO) buffer with a plurality of stages positioned between a computer input-output bus and input/output devices, each stage providing storage for both data and a virtual address which are translated to provide a physical address for commands decoupled from application programs originating the commands, comprising the steps of: detecting a value indicating an amount of space available for data in the FIFO buffer, transferring to the FIFO buffer an amount of data up to an amount determine from the value detected, and detecting another value indicating an amount of space available for data in the FIFO buffer before transferring additional data to the FIFO buffer.” Rosenthal ’930 at col. 24, ll. 51–65.

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
[7e] a reservation register implemented in hardware storing a value indicative of available space in said input queue, said reservation register accessible by both said first processing element and said at least one second processing element;	Rosenthal '930 teaches “a reservation register implemented in hardware storing a value indicative of available space in said input queue, said reservation register accessible by both said first processing element and said at least one second processing element,” including at least the following exemplary citations. <i>See above at row [1e].</i>
[7f] computer storage storing instructions, which when executed by said at least one second processing element:	Rosenthal '930 teaches computer storage storing instructions executed by said at least one second processing element, including at least the following exemplary citations. <i>See above at row [1f].</i>
[7g] receives a command from said first processing element;	Rosenthal '930 teaches “receives a command from said first processing element,” including at least the following exemplary citations. For example, Rosenthal '930 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, the CPU sending commands to the I/O unit. The I/O unit, the second processing element receives a command from the first processing element. <i>See, e.g.:</i> “A digital system which uses an arrangement of one or more parallel FIFO buffers in which each FIFO buffer handles data from only one application program at any time. In order to assure that no data written to a FIFO buffer by an application program will overflow the FIFO buffer, each FIFO buffer includes a flow control register which must be read by the processing unit running the application before writing data to an input/output device. The register stores a value which indicates the amount of space available in the FIFO buffer to which data may be written. Reading this register tells the application program how much data may be written without running the risk of overflowing the data storage area which the input/output device has available.” Rosenthal '930 at Abstract. “It is another object of the present invention to provide a flow control system for an architecture which utilizes FIFO buffers for buffering write operations.

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	These and other objects of the present invention are realized in a system which uses an arrangement of parallel FIFO buffers in which each FIFO buffer handles data from only one application program at any time. In order to assure that no data written to a FIFO buffer by an application program will overflow the FIFO buffer, each FIFO buffer includes a flow control register which must be read by the central processing unit running the application before writing data to an input/output device. The register stores a value which indicates the amount of space available in the FIFO buffer to which data may be written. Reading this register tells the application program how much data may be written without running the risk of overflowing the data storage area which the input/output device has available. Provided that no application program ever writes more than the data for which it has permission, no data will be lost.” Rosenthal ’930 at col. 2, ll. 15–33. “The use of the flow control register 45 allows an application program presently using the resources of the input/output control unit 29 to proceed without causing the overflow of the FIFO unit 31. Whenever the application has written commands totaling the amount previously read from the register 45, it must read the register again. This is especially useful for application programs which may wish to transfer a series of commands in sequence, possibly using the burst transfer mode of operation provided by many of the modern buses. By ascertaining the space available in the FIFO buffer 39 before transferring a sequence of commands, the application can know that sufficient space will be available for the operation. Provided that no application program ever writes more than the data for which it has permission, no data will be lost.” Rosenthal ’930 at col. 17, ll. 4–18. “In one embodiment, the application program may obtain a free count value indicating the number of free entries in a FIFO buffer 39 by reading from a designated method offset in any of the sub-areas of the area mapped to the application program. The value read is that stored in the flow control register 45 associated with the particular FIFO buffer 39. This value is then placed on the bus when the central processing unit provides the appropriate control signals to implement the read operation. The application may write up to the amount of data designated by this free count value without further testing and be certain that overflow of the FIFO buffer 39 will not occur.” Rosenthal ’930 at col. 18, ll. 16–27. “As with the embodiments which use one FIFO buffer for each application area, a free count register 45 and a puller are included with each FIFO buffer 39 as a part of the input/output control unit 29. The register 45 stores an indication of the number of available entries in each FIFO buffer 39 of the FIFO unit 31 and allows a requirement to be placed on an application program attempting to access the input/output control unit 29 that it first determine whether space is available in the FIFO unit 31. It also allows an application presently using the resources of the input/output control unit 29 to proceed without overflowing the FIFO unit 31. The

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	application program may obtain a boundary value indicating the number of free entries in the FIFO buffer by reading from a designated offset in any of the sub-areas of its mapped area. The application may write up to the amount of data designated by this boundary value without further testing and be certain that overflow of the FIFO unit will not occur. As with the multiple FIFO buffer embodiments, the FIFO buffer arrangements of the described embodiment make use of a memory runout area; such a memory runout area assures that a FIFO buffer may be used to cache commands from a plurality of application programs. The use of a memory runout area also allows the system to make use of lying in order to enhance the speed of transfer by reducing the number of read operations by the central processing unit of the free count register 45.” Rosenthal ’930 at col. 22, ll. 27–52. “9. A method for controlling the flow of data in a data transfer path of a computer having a first-in first-out (FIFO) buffer with a plurality of stages positioned between a computer input-output bus and input/output devices, each stage providing storage for both data and a virtual address which are translated to provide a physical address for commands decoupled from application programs originating the commands, comprising the steps of: detecting a value indicating an amount of space available for data in the FIFO buffer, transferring to the FIFO buffer an amount of data up to an amount determine from the value detected, and detecting another value indicating an amount of space available for data in the FIFO buffer before transferring additional data to the FIFO buffer.” Rosenthal ’930 at col. 24, ll. 51–65.
[7h] accesses said reservation register and reads said stored value;	Rosenthal ’930 teaches “accesses said reservation register and reads said stored value,” including at least the following exemplary citations. <i>See above at row [1g].</i>
[7i] determines when said read value indicates available space in said input queue to issue said command;	Rosenthal ’930 teaches “determines when said read value indicates available space in said input queue to issue said command,” including at least the following exemplary citations. <i>See above at row [1h].</i>

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
[7j] issues said command to said input queue on behalf of said first processing element; and	Rosenthal '930 teaches "issues said command to said input queue on behalf of said first processing element," including at least the following exemplary citations. For example, Rosenthal '930 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 I/O control unit that issues the command. The I/O control unit issues the command to the input queue on behalf of the first processing element. <i>See, e.g.:</i> "A digital system which uses an arrangement of one or more parallel FIFO buffers in which each FIFO buffer handles data from only one application program at any time. In order to assure that no data written to a FIFO buffer by an application program will overflow the FIFO buffer, each FIFO buffer includes a flow control register which must be read by the processing unit running the application before writing data to an input/output device. The register stores a value which indicates the amount of space available in the FIFO buffer to which data may be written. Reading this register tells the application program how much data may be written without running the risk of overflowing the data storage area which the input/output device has available." Rosenthal '930 at Abstract. "It is another object of the present invention to provide a flow control system for an architecture which utilizes FIFO buffers for buffering write operations. These and other objects of the present invention are realized in a system which uses an arrangement of parallel FIFO buffers in which each FIFO buffer handles data from only one application program at any time. In order to assure that no data written to a FIFO buffer by an application program will overflow the FIFO buffer, each FIFO buffer includes a flow control register which must be read by the central processing unit running the application before writing data to an input/output device. The register stores a value which indicates the amount of space available in the FIFO buffer to which data may be written. Reading this register tells the application program how much data may be written without running the risk of overflowing the data storage area which the input/output device has available. Provided that no application program ever writes more than the data for which it has permission, no data will be lost." Rosenthal '930 at col. 2, ll. 15–33. "In one embodiment of the new architecture, a flow control register 45 is included for each FIFO buffer 39 of the input/output control unit 29. The register 45 stores an indication of the number of available spaces for commands in the associated FIFO buffer 39. The use of this flow control register 45 allows a requirement to be placed on an application program attempting an access of the input/output control unit 29 that it first determine whether space is available in the FIFO unit 31 before it writes any command. Before writing any data, the application program must determine the amount of "free" space to which it may write. This it does

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	by having the central processing unit read the value in the flow control register 45 associated with the FIFO buffer 39 being utilized. Once this value has been read, the application program may send up to that amount of data before it need test again to determine whether free space remains to which it may send more data.” Rosenthal ’930 at col. 16, ll. 33–49. “The use of the flow control register 45 allows an application program presently using the resources of the input/output control unit 29 to proceed without causing the overflow of the FIFO unit 31. Whenever the application has written commands totaling the amount previously read from the register 45, it must read the register again. This is especially useful for application programs which may wish to transfer a series of commands in sequence, possibly using the burst transfer mode of operation provided by many of the modern buses. By ascertaining the space available in the FIFO buffer 39 before transferring a sequence of commands, the application can know that sufficient space will be available for the operation. Provided that no application program ever writes more than the data for which it has permission, no data will be lost.” Rosenthal ’930 at col. 17, ll. 4–18. “Various operating conditions can cause the flow control register 45 associated with a buffer 39 to store different values. If the central processing unit is writing at a rate faster than the input/output device can handle the commands, then a FIFO buffer 39 which is initially empty and provides a maximum free count value when read by the central processing unit will provide a lower free count number after the amount of data first indicated by the free count register has been sent. This will occur because the FIFO buffer 39 will empty more slowly than it is filled. In such a case, the next value read by the application program after sending the amount of data designated by the free count register will be a smaller number. The application program may again send this amount of data and be sure that the FIFO buffer will not overflow. On the other hand, if the central processing unit is writing at a rate slower than the input/output device can handle the commands, then a FIFO buffer which is initially empty and provides a maximum free count value when read by the central processing unit, will provide the same free count number after the amount of data first indicated by the flow control register has been sent. This will occur because the FIFO will empty more rapidly than it is filled and will always be empty when the register 45 is read by the central processing unit.” Rosenthal ’930 at col. 17, ll. 19–42. “In one embodiment, the application program may obtain a free count value indicating the number of free entries in a FIFO buffer 39 by reading from a designated method offset in any of the sub-areas of the area mapped to the application program. The value read is that stored in the flow control register 45 associated with the particular FIFO buffer 39. This value is then placed on the bus when the central processing unit

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	provides the appropriate control signals to implement the read operation. The application may write up to the amount of data designated by this free count value without further testing and be certain that overflow of the FIFO buffer 39 will not occur.” Rosenthal ’930 at col. 18, ll. 16–27. “As with the embodiments which use one FIFO buffer for each application area, a free count register 45 and a puller are included with each FIFO buffer 39 as a part of the input/output control unit 29. The register 45 stores an indication of the number of available entries in each FIFO buffer 39 of the FIFO unit 31 and allows a requirement to be placed on an application program attempting to access the input/output control unit 29 that it first determine whether space is available in the FIFO unit 31. It also allows an application presently using the resources of the input/output control unit 29 to proceed without overflowing the FIFO unit 31. The application program may obtain a boundary value indicating the number of free entries in the FIFO buffer by reading from a designated offset in any of the sub-areas of its mapped area. The application may write up to the amount of data designated by this boundary value without further testing and be certain that overflow of the FIFO unit will not occur. As with the multiple FIFO buffer embodiments, the FIFO buffer arrangements of the described embodiment make use of a memory runout area; such a memory runout area assures that a FIFO buffer may be used to cache commands from a plurality of application programs. The use of a memory runout area also allows the system to make use of lying in order to enhance the speed of transfer by reducing the number of read operations by the central processing unit of the free count register 45.” Rosenthal ’930 at col. 22, ll. 27–52. “9. A method for controlling the flow of data in a data transfer path of a computer having a first-in first-out (FIFO) buffer with a plurality of stages positioned between a computer input-output bus and input/output devices, each stage providing storage for both data and a virtual address which are translated to provide a physical address for commands decoupled from application programs originating the commands, comprising the steps of: detecting a value indicating an amount of space available for data in the FIFO buffer, transferring to the FIFO buffer an amount of data up to an amount determine from the value detected, and detecting another value indicating an amount of space available for data in the FIFO buffer before transferring additional data to the FIFO buffer.” Rosenthal ’930 at col. 24, ll. 51–65.
[7k] wherein said first processing element receives a response	Rosenthal ’930 teaches “wherein said first processing element receives a response corresponding to said command from said output queue,” including at least the following exemplary citations.

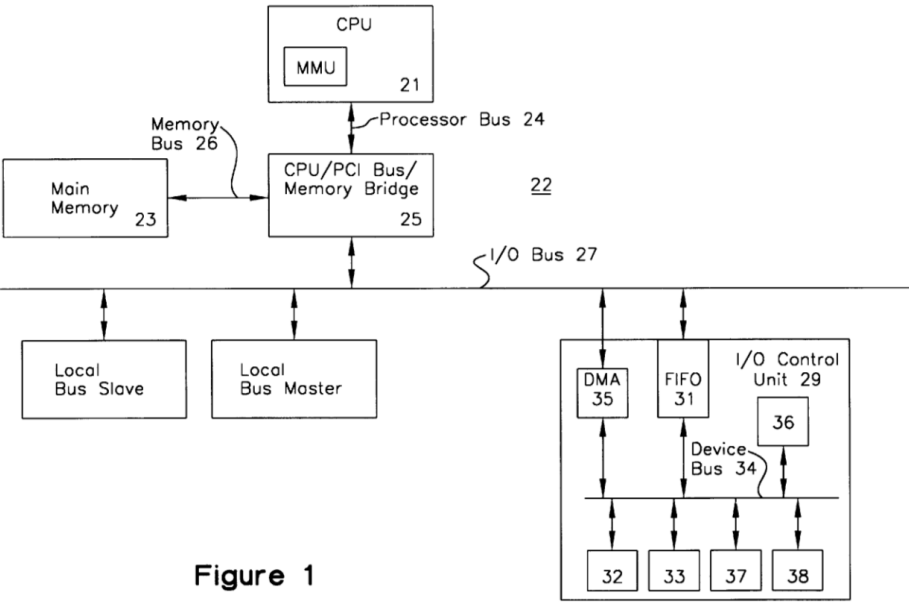
Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
corresponding to said command from said output queue.	For example, Rosenthal '930 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 from the DMA unit. The CPU receives a response from the output queue corresponding to the command. <i>See, e.g.:</i> “When the operation involves a device directly associated with the control unit 29 on its device bus 34, the commands after the first commands (creating the new data structure, attaching its new virtual name, providing any necessary device context, and creating the address translation) are sent by hardware directly to that device for execution. If the command requires that data be transferred to or from the application, the input/output device performs the transfer using the DMA unit 35. Upon the return of data in response to a command, the DMA unit 35 of the control unit 29 responds by transferring the data to main memory and notifying the central processing unit in a separate DMA operation of the existence of the data so that no local bus read operation by the central processing unit 21 is necessary to ascertain the result of the operation or to receive the data provided.” Rosenthal '930 at col. 7, ll. 30–45.
8. The system of claim 7, wherein said input queue is a first-in-first-out (FIFO) buffer.	Rosenthal '930 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. <i>See above at row 2.</i>
9. The system of claim 7, wherein said system is part of a storage area network (SAN) controller.	
10. The system of claim 7, wherein said system is part of a network attached storage (NAS) controller.	
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.	Rosenthal '930 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. <i>See above at row 5.</i>

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
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.	Rosenthal '930 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. <i>See above at row 6.</i>
[13pre] A system for dividing and synchronizing a processing task across a plurality of processing elements comprising:	Rosenthal '930 teaches "[a] system for dividing and synchronizing a processing task across a plurality of processing elements," including at least the following exemplary citations. <i>See above at row [1pre].</i>
[13a] an input queue implemented in hardware;	Rosenthal '930 teaches "an input queue implemented in hardware," including at least the following exemplary citations. <i>See above at row [1a].</i>
[13b] an output queue implemented in hardware;	Rosenthal '930 teaches "an output queue implemented in hardware," including at least the following exemplary citations. <i>See above at row [1b].</i>
[13c] a first processing element having access to said input queue and said output queue;	Rosenthal '930 teaches "a first processing element having access to said input queue and said output queue," including at least the following exemplary citations. <i>See above at row [1c].</i>
[13d] at least one second processing element in communication with said input queue and said output queue;	Rosenthal '930 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, Rosenthal '930 teaches at least one second processing element in communication with said input queue and said output queue, by disclosing, as non-limiting examples, an I/O control unit. The I/O control unit, the second processing element is in communication with the input and output queue. <i>See, e.g.:</i>

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Figure 1 Rosenthal '930 at Figure 1.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	<pre>graph TD; AP[Application Program] --> S[System]; S --> ID1[I/O Driver #1]; ID1 --> IOCU[Input/Output Control Unit 29]; AP --> IOCU; S --> ID2[I/O Driver #2]; ID2 --> RM[I/O Driver #2 Resource Manager]; RM --> IOCU;</pre> The diagram illustrates a system architecture. At the top is a box labeled 'Application Program'. An arrow points down from it to a box labeled 'System'. From 'System', an arrow points down to a box labeled 'I/O Driver #1'. From 'I/O Driver #1', an arrow points down to a wide box at the bottom labeled 'Input/Output Control Unit 29'. A second arrow points down from 'Application Program' to the 'Input/Output Control Unit 29'. From 'System', an arrow points down to a box labeled 'I/O Driver #2'. From 'I/O Driver #2', an arrow points down to a box labeled 'I/O Driver #2 Resource Manager'. From 'I/O Driver #2 Resource Manager', an arrow points down to the 'Input/Output Control Unit 29'. Figure 2 Rosenthal '930 at Figure 2.

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	The diagram illustrates a system architecture. At the top, a horizontal line represents Bus 27. Below it, a large rectangular block represents the Input/Output Control Unit 29. Inside this unit, there is a 'Decode' block 30 connected to Bus 27. Below 'Decode' is a 'Pusher' block 53. To the left of the Pusher is a block containing 'MMU 50' and 'DMA 35'. Below the Pusher is a large grid-like structure with various numbered cells (7, 3, 11, 32, 39, 40, 45, etc.). To the right of this grid is a 'Resource Manager' block (dashed outline) and a 'Translation Table' block 38. Below the main unit, a horizontal line represents Bus 34. Connected to Bus 34 are three blocks: 'Disk Controller 32', 'Graphics Output Controller Memory 33', and 'Sound Generator 37'. Arrows indicate data flow between these components and the buses. Figure 3 Rosenthal '930 at Figure 3.

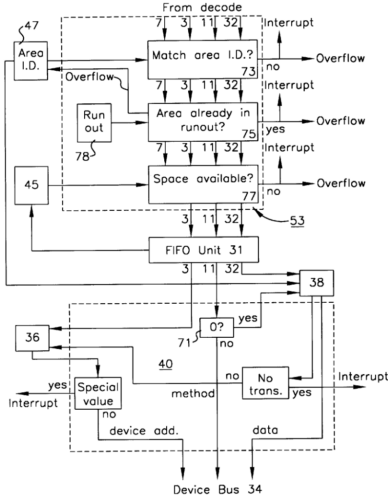
Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	 Figure 7 Rosenthal '930 at Figure 7. “It is another object of the present invention to provide a flow control system for an architecture which utilizes FIFO buffers for buffering write operations. These and other objects of the present invention are realized in a system which uses an arrangement of parallel FIFO buffers in which each FIFO buffer handles data from only one application program at any time. In order to assure that no data written to a FIFO buffer by an application program will overflow the FIFO buffer, each FIFO buffer includes a flow control register which must be read by the central processing unit running the application before writing data to an input/output device. The register stores a value which indicates the amount of space available in the FIFO buffer to which data may be written. Reading this register tells the application program how much data may be written without running the risk of overflowing the data storage area which the input/output device has available. Provided that no application program ever writes more than the data for which it has permission, no data will be lost.” Rosenthal '930 at col. 2, ll. 15–33. “The architecture of the system 22 has been designed so that it eliminates almost all read operations of input/output devices by the central processing unit. In order to accomplish this, the input/output control unit 29 includes a first-in first-out (FIFO) unit 31 for storing write operations directed to the input/output control

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	unit. The FIFO unit 31 queues incoming write operations. Unlike FIFO units in prior art systems, it stores both addresses and data. This allows the write operations to the input/output control unit 29 to occur asynchronously so that both the central processing unit 21 and the input/output control unit 29 may be functioning independently of one another and neither need wait for operations of the other.” Rosenthal ’930 at col. 3, l. 66 – col. 4, l. 11. “In one embodiment, the FIFO unit 31 includes a plurality of FIFO buffers 39 (see FIG. 3). When an application program desires to write to an input/output device on the device bus 34, it addresses that device. Decoding circuitry decodes the address by reviewing a number of the highest order bits decoded by the chip sufficient to indicate a unique portion of the input/output address space assigned to an application program and places the command in the appropriate FIFO buffer 39 for that application program. Each FIFO buffer 39 handles commands only from the application program to which the address area has been mapped. An arrangement using a smaller number of FIFO buffers 39 than one buffer for each application program (as few as a single buffer 39) could also be utilized as long as the address of the FIFO buffer is mapped to only one application program at a time.” Rosenthal ’930 at col. 5, ll. 51–66. “The unit 29 receives physical addresses furnished by the memory management unit and virtual names furnished by application programs for operations to be performed which have been sent to the FIFO unit 31 and controls the translation of those virtual names for all input/output devices. The hardware unit 29 includes the device bus 34 to which the individual input/output devices such as a disk controller 32, a graphics output controller 33, and a sound generator 37 are shown joined. The unit 29 also includes a DMA unit 35 which is adapted to transfer data between the individual input/output devices and main memory for use by the central processing unit or other components of the system.” Rosenthal ’930 at col. 5, l. 67 – col. 6, l. 10. “FIG. 3 is a block diagram illustrating details of the input/output control unit 29 including the device bus 34 and the input/output devices arranged on that bus. As described above, the input/output control unit 29 includes a decode unit 30 which receives commands directly from the input/output bus 27 and transfers the commands to a pusher circuit 53 which transfers the command to one of the FIFO buffers 39 of the FIFO unit 31. The FIFO unit 31 stores the data along with the addresses for each of the commands being transferred to all of the input/output devices associated with the input/output control unit 29. The buffers 39 of the FIFO unit replace the individual data registers used by the input/output devices of the prior art. However, in contrast to the registers used by the prior art for communication on the input/output bus 27, the FIFO unit 31 allows a very much larger number of commands to be processed much more rapidly and facilitates the asynchronous operations of the input/output devices and the central processing unit. In one embodiment, the FIFO unit 31

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	includes 128 individual FIFO buffers 39 each of which has thirty-two stages. The use of 128 individual FIFO buffers 39 allows a FIFO buffer to be assigned to each of 128 different application programs which may be running on the central processing unit for the use of that application program alone. The thirty-two stages of each buffer 39 allow any FIFO buffer to hold thirty-two individual serially ordered commands at any time. Although each of the stages of the FIFO unit 31 holds (along with the address bits) the data held by an individual register of a typical prior art input/output device, a particular system has the ability to store commands for over 16 thousand 32 bit registers for each of 128 different application programs which may map different addresses decoded by the input/output control unit 29.” Rosenthal ’930 at col. 8, ll. 21–53. “The use of the flow control register 45 allows an application program presently using the resources of the input/output control unit 29 to proceed without causing the overflow of the FIFO unit 31. Whenever the application has written commands totaling the amount previously read from the register 45, it must read the register again. This is especially useful for application programs which may wish to transfer a series of commands in sequence, possibly using the burst transfer mode of operation provided by many of the modern buses. By ascertaining the space available in the FIFO buffer 39 before transferring a sequence of commands, the application can know that sufficient space will be available for the operation. Provided that no application program ever writes more than the data for which it has permission, no data will be lost.” Rosenthal ’930 at col. 17, ll. 4–18. “Various operating conditions can cause the flow control register 45 associated with a buffer 39 to store different values. If the central processing unit is writing at a rate faster than the input/output device can handle the commands, then a FIFO buffer 39 which is initially empty and provides a maximum free count value when read by the central processing unit will provide a lower free count number after the amount of data first indicated by the free count register has been sent. This will occur because the FIFO buffer 39 will empty more slowly than it is filled. In such a case, the next value read by the application program after sending the amount of data designated by the free count register will be a smaller number. The application program may again send this amount of data and be sure that the FIFO buffer will not overflow. On the other hand, if the central processing unit is writing at a rate slower than the input/output device can handle the commands, then a FIFO buffer which is initially empty and provides a maximum free count value when read by the central processing unit, will provide the same free count number after the amount of data first indicated by the flow control register has been sent. This will occur because the FIFO will empty more rapidly than it is filled and will always be empty when the register 45 is read by the central processing unit.” Rosenthal ’930 at col. 17, ll. 19–42.

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
	“Those skilled in the art will recognize that the use of 128 individual FIFO buffers 39 to receive commands written by the central processing unit is very expensive and probably not economically feasible for use with personal computers. In addition, it is expensive of system resources for an operating system to context switch the central processing unit from one application program to another. Thus, the operating system will try to ensure that once an application is given the central processing unit the application will retain control long enough to amortize the cost of the context switch. This means that typically in a one processor system at most one FIFO buffer will be active at a time. In a multi-processor system, typically at most as many FIFO buffers will be active as there are processors.” Rosenthal ’930 at col. 21, ll. 39–52. “One embodiment of the new architecture provides a means for attaining almost the same speed as is possible with 128 FIFO buffers but much more economically. Instead of 128 individual FIFO buffers 39, the described embodiment utilizes a smaller number of FIFO buffers 39 than the number of areas into which the input/output address space may be divided and thus the number of application programs which may be run. Utilizing a smaller number of FIFO buffers causes each FIFO buffer to function as a cache for commands which may be sent by any of the application programs. In the preferred embodiment, one FIFO buffer 39 is used for each processor which may be writing to the input/output control unit 29. For example, if the computer utilizes two processors arranged as symmetrical multi-processors, then two individual FIFO buffers are utilized. If, as is usually the case, only a single central processing unit is used, then a single FIFO buffer 39 may in the limit act as a cache for commands from all application programs which may be running on the central processing unit. Such an arrangement is illustrated in FIG. 6. Of course, in either case more FIFO buffers might be used.” Rosenthal ’930 at col. 21, l. 53 – col. 22, l. 6. “9. A method for controlling the flow of data in a data transfer path of a computer having a first-in first-out (FIFO) buffer with a plurality of stages positioned between a computer input-output bus and input/output devices, each stage providing storage for both data and a virtual address which are translated to provide a physical address for commands decoupled from application programs originating the commands, comprising the steps of: detecting a value indicating an amount of space available for data in the FIFO buffer, transferring to the FIFO buffer an amount of data up to an amount determine from the value detected, and detecting another value indicating an amount of space available for data in the FIFO buffer before transferring additional data to the FIFO buffer.” Rosenthal ’930 at col. 24, ll. 51–65.

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
[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;	Rosenthal '930 teaches “a reservation register implemented in hardware storing a value indicative of available space in said input queue, said reservation register accessible by both said first processing element and said at least one second processing element,” including at least the following exemplary citations. <i>See above at row [1e].</i>
[13f] computer storage storing instructions, which when executed by said at least one second processing element:	Rosenthal '930 teaches computer storage storing instructions that are executed by said at least one second processing element including at least the following exemplary citations. <i>See above at row [1f].</i>
[13g] receives a command from said first processing element;	Rosenthal '930 teaches “receives a command from said first processing element,” including at least the following exemplary citations. <i>See above at row [7g].</i>
[13h] accesses said reservation register and reads said stored value;	Rosenthal '930 teaches “accesses said reservation register and reads said stored value,” including at least the following exemplary citations. <i>See above at row [1g].</i>
[13i] determines when said read value indicates available space in said input queue to issue said command;	Rosenthal '930 teaches “determines when said read value indicates available space in said input queue to issue said command,” including at least the following exemplary citations. <i>See above at row [1h].</i>
[13j] issues said command to said input queue on behalf of said first processing element; and	Rosenthal '930 teaches “issues said command to said input queue on behalf of said first processing element,” including at least the following exemplary citations. <i>See above at row [7j].</i>

Exhibit 596-16

Claim Language	Exemplary Evidence of Anticipation and/or Obviousness
[13k] receives a response corresponding to said command from said output queue.	Rosenthal '930 teaches “receives a response corresponding to said command from said output queue,” including at least the following exemplary citations. For example, Rosenthal '930 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 from the DMA unit. The I/O control unit receives a corresponding response from the output queue. <i>See, e.g.:</i> “When the operation involves a device directly associated with the control unit 29 on its device bus 34, the commands after the first commands (creating the new data structure, attaching its new virtual name, providing any necessary device context, and creating the address translation) are sent by hardware directly to that device for execution. If the command requires that data be transferred to or from the application, the input/output device performs the transfer using the DMA unit 35. Upon the return of data in response to a command, the DMA unit 35 of the control unit 29 responds by transferring the data to main memory and notifying the central processing unit in a separate DMA operation of the existence of the data so that no local bus read operation by the central processing unit 21 is necessary to ascertain the result of the operation or to receive the data provided.” Rosenthal '930 at col. 7, ll. 30–45.
14. The system of claim 13, wherein said input queue is a first-in-first-out (FIFO) buffer.	Rosenthal '930 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. <i>See</i> above at row 2.
15. The system of claim 13, wherein said system is either part of a storage area network (SAN) controller.	Rosenthal '930 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.
16. The system of claim 13, wherein said system is part of a network attached storage (NAS) controller.	Rosenthal '930 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.

Exhibit 596-16

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.	Rosenthal '930 teaches “[t]he system of claim 13, wherein said stored value is decremented by a predefined value prior to returning said stored value to said at least one second processing element,” including at least the following exemplary citations. <i>See above at row 5.</i>
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.	Rosenthal '930 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. <i>See above at row 6.</i>