US 20020138675A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2002/0138675 A1

Mann (43) Pub. Date: Sep. 26, 2002

(54) PROCESSOR FOR DETERMINING PHYSICAL LANE SKEW ORDER

(75) Inventor: Gregory J. Mann, Colchester, VT (US)

Correspondence Address:
FREDERICK W. GIBB, III
MCGINN & GIBB, PLLC
2568-A RIVA ROAD
SUITE 304
ANNAPOLIS, MD 21401 (US)

(73) Assignee: International Business Machines Corporation, Armonk, NY (US)

(21) Appl. No.: 09/816,969

(22) Filed: Mar. 23, 2001

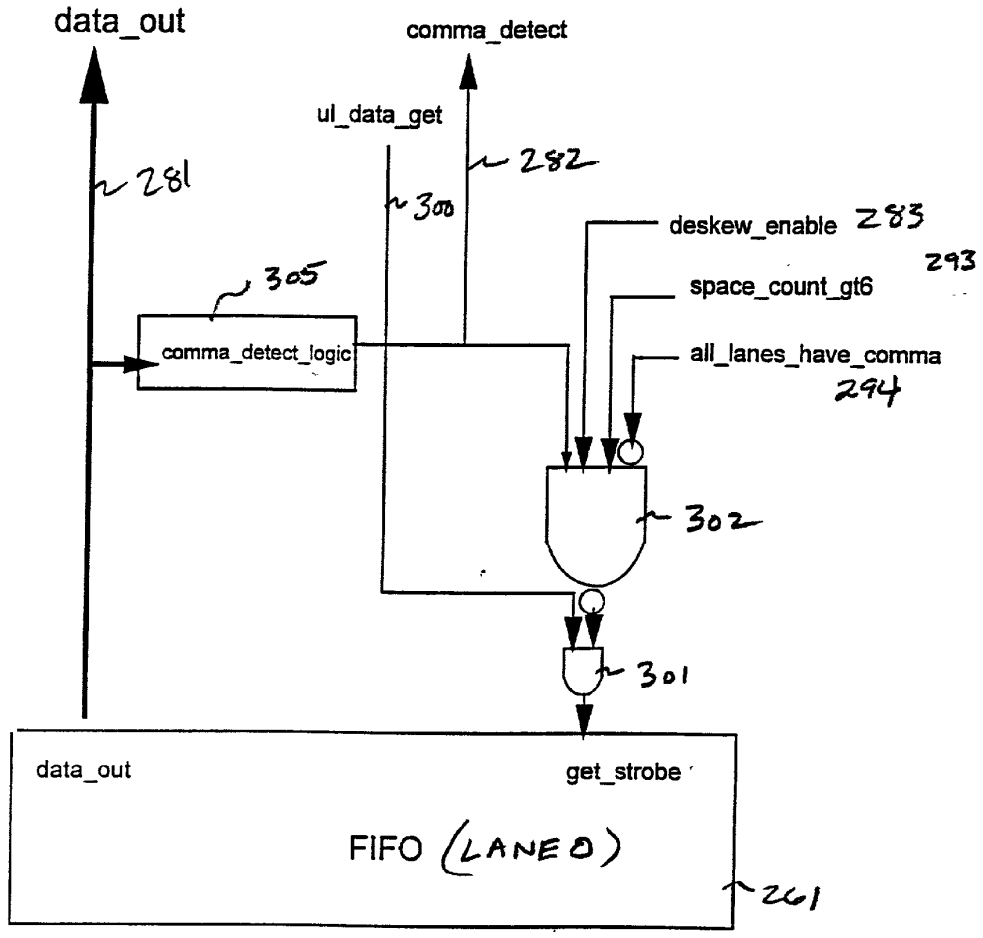
Publication Classification

(51) Int. Cl.[7] G06F 3/00

(52) U.S. Cl. 710/61

(57) ABSTRACT

In parallel-serial architecture based networks that include a transmission media and at least one I/O processor connected to the transmission media by a core, a buffering device is provided that compensates for different latencies from all physical lanes in data links so that data transmission occurs at the same time in the receive path of the I/O "processor." The processor can be an I/O device for a host channel adapter, a target channel adapter, or an interconnect switch in an InfiniBand-type network.



Google Exhibit 1071
Google v. VirtaMove

Oracle Ex. 1071
Oracle v. VirtaMove, IPR2025-00981

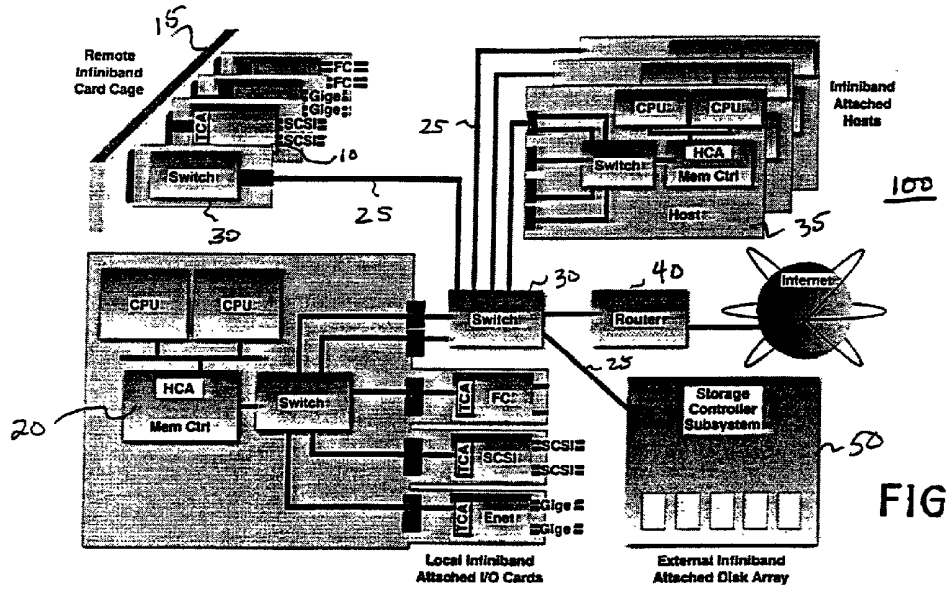

15
Remote Infiniband Card Cage
FC
FC
Gige
Gige
SCSI
SCSI
TCA
10
Switch
30
25
Infiniband Attached Hosts
CPU
CPU
HCA
Mem Ctrl
Switch
Host
35
100
25
30
40
Router
Internet
CPU
CPU
HCA
Mem Ctrl
20
Switch
TCA
FC
TCA
SCSI
SCSI
TCA
Enet
Gige
Gige
Storage Controller Subsystem
50
Local Infiniband Attached I/O Cards
External Infiniband Attached Disk Array

FIG. 1a

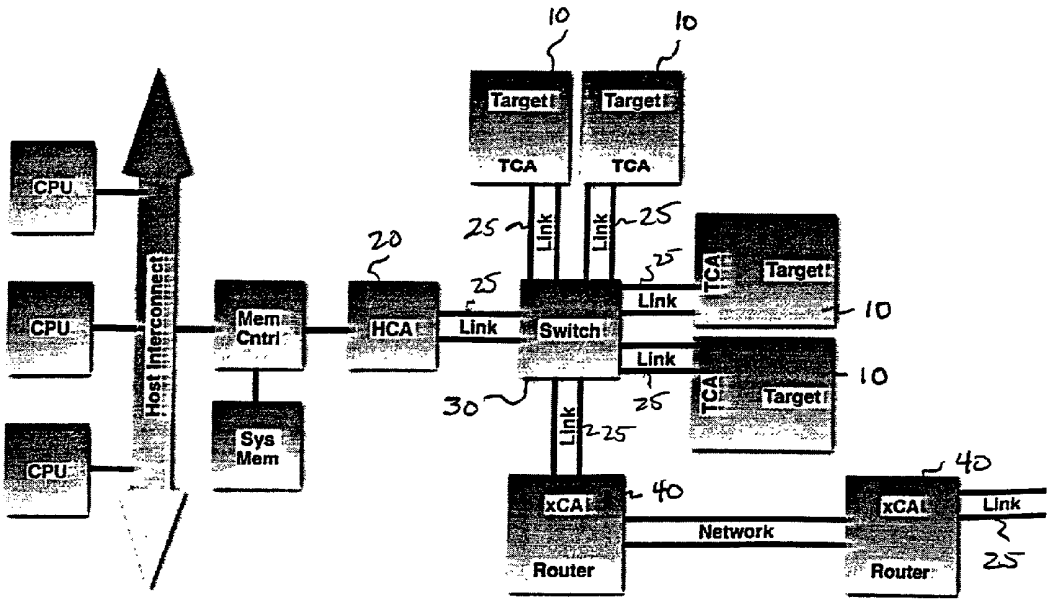

10
10
Target
Target
TCA
TCA
25
Link
Link
25
CPU
CPU
CPU
Host Interconnect
Mem Cntrl
Sys Mem
20
HCA
25
Link
Switch
25
Link
Target
TCA
10
Link
Target
TCA
10
30
Link
25
xCAL
40
Network
Router
xCAL
40
Link
25
Router

FIG. 1b

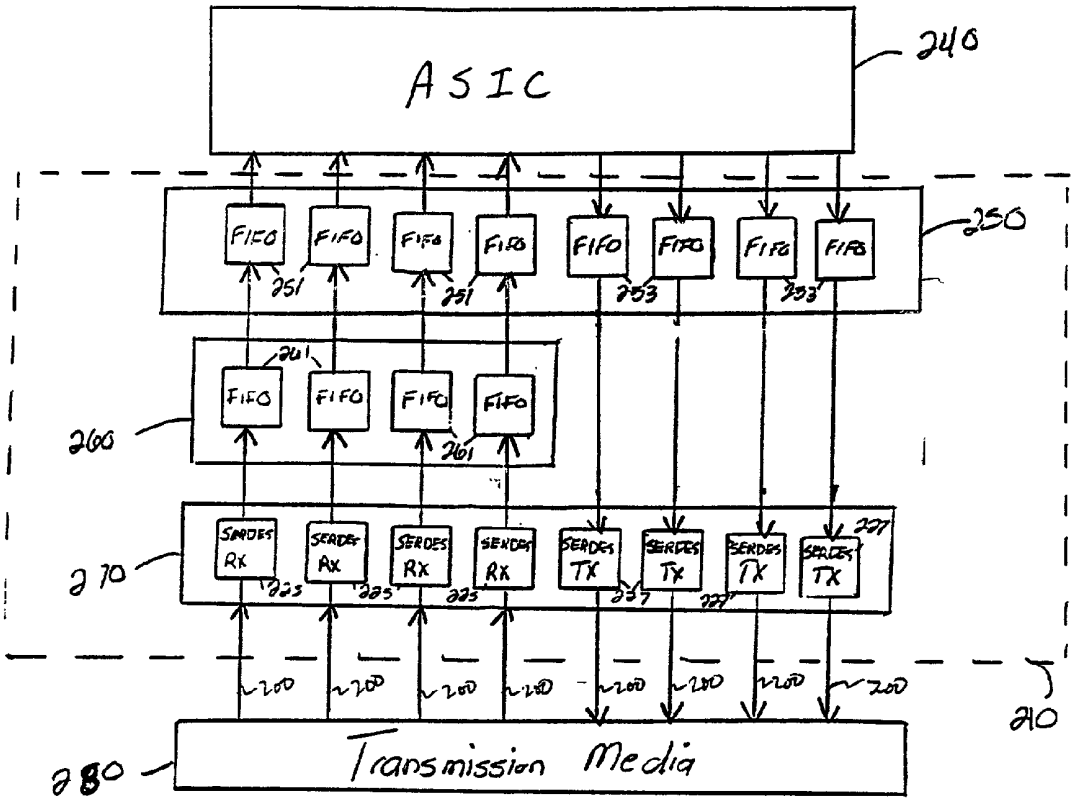

ASIC
240
FIFO
250
260
270
SERDES RX
SERDES TX
280
Transmission Media
210

FIG. 2

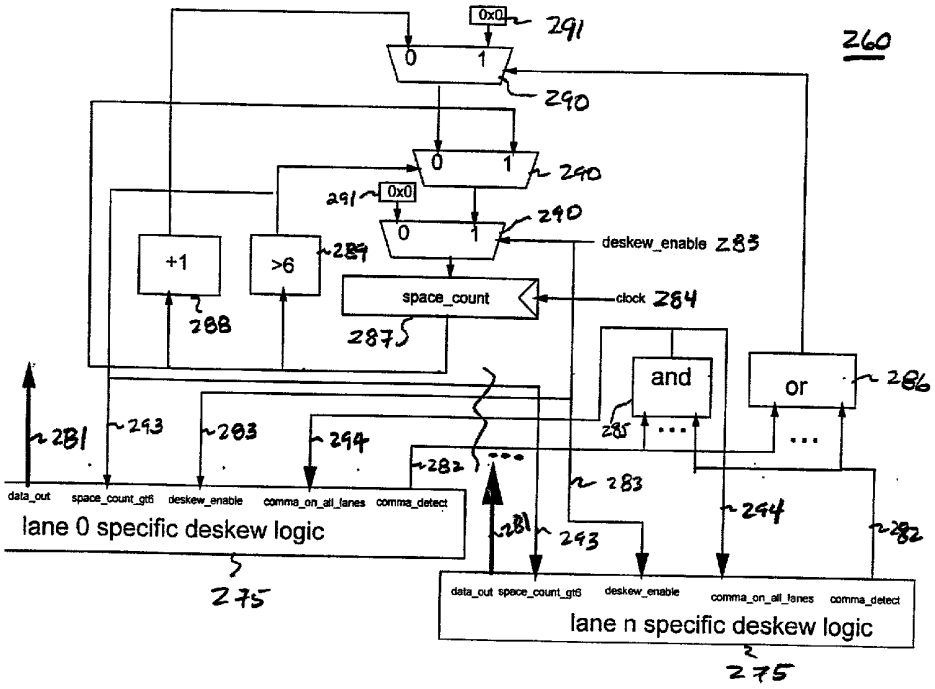

0x0
291
260
0
1
290
+1
>6
289
288
space_count
287
clock 284
deskew_enable 285
and
or
286
281
293
283
294
282
data_out
space_count_gt6
deskew_enable
comma_on_all_lanes
comma_detect
lane 0 specific deskew logic
275
lane n specific deskew logic

FIG. 3a

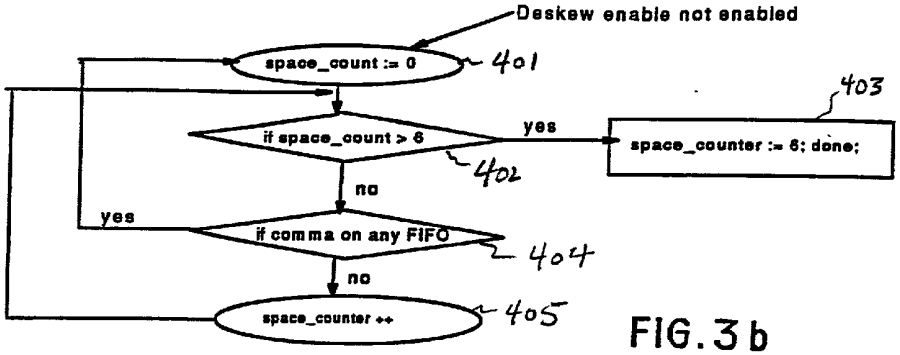

Deskew enable not enabled
space_count := 0
401
if space_count > 6
yes
space_counter := 6; done;
403
402
no
yes
if comma on any FIFO
404
no
space_counter ++
405

FIG. 3b

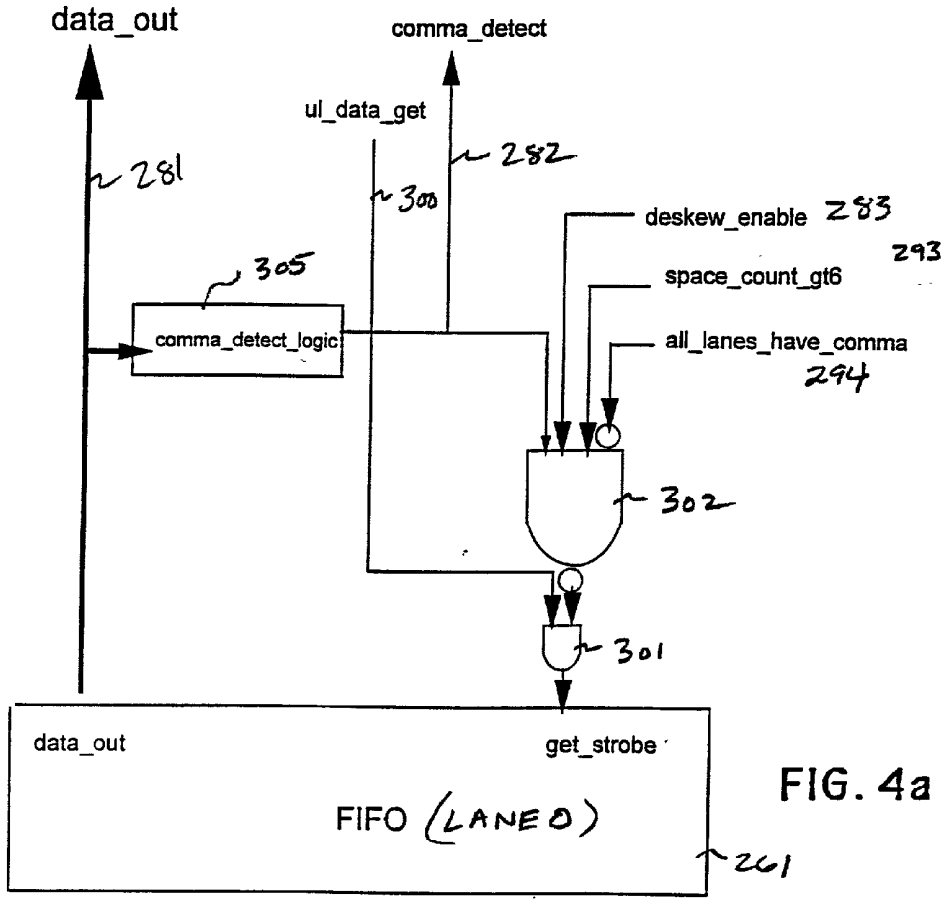

data_out
comma_detect
ul_data_get
281
282
300
deskew_enable
283
293
space_count_gt6
305
comma_detect_logic
all_lanes_have_comma
294
302
301
data_out
get_strobe
FIFO (LANE 0)
261

FIG. 4a

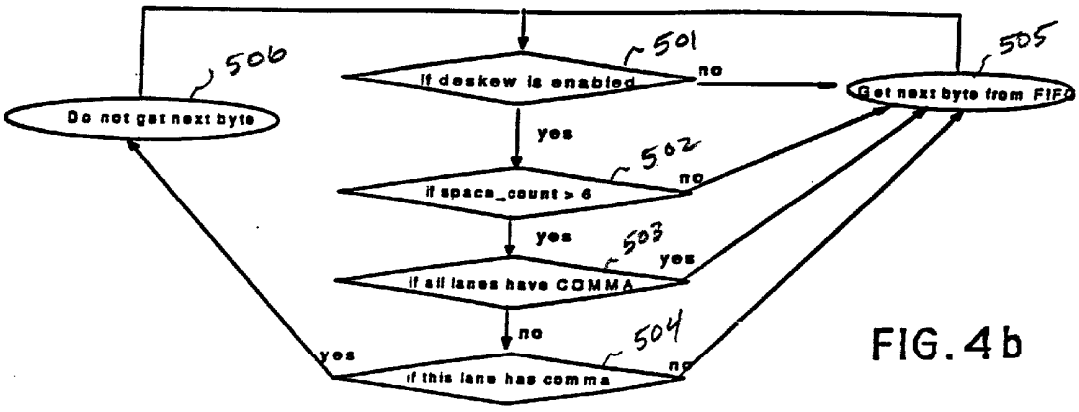

506
501
505
If deskew is enabled
no
Get next byte from FIFO
Do not get next byte
yes
502
if space_count > 6
no
yes
503
yes
if all lanes have COMMA
no
504
yes
if this lane has comma
no

FIG. 4b

PROCESSOR FOR DETERMINING PHYSICAL LANE SKEW ORDER

BACKGROUND OF THE INVENTION

[0001] 1. Field of the Invention

[0002] The present invention generally relates to input/output (I/O) data transmission devices, and more particularly to first-in-first-out (FIFO) buffer devices in I/O data transmission paths that compensates for lane skew order.

[0003] Description of the Related Art

[0004] InfiniBand (registered Trademark of the InfiniB and Trade Association, Portland, Oregon) architecture is a new common I/O specification to deliver a channel based, switched-fabric technology that the entire hardware and software industry can adopt. A network and components associated with an InfiniBand network **100** are shown in **FIG. 1***a*. InfiniBand based networks are designed to satisfy bandwidth-hungry network applications, such as those combining voice, data, and video on the Internet. InfiniBand architecture is being developed by the InfiniBand Trade Association that includes many hardware and software companies. Its robust layered design enables multiple computer systems and peripherals to work together more easily as a single high-performance and highly available server.

[0005] Being a fabric-centric, message-based architecture, InfiniBand is ideally suited for clustering, input/output extension, and native attachment in diverse network applications. InfiniBand technology can be used to build remote card cages **15** or connect to attached hosts **35**, routers **40** , or disk arrays **50**. InfiniBand also features enhanced fault isolation, redundancy support, and built-in failover capabilities to provide high network reliability and availability. Featuring high-performance and reliability, these devices provide solutions for a range of network infrastructure components, including servers and storage area networks.

[0006] In **FIG. 1***b*, a block diagram is shown in exemplary form of InfiniBand components in a portion of the network shown in **FIG. 1***a*. These components have input/output interfaces, each forming part of a target channel adapter (TCA) **10**, host channel adapter (HCA) **20**, an interconnect switch device **30**, and routers **40**, each that have application specific integrated circuits (ASIC) core interfaces that include InfiniBand Technology Link Protocol Engine (IBT-LPE) cores that connect ASICs between each of these components through links **25** in an InfiniBand Technology (IBT) network **100**. The IBT-LPE core supports a range of functionality that is required by all IBT devices in the upper levels of the physical layer and the lower link layer. It also handles the complete range of IBT bandwidth requirements, up to and including a 4-wide link operating at 2.5 gigabits per second. The IBT-LPE core (large integrated circuit design) in the upper levels of the physical layer and the link layer core of the ASIC comply with standards established by the InfiniBand Trade Association in the IBTA 1.0 specifications (2001). Such architectures decouple the I/O subsystem from memory by using channel based point to point connections rather than shared bus, load and store configurations.

[0007] The TCA **10** provides an interface for InfiniBand-type data storage and communication components. Creating InfiniBand adapters that leverage the performance benefits of the InfiniBand architecture is accomplished through a cooperative, coprocessing approach to the design of an InfiniBand and native I/O adapter. The TCA **10** provides a high-performance interface to the InfiniBand fabric, and the host channel communicates with a host based I/O controller using a far less complex interface consisting of queues, shared memory blocks, and doorbells. Together, the TCA and the I/O controller function as an InfiniBand I/O channel deep adapter. The TCA implements the entire mechanism required to move data between queues and to share memory on the host bus and packets on the InfiniBand network in hardware. The combination of hardware-based data movement with optimized queuing and interconnect switch priority arbitration schemes working in parallel with the host based I/O controller functions maximizes the InfiniBand adapter's performance.

[0008] The HCA **20** enables connections from a host bus to a dual 1× or 4× InfiniBand network. This allows an existing server to be connected to an InfiniBand network and communicate with other nodes on the InfiniBand fabric. The host bus to InfiniBand HCA integrates a dual InfiniBand interface adapter (physical, link and transport levels), host bus interface, direct memory target access (DMA) engine, and management support. It implements a layered memory structure in which connection-related information is stored in either channel on-device or off-device memory attached directly to the HCA. It features adapter pipeline header and data processing in both directions. Two embedded InfiniBand microprocessors and separate direct memory access (DMA) engines permit concurrent receive and transmit data-path processing.

[0009] The interconnect switch **30** can be an 8-port 4× switch that incorporates eight InfiniBand ports and a management interface. Each port can connect to another switch, the TCA **10**, or the HCA **20**, enabling configuration of multiple servers and peripherals that work together in a high-performance InfiniBand based network. The interconnect switch **30** integrates the physical and link layer for each port and performs filtering, mapping, queuing, and arbitration functions. It includes multicast support, as well as performance and error counters. The management interface connects to a management processor that performs configuration and control functions. The interconnect switch **30** typically can provide a maximum aggregate channel throughput of 64 gigabits, integrates buffer memory, and supports up to four data virtual lanes (VL) and one management VL per port.

[0010] **FIG. 2** illustrates the core logic **210** that connects an InfiniBand transmission media **280** (the links **25** shown in **FIG. 1***b*) to an application specific integrated circuit (ASIC) **240** (such as the TCA **10**, the HCA **20**, the switch **30**, the router **40**, etc. as shown in **FIG. 1***b*). The core logic **210** illustrated in **FIG. 2** is improved using the invention described below. However, the core logic **210** shown in **FIG. 2** is not prior art and may not be generally known to those ordinarily skilled in the art at the time of filing of this invention. Received data through SERDES **225** over the lanes **200** in the data links can be in data packets.

[0011] To accommodate the different speeds of the data signals being handled, the core logic **210** includes a serialization portion **270** that includes the serialization/deserialization (SERDES) units **225**. The structure and operation of

such serialization/deserialization units is well known to those ordinarily skilled in the art and such will not be discussed in detail herein so as not to unnecessarily obscure the salient features of the invention.

[0012] The InfiniBand transmission media **280** is made up of a large number of byte-striped serial transmission lanes **200** that form the links **25**. The receive serialization/deserialization units **225** deserialize the signals from the transmission media **280** and perform sufficient conversion to a frequency that is acceptable to the core logic **210**. For example, if the serialization/deserialization receive units **225** operate to deserialize 10 bits at a time, a 10-to-1 reduction occurs that reduces the 2.5 gigabit per second speed on the transmission media **280** into a 250 MHz frequency that is acceptable to the core logic **210**.

[0013] The core logic **210** also includes a data control unit **260**. The frequency of the signal propagating along the transmission media **280** may not always occur at this wire speed, but instead may be slightly above or below the desired frequency (e.g., by up to 100 parts per million) or control proper data transmit to the upper link layer that is synchronized. This inconsistency in the frequency is transferred through the serialization/deserialization units **225**. The data control unit **260** includes FIFO buffers **261** that buffer the data being received by the serialization/deserialization units (SERDES) **225** so as to provide the signal in a uniform 250 MHz frequency to the upper layer logic **250**.

[0014] The upper link layer logic **250** includes additional FIFO buffers **251** that convert the frequency of the signal output from the frequency correction unit **260** into a frequency that is acceptable to the ASIC **240**. During transmission of a signal from the ASIC **240** to the transmission media **280**, the process is reversed and the upper link layer logic **250** use FIFO buffers **253**. The serialization unit **270** uses other transmission serialization/deserialization units **227**. Note that no correction is required by the frequency correction unit **262** for signals that are being transmitted to the transmission media **280** because the ASIC **240** generally produces a signal that does not need to be corrected.

[0015] One disadvantage of current "parallel-serial" type architectures include latency differences between the transmission media **280** and receive buffering devices **261**. This problem requires a core that includes a controller for a receive buffer device that compensates for these different latencies such that detection of pieces of the data from all physical lanes **200** in the data links that occur in data packets from all physical lanes **200** are received at the same time.

SUMMARY OF THE INVENTION

[0016] The present invention resolves the above problem in "parallel-serial" architectures that include a transmission media and at least one I/O processor connected to the transmission media by providing a core that communicates with a remote transmitter processor using a training sequence protocol. A buffering device compensates for different latencies from all physical lanes in data links so that data transmission occurs at the same time in the receive path of the I/O "processor." The processor can be an I/O device for a host channel adapter, a target channel adapter, or an interconnect switch.

[0017] The core includes a logic layer connected to the processor, serial lanes connecting the logic layer to the transmission media, and receive and transmit buffers within the serial lanes. The receive buffers have a control to correct for fluctuations in the transmission media and alters the signals being processed along the serial lanes to compensate for different latencies from all physical lanes in the data links so that data transmission occurs at the same time. The invention can be part of a processor that includes serializer/deserializers within the serial lanes. The receive buffers are preferably elastic first-in, first-out (FIFO) buffers and are external to the upper logic layer.

[0018] In another embodiment, the invention encompasses a parallel-serial architecture communication system that compensates for different latencies in data lanes of the system. The system includes a parallel-serial transmission media, a processor in communication with the transmission media, a plurality of serial lanes providing data from the transmission media to the processor, buffers interposed along each of the serial lanes, and training logic connected to the buffers. Upon receiving a data communication from the transmission media, the training logic pauses each of the buffers until all of the buffers are synchronized. The training logic simultaneously restarts the buffers after the buffers are synchronized.

[0019] The training logic operates on a training sequence contained within the data communication. The training logic also includes an AND gate that determines when all of the buffers have reached a predetermined point in the training sequence. The training logic stops data processing through each of the buffers until all of the buffers reach the predetermined point in the training sequence. When all of the buffers reach the predetermined point in the training sequence, all of the buffers are simultaneously restarted. The training sequence includes an initial training sequence identifier and a final training sequence identifier. The final training sequence identifier is the aforementioned "predetermined point".

[0020] The training sequence also includes a predetermined data stream between the initial training sequence identifier and the final training sequence identifier. The training logic allows each of the buffers to reach the predetermined point only when a training quantity (equal to a number greater than a maximum skew value between adjacent ones of the data lanes) of the data stream has been processed by each of the buffers.

[0021] The invention also includes a method of compensating for different latencies in data lanes in a parallel-serial architecture. The method receives a data communication from a transmission media, provides the data communication to buffers interposed along a plurality of serial lanes, pauses each of the buffers until all of the buffers are synchronized, simultaneously restarts the buffers after the buffers are synchronized, and forward synchronized data from the buffers to a processor.

[0022] The pausing process is based on a training sequence contained within the data communication and determines when all of the buffers have reached a predetermined point in the training sequence. The invention stops data processing through each of the buffers until all of the buffers reach the predetermined point in the training sequence. When all of the buffers reach the predetermined point in the training sequence, the invention restarts all of the buffers simultaneously.

[0023] The training sequence includes an initial training sequence identifier and a final training sequence identifier (e.g., the predetermined point). The training sequence also includes a predetermined data stream between the initial training sequence identifier and the final training sequence identifier. The invention allows each of the buffers to reach the predetermined point only when a training quantity (a number greater than a maximum skew value between adjacent ones of the data lanes) of the data stream has been processed by each of the buffers.

[0024] Thus, the invention performs a training sequence to correct skew between the physical lanes. The training sequence pauses each of the FIFO buffers at the same point in the training process. Once trained, all buffers are then simultaneously restarted to allow the FIFO buffers to be synchronized in their data transmission.

BRIEF DESCRIPTION OF THE DRAWINGS

[0025] The foregoing and other objects, aspects and advantages will be better understood from the following detailed description of a preferred embodiment(s) of the invention with reference to the drawings, in which:

[0026] **FIG. 1***a* shows a schematic diagram of an exemplary InfiniBand network in which the device of the invention is preferably used;

[0027] **FIG. 1***b* is a portion of the network with components as shown in **FIG. 1***a;*

[0028] **FIG. 2** is a schematic diagram of a core that provides transmission between an ASIC and a transmission media;

[0029] **FIG. 3***a* is a diagram of deskew logic counter of the core;

[0030] **FIG. 3***b* is a flow diagram for determining physical lane skew order in the core as to the deskew logic counter shown in **FIG. 3***a;*

[0031] **FIG. 4***a* is a schematic diagram of a lane specific deskew logic counters used to implement the invention; and

[0032] **FIG. 4***b* is a flow diagram for determining lane skew order in **FIG. 4***a*.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS OF THE INVENTION

[0033] As mentioned above, there is a need for a receive buffering device that compensates for different latencies from all physical lanes **200** in data links, so that data transmission occurs at the same time in the receive path of the ASIC **240**. The invention uses an exemplary processor shown in **FIG. 2** (which is shown in greater detail in **FIG. 3***a*) that includes deskew logic counters **275** that cooperatively control the elastic buffering devices **261**. The lane deskew counters **275** are replicas for each lane required in the receiving processor. In InfiniBand components, typically one, four or twelve lanes are used on the receive data paths (note only four exemplary lanes are shown in **FIG. 2**). Each of the elastic buffering devices **261** reside between the upper link layer logic **250** and the lower link layer **270**. In "parallel-serial" architectures, individual serial lanes **200** may have differing latency from the transmission media **280** to respective receive data transmission buffer devices **261**. Thus, the receiver section of the core **210** compensates for the different latencies. The receiver section **210** can detect pieces of the data from all physical lanes **200** that are transmitted at the same time from a remote transmitting processor (e.g., an HCA **20**, switch **30** etc.).

[0034] Referring now to **FIG. 3***a,* a deskew logic counters **275** is implemented in the invention as part of the frequency correction and data control unit **260**. This design is shown for lane "0" and lane "n" only in the receive data path of the processor shown in **FIG. 2** (the intervening lanes are omitted for clarity and would be replicated for each receive lane). Each of the deskew logic counters **275** includes an AND gate **285** and OR gate **286** with input signals as shown. A box labeled "+1"**288** increments input form a space_count latch register **287**. A box labeled ">6"**289** indicates whether input is greater than six. The 0×1 boxes are selectors **290** that perform the necessary select signal control function. The 0×0 labeled boxes **291** input a zero value into the selectors **290** when selected and passed to the space_count latch register **287** to be zero. When a path to a 0×0 box **290** is selected, it loads the space_count register **287** with all zeros.

[0035] A data_out signal **281** includes data from a FIFO **261** for lane **0**. A comma_detect signal **282** asserts '1' if a byte on data_out **281** from the particular FIFO **261** is a "comma." A deskew_enable signal **283** enables the counter **275** from upper layer logic **250** to deskew the receive lanes **200**. A clock signal **284** is provided by a clock signal from the upper layer logic **250**. The comma_detect signal **292** for each lane having a specific deskew logic signal **292** provides the necessary logic that binds all of the lane specific deskew logic counters **275** together. This signal comes from the comma_detect output on the specific deskew logic counter **275** for a specified lane n. A space_count_gt**6** signal **293** provides an indication from central deskew logic that its space_count_indication is greater than six. A comma_on$_{13}$ all_lanes signal **294** provides indication from the central deskew logic that all lanes have a comma on respective data$_{13}$ out signal **281**.

[0036] Referring now to **FIG. 3***b,* the operation of the structure shown in **FIG. 3***a* is explained with reference to the flow diagram for determining physical lane skew order in the core **210** using the deskew counters **275**. The "deskew not enable" command **401** corresponds to the signal **283** for each lane used labeled "space_count:=0" that loads the space_count register **287** with a zero value. If the decision logic step **402** labeled "space_count>6," is satisfied, then the latch register **287** is loaded with **6** at step **403**. Next step **404** detects whether a "comma" is on any FIFO **261**. If so, then space count latch register **287** returns to the beginning step **401** and starts the process again. If not, the second command step **405** labeled "space_counter++" increments the space_count register **287** by one and returns to step **402**.

[0037] Each of the elastic buffer devices **261** on the receive path of the core **210** is controlled by a process in the upper link layer logic **250** that includes a controller including the deskew counter **275** that counts symbol times in which a "comma" symbol has not been detected on any of the receiving physical lanes **200**. If that counter **275** reaches a value of 7 (the maximum amount of skew between any two physical lanes is 6 symbol times) or higher, the next "comma" symbol detected is the earliest physical lane to be deskewed. The invention stops taking data from that physi-

cal lane FIFO **261**, and from each FIFO **261** where a "comma" is detected, until all lanes **200** have a "comma" symbol at the FIFO **261** output.

[**0038**] In other words, the invention performs a training sequence to correct skew between the physical lanes. The training sequence pauses each of the FIFO buffers at the same point in the training process. Once trained, all buffers are then simultaneously restarted to allow the FIFO buffers to be synchronized in their data transmission.

[**0039**] The training sequence comprises an initial comma followed by a specific number of bytes of data (e.g., 15 bytes of data), followed by a final comma. The data is essentially meaningless and is not used, but does function as a place-holder within the memory locations of the FIFO buffers. This allows all memory locations within each of the buffers to be full when the meaningful data begins to be transmitted, and maintains the synchronization of the FIFO buffers operations.

[**0040**] In operation, an external device, such as an HCA **20**, switch **30**, etc. sends data preceded by a training sequence. The format of the training sequence matches a predetermined standard (e.g., an initial comma, followed by 15 bytes of meaning less data, followed by a final comma). The initial commas represent the beginning of the training sequence and the final commas represent the end of the training sequence. The meaningless 15 bytes of data is greater than the size of the buffers and allows all memory locations within the FIFO buffers to be completely full when the second set of commas (representing the end of the training sequence) are received.

[**0041**] When the end of the training sequence is identified, each FIFO buffers is stopped until all FIFO buffers received the training end comma. Once all FIFO buffers have received the training end comma, all FIFO buffers are simultaneously restarted. This allows all FIFO buffers to begin the process of sequentially passing data within their internal memory locations at the same time, which synchronizes the output of the FIFO buffers. Once the meaningless training data is cleared out of the memory locations of the FIFO buffers, actual data transmission will take place.

[**0042**] To train a set of physical lanes, the FIFOs on each lane are monitored to determine when all FIFOs have receive comma, which represents the beginning of a training sequence. When the training sequence (all commas) has been received by all FIFOs, the FIFOs are started and the number of cycles since the beginning of the training sequence is counted. As shown in **FIG. 3***b,* the data byte count that occurs after the first comma is received, must exceed the maximum amount of skew that is possible between any two physical lanes. After that count is reached by each FIFO, the FIFO then looks for the final comma. Once that final comma is received by the FIFO, the FIFO stops receiving data until all other FIFOs also received the final comma. Once all FIFOs receive the final comma, the invention simultaneously begins meaningful data input into all lane FIFOs, which allows the meaningful data output from the FIFOs to be synchronized without skew.

[**0043**] This is shown in **FIG. 4***a,* where one of the elastic FIFO's **261** with the related deskew logic counter **275** for lane "**0**" is shown. This architecture is replicated in the core for each receive serial lane used. A comma_detect_logic section **305** comprises a logic section that determines if the data_out signal **281** for a particular lane is a comma symbol. The "comma" symbol is also the start of a training sequence. The data_out signal **281** includes data from the FIFOs **261** for that particular lane. The comma_detect signal **282** asserts '**1**' if a byte on data_out **281** is a "comma." An "upper layer (ul)_data_get" signal **300** from the upper layer logic **250** is an input signal for a particular lane as to a pause signal, that in turn is input to AND gate **301** that also receives a complimented signal from NOR gate **302**. The NOR gate **302** receives and combines logically signals from: the deskew_enable signal **283** that enables the deskew logic to operate; the all_lanes_have_comma signal **294** that provides an indication from the deskew logic counter **275** that all lanes have a "comma" on respective data_out signals **281** from each FIFO **261** on the receive data path; and the space_count_gt**6**: signal **293** provides indication from the deskew logic counters **275** that its space_count_indication is greater than six.

[**0044**] Referring now to **FIG. 4***b,* a flow diagram is shown for a process of determining physical lane skew order in the core **210** as to one data lanes shown in **FIG. 4***a.* The process begins at initial step **501** where determination is made of whether the deskew_enable signal **283** is enabled. If not, the process goes to step **505**, wherein all actions that result correspond to an assertion of the get_strobe signal **301** shown in **FIG. 4***a.* Otherwise, the process goes to step **502**, which determines whether the space_count latch register **287** is greater than six. If not, the process goes to step **505** again. Next, the process determines whether all lanes **0**-n have a "comma" signal present. If affirmative, the process goes to step **505** again. Otherwise, the process goes to step **504** and determines whether the particular lane has a "comma." If not, the process goes to step **505** again and starts the process again. Otherwise when affirmative, all other actions result in step **506**, which corresponds to the get_strobe signal **301** being deasserted.

[**0045**] In other words, the process for specific deskew logic counter **275** in this flow diagram provides a maximum allowed skew between any two serial lanes **200** in which a training sequence used to train the unit **260**, which is provided by a remote transmitting processor (e.g., an HCA **20**) that is at least twice as long as the maximum skew between any two lanes **200**. Each of the elastic FIFOs **261** for each physical lane **200** must be capable of containing at least the amount of data representing the maximum skew between any two physical lanes **200**. Each of the elastic FIFOs **261** on the receive path must be capable of pausing. The upper link layer logic **250** determines the number of cycles since the beginning of the training sequence on any physical lane **200**. The training sequence is generated by the remote transmitting processor (e.g., the processor HCA **20**, switch **30**, etc.) external to the receiving processor shown in **FIG.2**.

[**0046**] The training sequence trains the receiving processor by use of a commonly agreed upon coded sequence for a period of time that is transmitted with the data over the links **25**. The upper link layer logic **250** provides control for the logic that pauses an individual elastic FIFO **261** once the number of cycles since the beginning of a training sequence is greater than the maximum amount of skew between any two lanes **200** and the final comma has been received.

[0047] The physical lane **200** with the least amount of latency between the attached transmission media **280** and the I/O processor is determined and correct deskew of all physical lanes **200** can be achieved. Therefore, the invention provides a control section **260** having receive buffer devices **261** that compensate for different latencies from at least two physical lanes **200** in data links of parallel-serial architectures so that data transmission occurs at the same time in the processors receive path.

[0048] The invention performs a training sequence to correct skew between the physical lanes. The training sequence pauses each of the FIFO buffers at the same point in the training process. Once trained, all buffers are then simultaneously restarted to allow the FIFO buffers to be synchronized in their data transmission.

[0049] While the invention has been described in terms of preferred embodiments, those skilled in the art will recognize that the invention can be practiced with modification within the scope of the appended claims.

What is claimed is:

1. A core that compensates for different latencies in data lanes in a parallel-serial architecture, said core comprising:

a plurality of serial lanes providing data from a transmission media to a processor;

buffers interposed along each of said serial lanes; and

training logic connected to said buffers,

wherein, upon receiving a data communication from said transmission media, said training logic pauses each of said buffers until all of said buffers are synchronized, and

wherein said training logic simultaneously restarts said buffers after said buffers are synchronized.

2. The core in claim 1, wherein said training logic operates on a training sequence contained within said data communication.

3. The core in claim 2, wherein said training logic includes an AND gate that determines when all of said buffers have reached a predetermined point in said training sequence.

4. The core in claim 3, wherein said training logic stops data processing through each of said buffers until all of said buffers reach said predetermined point in said training sequence, and, when all of said buffers reach said predetermined point in said training sequence, all of said buffers are simultaneously restarted.

5. The core in claim 3, wherein said training sequence includes an initial training sequence identifier and a final training sequence identifier, wherein said final training sequence identifier comprises said predetermined point.

6. The core in claim 5, wherein said training sequence includes a predetermined data stream between said initial training sequence identifier and said final training sequence identifier,

wherein said training logic allows each of said buffers to reach said predetermined point only when a training quantity of said data stream has been processed by each of said buffers.

7. The core in claim 6, wherein said training quantity comprises a number greater than a maximum skew value between adjacent ones of said data lanes.

8. A parallel-serial architecture communication system that compensates for different latencies in data lanes of said system, said system comprising:

a parallel-serial transmission media;

a processor in communication with said transmission media;

a plurality of serial lanes providing data from said transmission media to said processor;

buffers interposed along each of said serial lanes; and

training logic connected to said buffers,

wherein, upon receiving a data communication from said transmission media, said training logic pauses each of said buffers until all of said buffers are synchronized, and

wherein said training logic simultaneously restarts said buffers after said buffers are synchronized.

9. The system in claim 8, wherein said training logic operates on a training sequence contained within said data communication.

10. The system in claim 9, wherein said training logic includes an AND gate that determines when all of said buffers have reached a predetermined point in said training sequence.

11. The system in claim 10, wherein said training logic stops data processing through each of said buffers until all of said buffers reach said predetermined point in said training sequence, and, when all of said buffers reach said predetermined point in said training sequence, all of said buffers are simultaneously restarted.

12. The system in claim 10, wherein said training sequence includes an initial training sequence identifier and a final training sequence identifier, wherein said final training sequence identifier comprises said predetermined point.

13. The system in claim 12, wherein said training sequence includes a predetermined data stream between said initial training sequence identifier and said final training sequence identifier,

wherein said training logic allows each of said buffers to reach said predetermined point only when a training quantity of said data stream has been processed by each of said buffers.

14. The system in claim 13, wherein said training quantity comprises a number greater than a maximum skew value between adjacent ones of said data lanes.

15. A method of compensating for different latencies in data lanes in a parallel-serial architecture, said method comprising:

receiving a data communication from a transmission media;

providing said data communication to buffers interposed along a plurality of serial lanes;

pausing each of said buffers until all of said buffers are synchronized;

simultaneously restarting said buffers after said buffers are synchronized; and

forwarding synchronized data from said buffers to a processor.

16. The method in claim 15, wherein said pausing is based on a training sequence contained within said data communication.

17. The method in claim 16, wherein said pausing includes determining when all of said buffers have reached a predetermined point in said training sequence.

18. The method in claim 17, wherein said pausing stops data processing through each of said buffers until all of said buffers reach said predetermined point in said training sequence, and, when all of said buffers reach said predetermined point in said training sequence, said restarting restarts all of said buffers simultaneously.

19. The method in claim 18, wherein said training sequence includes an initial training sequence identifier and a final training sequence identifier, wherein said final training sequence identifier comprises said predetermined point.

20. The method in claim 6, wherein said training sequence includes a predetermined data stream between said initial training sequence identifier and said final training sequence identifier,

wherein said pausing allows each of said buffers to reach said predetermined point only when a training quantity of said data stream has been processed by each of said buffers,

wherein said training quantity comprises a number greater than a maximum skew value between adjacent ones of said data lanes.

* * * * *