



(19) **United States**

(12) **Patent Application Publication**
Garney et al.

(10) **Pub. No.: US 2004/0268010 A1**

(43) **Pub. Date: Dec. 30, 2004**

(54) **TRANSACTION SCHEDULING FOR A BUS SYSTEM**

Publication Classification

(76) Inventors: **John Garney**, Portland, OR (US); **John S. Howard**, Portland, OR (US)

(51) **Int. Cl.⁷** **G06F 13/36**

(52) **U.S. Cl.** **710/310**

Correspondence Address:
BLAKELY SOKOLOFF TAYLOR & ZAFMAN
12400 WILSHIRE BOULEVARD
SEVENTH FLOOR
LOS ANGELES, CA 90025-1030 (US)

(57) **ABSTRACT**

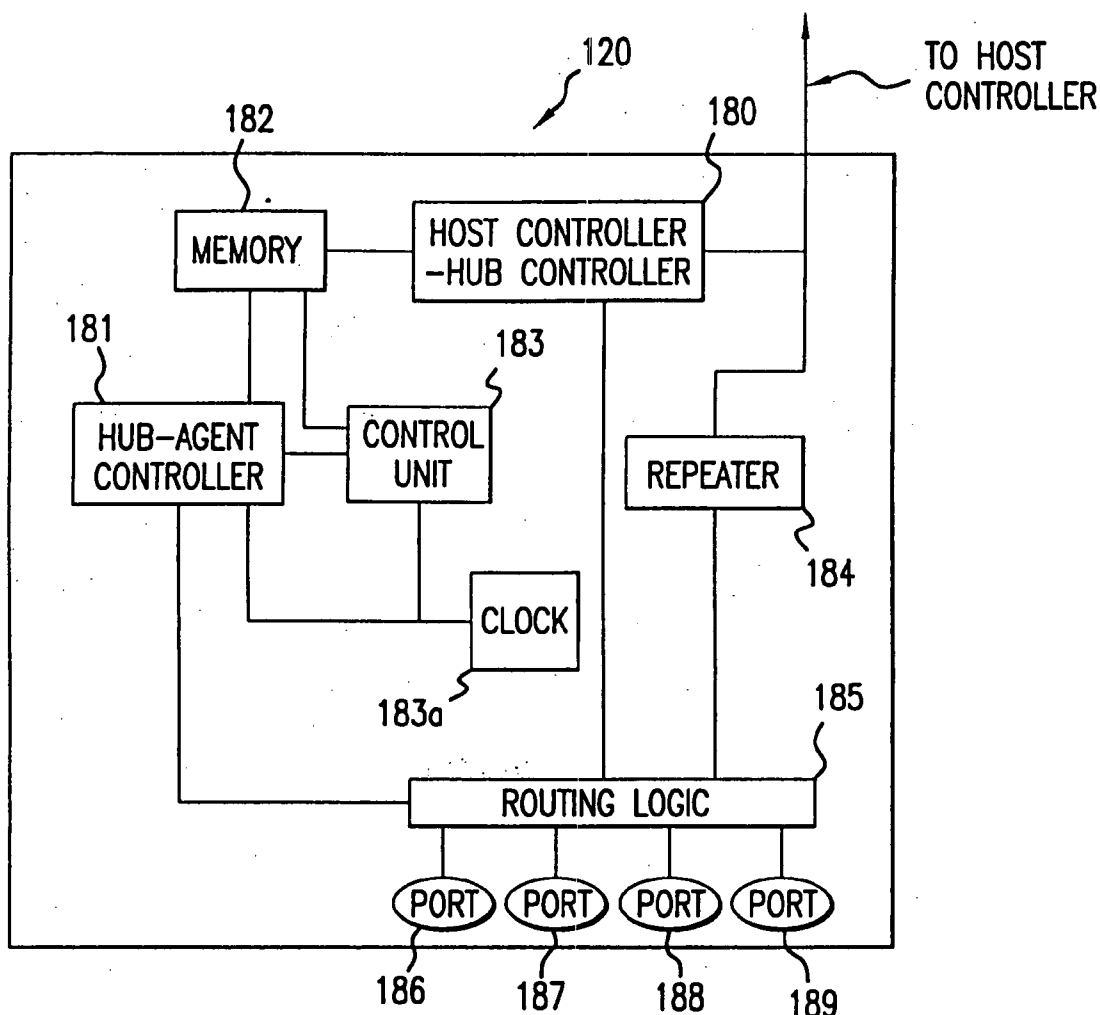
(21) Appl. No.: **10/888,665**

(22) Filed: **Jul. 8, 2004**

Related U.S. Application Data

(63) Continuation of application No. 09/361,677, filed on Jul. 27, 1999, now Pat. No. 6,792,495.

A method of and apparatus for communicating data using a hub. The method includes the step of buffering a single transfer request received at a hub during a transaction between the hub and a host controller, where the single transfer request is to be performed between the hub and an agent to generate a result. The method then includes the step of determining whether a transfer inquiry received at the hub from the host controller corresponds to the result.



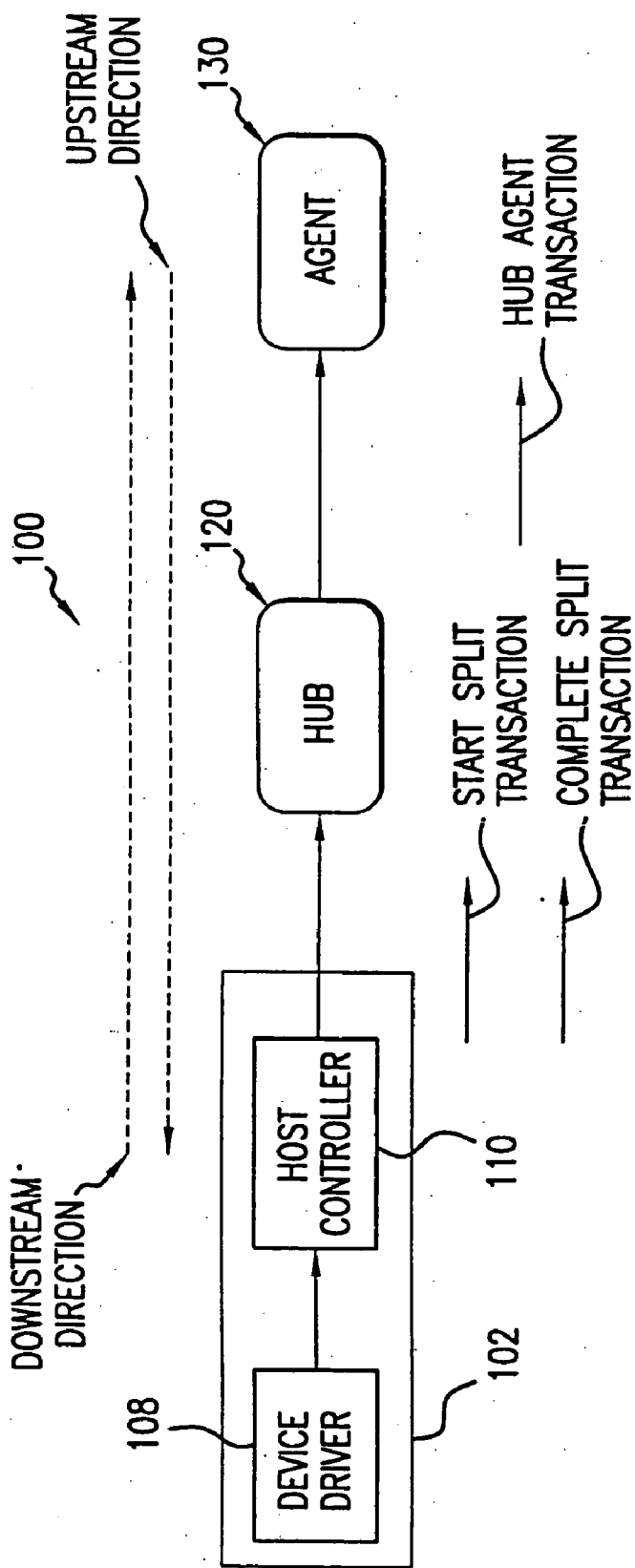


FIG.1A

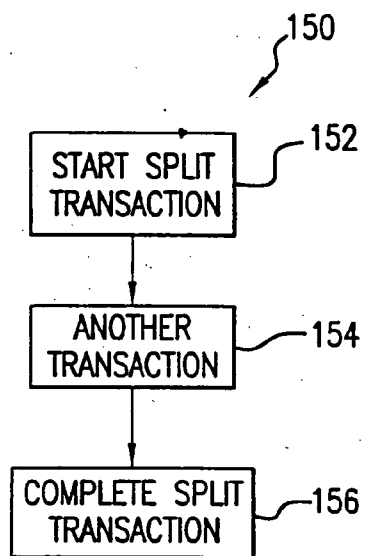


FIG.1B

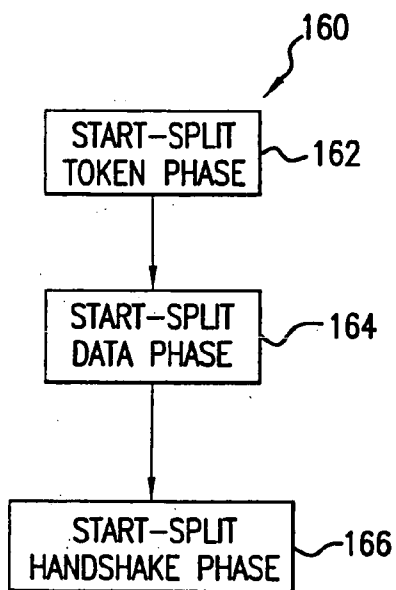


FIG.1C

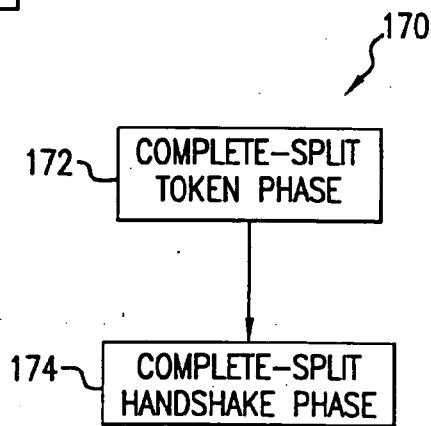


FIG.1D

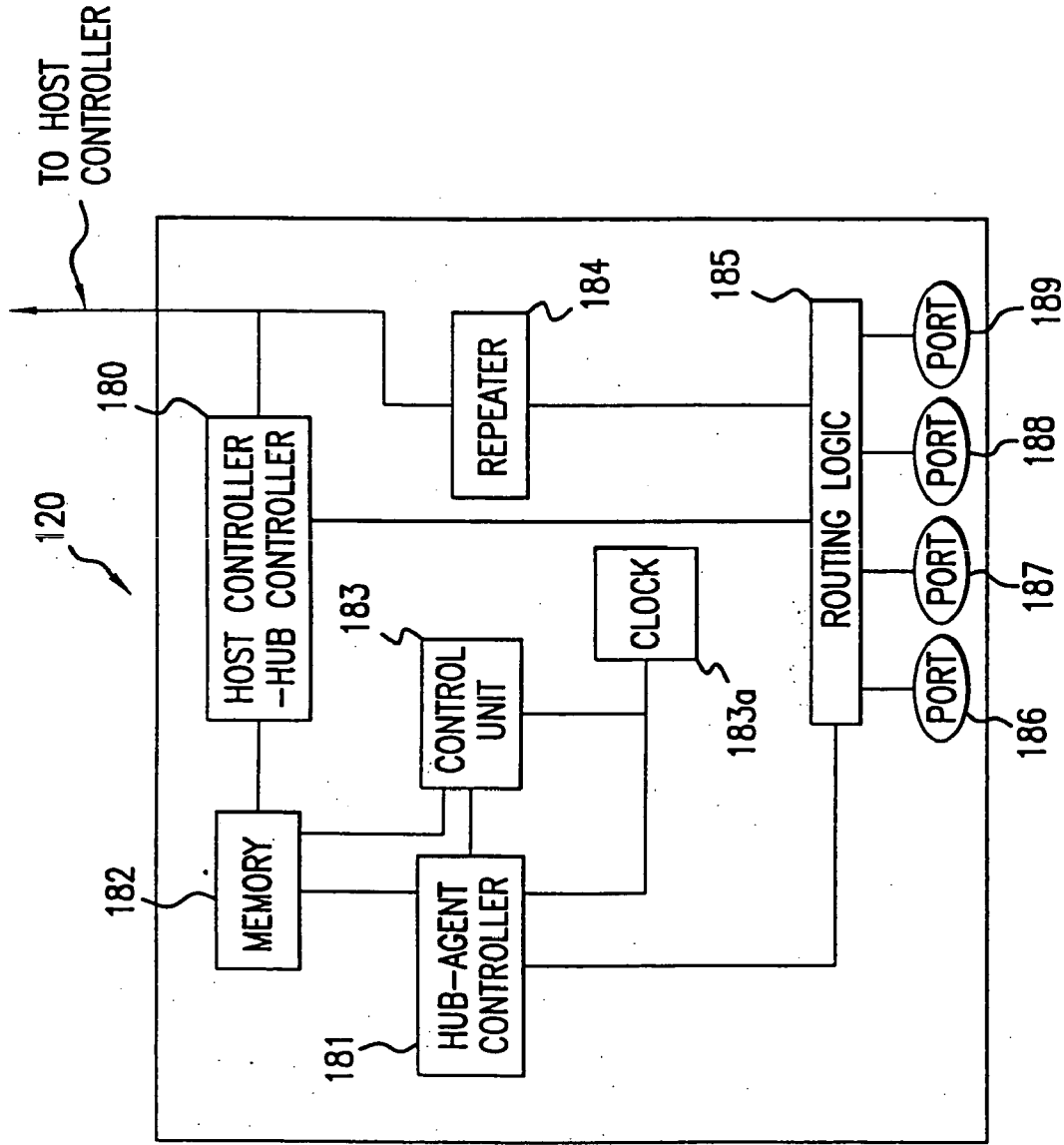


FIG.1E

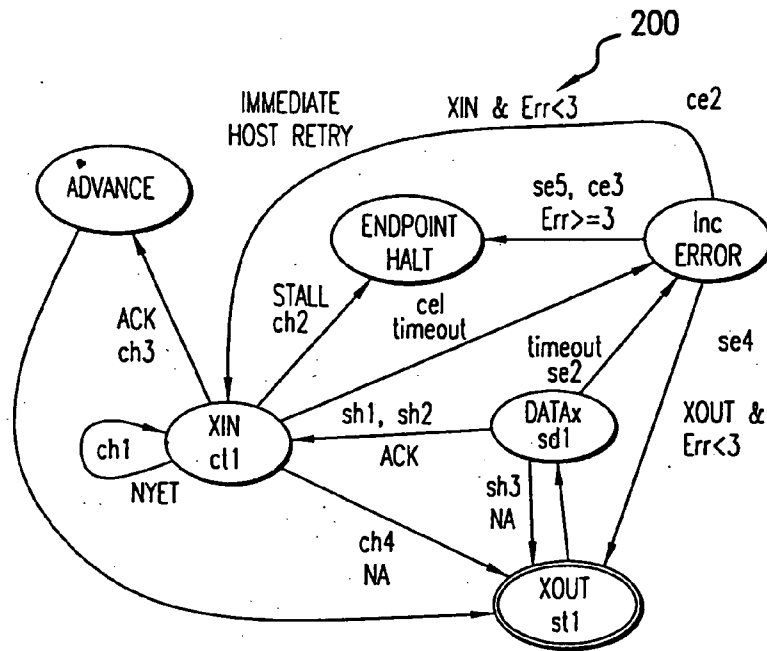


FIG.2A

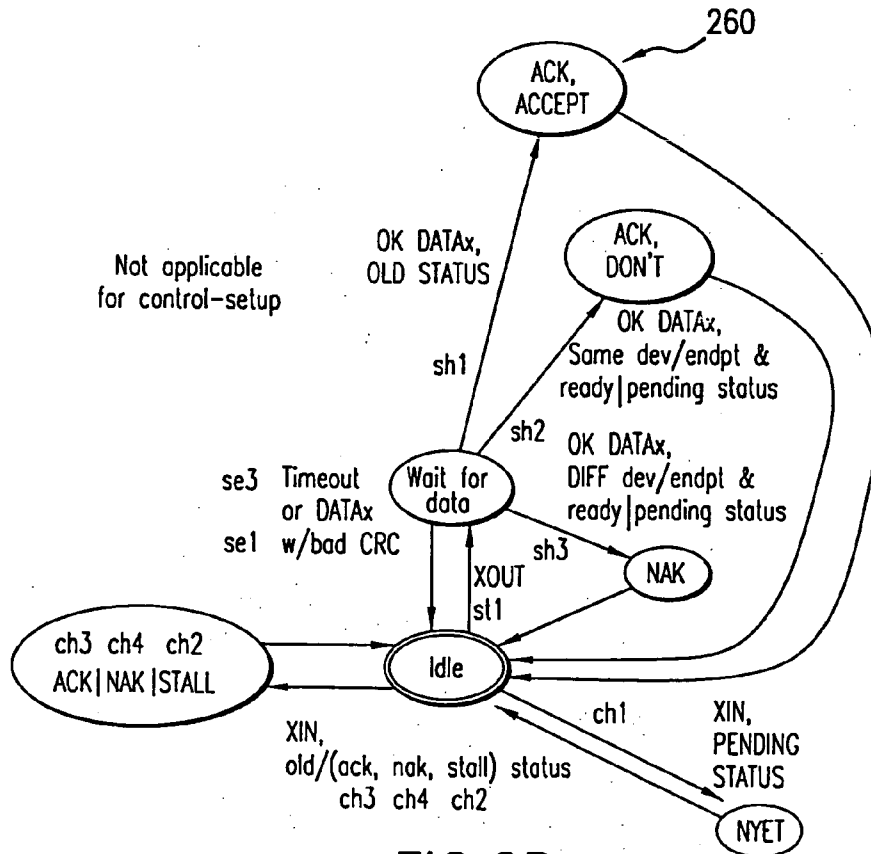


FIG.2B

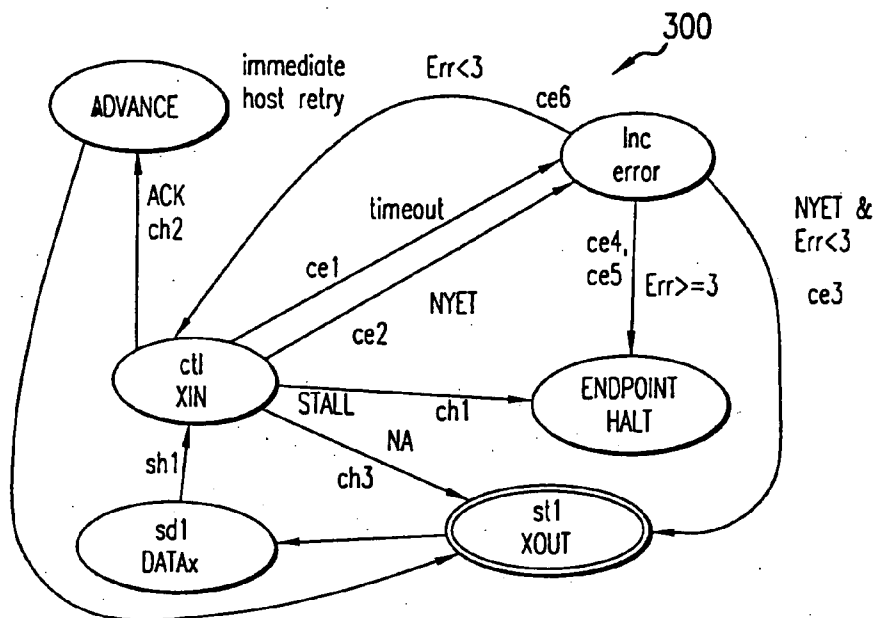


FIG.3A

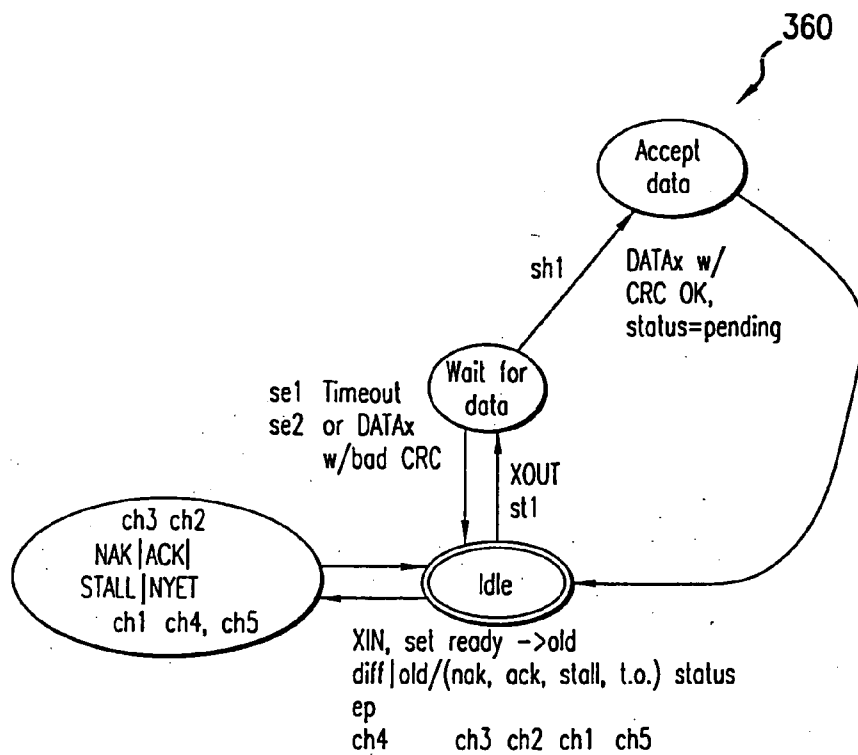


FIG.3B

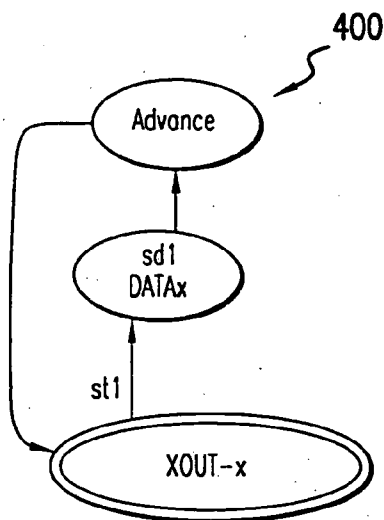


FIG. 4A

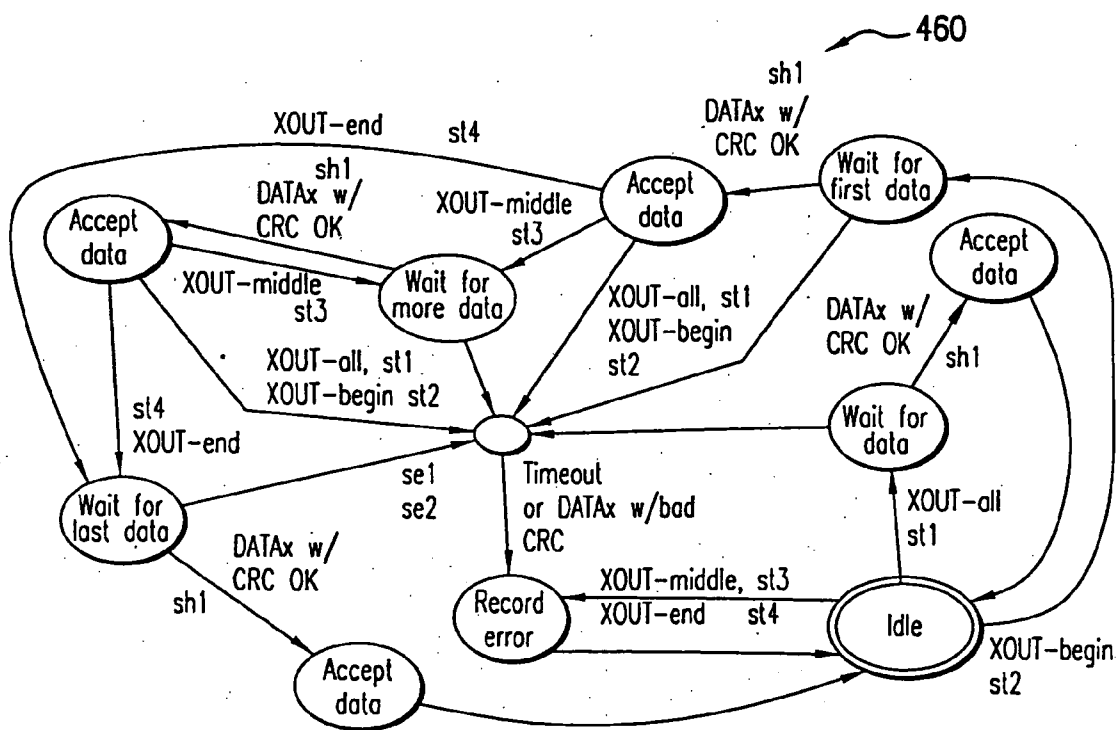


FIG. 4B

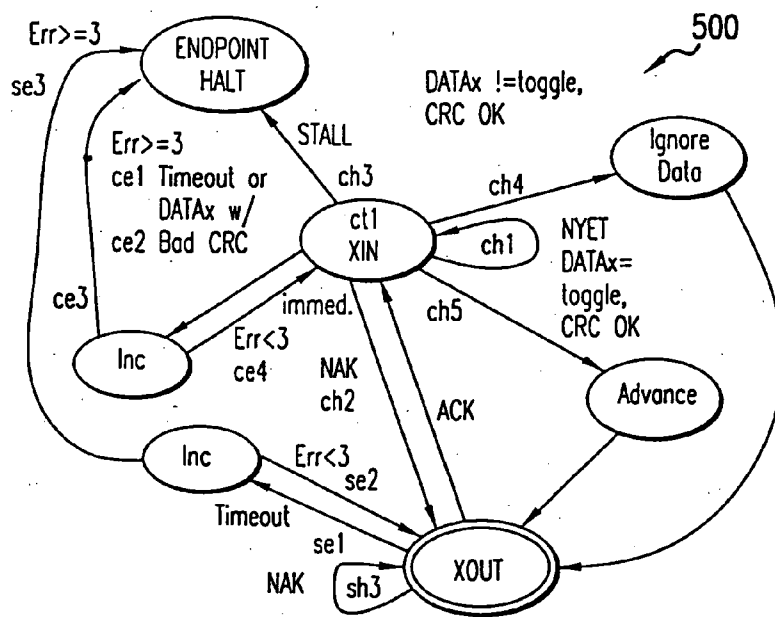


FIG.5A

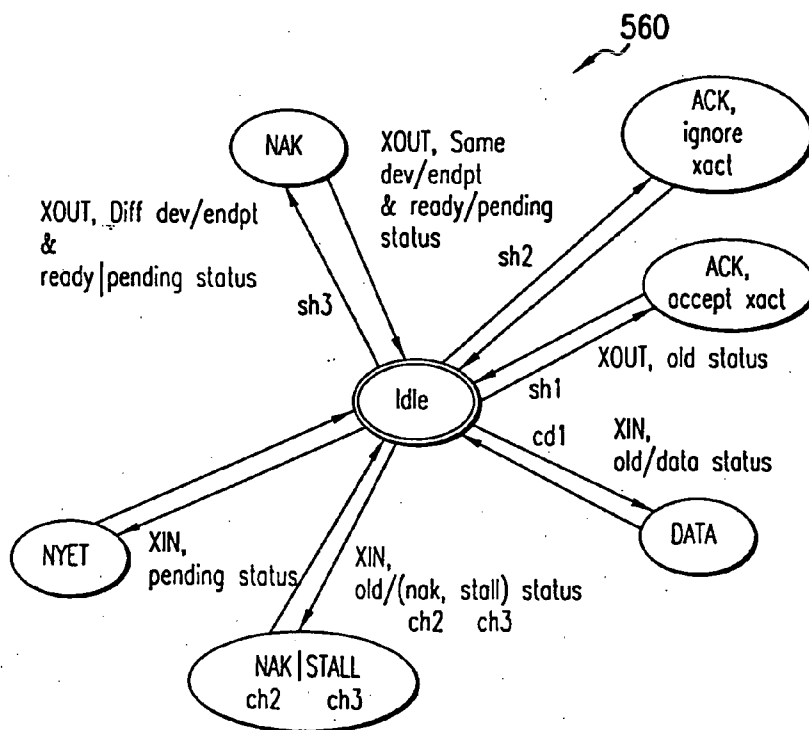


FIG.5B

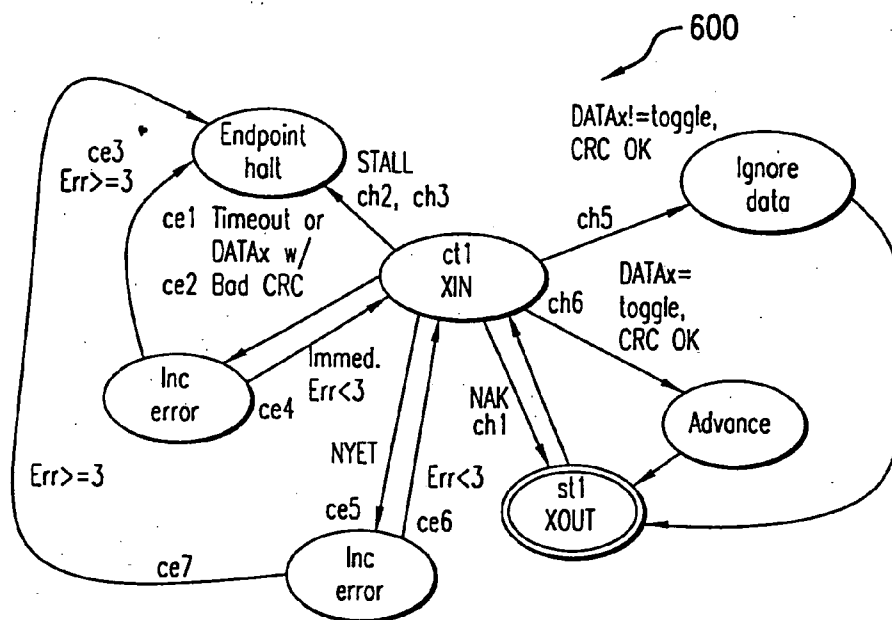


FIG.6A

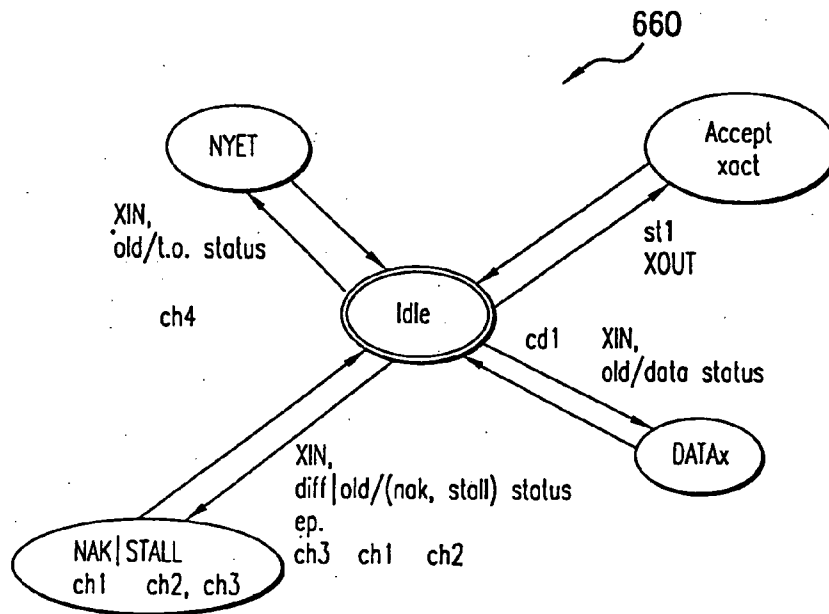


FIG.6B

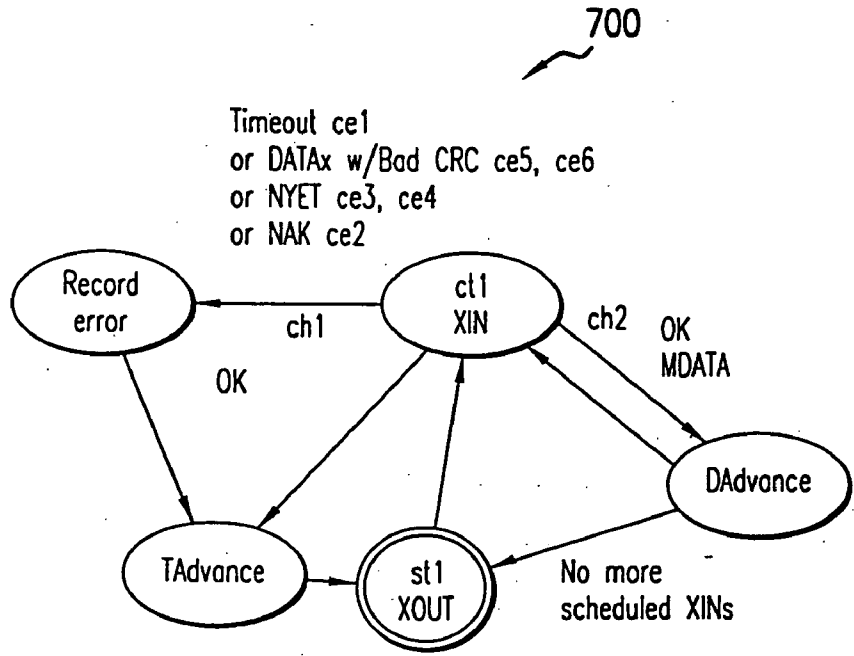


FIG.7A

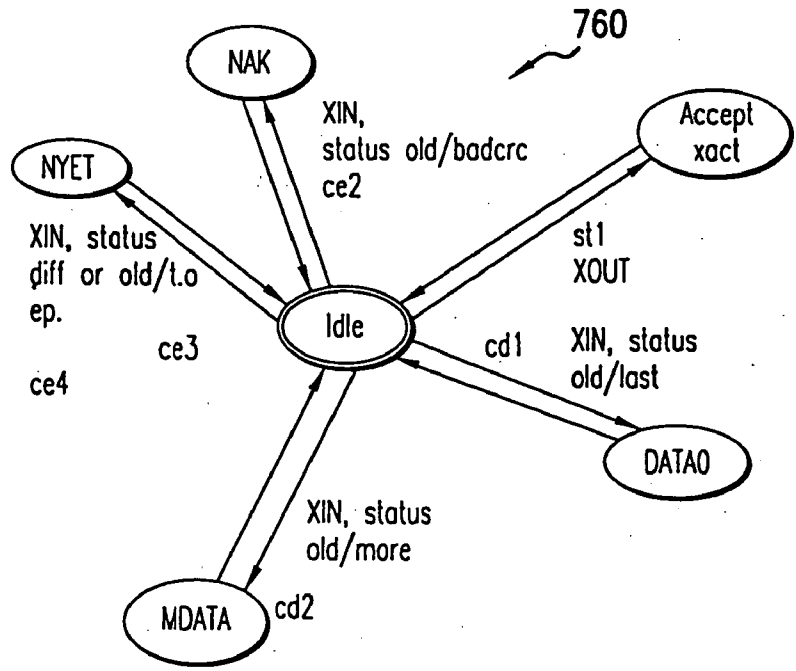


FIG.7B

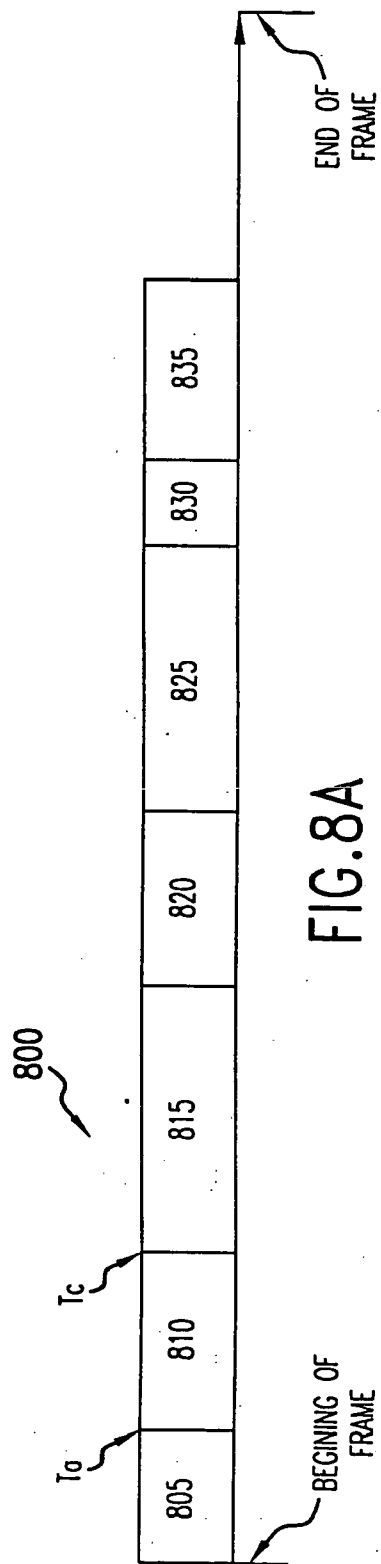


FIG.8A

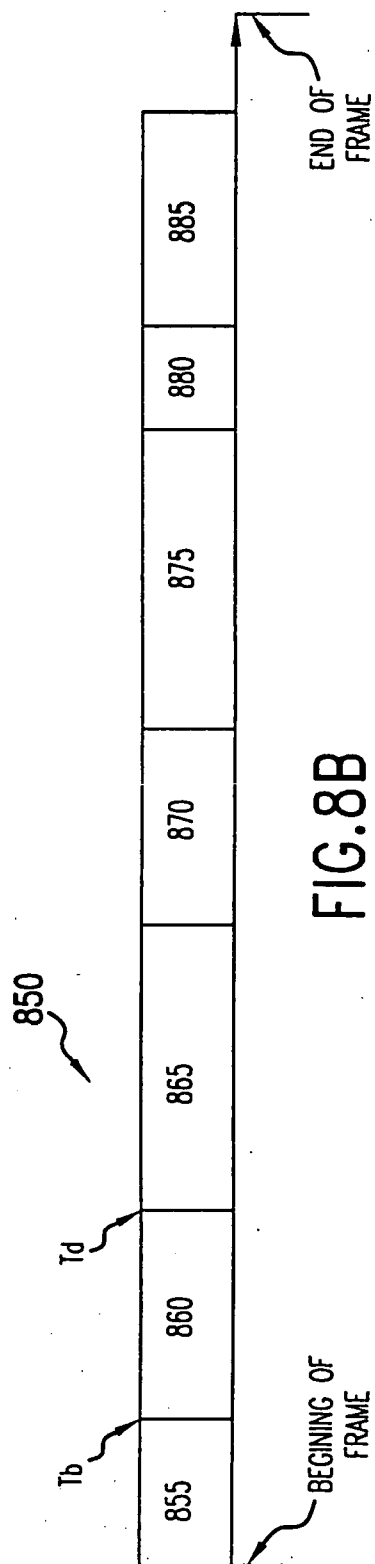


FIG.8B

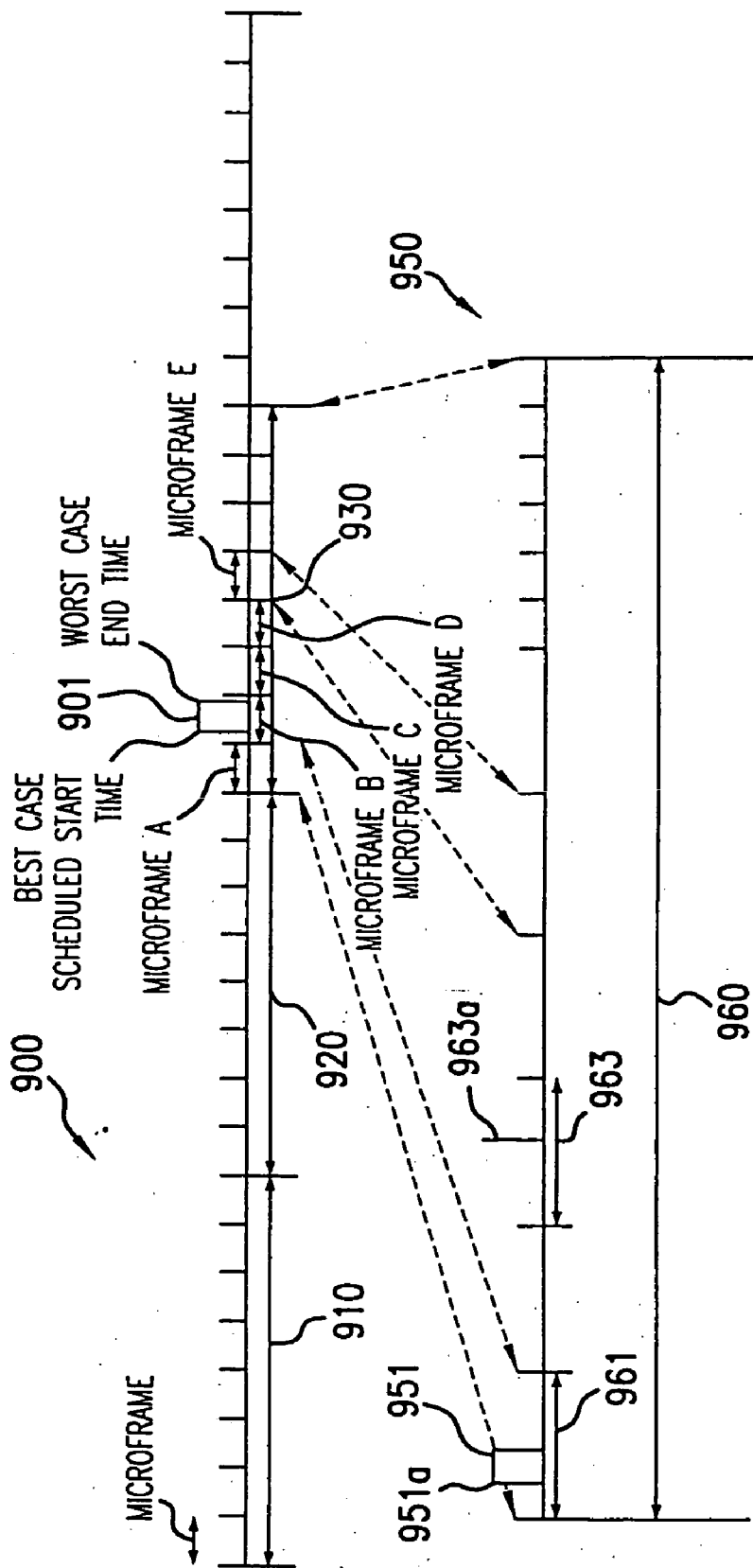


FIG.9

TRANSACTION SCHEDULING FOR A BUS SYSTEM

FIELD OF THE INVENTION

[0001] The present invention pertains to the field of data communication in a digital system. More particularly, the present invention relates to host controllers and hubs used to transfer information on a bus.

BACKGROUND OF THE INVENTION

[0002] A computer or similar device typically has a bus that connects devices to the computing system. Due to a variety of advances (e.g., computing power) in computing systems and devices, the amount of data that can be exchanged between a computing system and its attached devices has increased, requiring a concomitant increase in the bandwidth of the bus. Part of this increased demand for bandwidth has come from multi-media applications that require data to be transferred at regular time intervals (isochronous) either from the device to the computing system (in) via the bus, or in the opposite direction (out). Examples of devices requiring significant bandwidth include cameras, compact disc players, speakers, microphones, video display devices, scanners, and joy-sticks and mice, among other devices.

[0003] The bandwidth available on a bus architecture is partly determined by three factors: transmission medium, topology, and the protocol used to control access to the medium. The protocol and topology partly determine the nature of the relationship between a device and a computing system. One possible relationship is a master-slave relationship. In a master-slave relationship the computing system initiates typically all data transactions between the computing system and a device; i.e., the device only responds to requests from the computing system but never initiates a transaction. A benefit of the master-slave relationship is a bus architecture having relatively low cost and simplicity. The Universal Serial Bus (USB) Specification Revision 1.1, Sep. 23, 1998, is an example of a popular bus architecture having a master-slave relationship between elements connected by the bus. Unfortunately, many of today's devices and computing systems have bandwidth requirements (or data rates) that cannot be supported by existing master-slave bus standards, such as the USB Standard.

[0004] Even though USB does not support relatively high data rates, it has a relatively large base of users, and supports two data rates: 12 Mb/s ("full-speed") and 1.5 Mb/s ("low-speed"). USB allows multiple data rates on a bus because some devices do not require high data rates and can achieve significant cost savings by taking advantage of relatively low-cost, low data-rate drivers and cabling.

[0005] However, the USB protocol that allows the computing system to communicate at low-speed with low data rate devices and alternatively at full-speed with high data rate devices (speed-shifting) results in the amount of data actually transmitted over the bus (effective throughput) being less than that achievable by limiting the bus to full-speed transactions. In other words, speed shifting results in less bandwidth being available for higher speed (e.g., full-speed) devices, especially when there is a relatively large number of low-speed devices attached to the comput-

ing system. The effect of speed shifting on throughput is exacerbated where the ratio of high data rate to low data rate is relatively large.

[0006] Another possible bus protocol would require the host to (1) transmit at a high data rate a packet to a hub, (2) wait for the hub to forward at the low data rate the packet to the agent, (3) wait for the agent to respond at the low data rate to the hub, and (4) receive from the hub at a high data rate the agent's response to the packet. When the ratio of the high data rate to the low data rate is relatively large, this bus protocol may also result in a low effective throughput or bandwidth because of the need to wait for the hub to forward the packet at the low data rate and for the agent to respond at the low data rate.

[0007] Another popular bus technology is defined by "Firewire" or Institute of Electrical and Electronics Engineers (IEEE) Standard 1394 for a High Performance Serial Bus, 1995. IEEE 1394 supports multiple data rates, up to 400 Mb/s. While the aggregate bandwidth is substantially higher than USB, IEEE 1394 employs wasteful speed shifting and is a relatively costly technology.

[0008] The performance of a bus can be significantly affected by speed-shifting, waiting for a hub to perform transactions at a lower data rate than a host data rate, and the ratio of the host's data rate to the agent's data rate. Thus, it is desirable to have a host controller and/or hub that allow communication at the higher data rates required by today's bandwidth intensive systems while allowing backward compatibility with pre-existing solutions, such as USB, and without having to pay the penalties imposed by speed-shifting and the other disadvantages of the prior art.

[0009] One issue faced by high data rate systems communicating with low data rate devices has been described above. Another issue faced by computing systems arises from the multiplicity of bus protocols (or standards) that are available. Typically, a device manufactured to operate in accordance with a bus protocol will not operate in accordance with a different bus protocol. It may be wasteful to require a user to own largely duplicate devices simply because of differences in the protocol. Where there is a large base of devices being used that have a significant economic life, it may be desirable to allow such devices to be used with a computing system that has a host controller and/or an associated hub that provides backward compatibility to the protocol of the legacy devices.

SUMMARY OF THE INVENTION

[0010] According to an embodiment of the invention a method for communicating data using a hub is described. The method includes the step of buffering a single transfer request received at a hub during a transaction between the hub and a host controller, where the single transfer request is to be performed between the hub and an agent to generate a result. The method then includes the step of determining whether a transfer inquiry received at the hub from the host controller corresponds to the result.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] The present invention is illustrated by way of example, and not limitation, in the figures of the accompanying drawings in which like references denote similar elements, and in which:

[0012] FIGS. 1a illustrates a block diagram of a digital system using a protocol in accordance with the present invention;

[0013] FIGS. 1b, 1c & 1d each illustrates a process showing a method in accordance with this invention for communicating between a host controller and a hub;

[0014] FIG. 1e illustrates a hub in accordance with the present invention;

[0015] FIGS. 2a & 2b illustrate state machine diagrams for a host controller and a hub, respectively, performing a transfer in accordance with this invention;

[0016] FIGS. 3a & 3b illustrate state machine diagrams for a host controller and a hub, respectively, performing another transfer in accordance with this invention;

[0017] FIGS. 4a & 4b illustrate state machine diagrams for a host controller and a hub, respectively, performing another transfer in accordance with this invention;

[0018] FIGS. 5a & 5b illustrate state machine diagrams for a host controller and a hub, respectively, performing another transfer in accordance with this invention;

[0019] FIGS. 6a & 6b illustrate state machine diagrams for a host controller and a hub, respectively, performing another transfer in accordance with this invention;

[0020] FIGS. 7a & 7b illustrate state machine diagrams for a host controller and a hub, respectively, performing another transfer in accordance with this invention;

[0021] FIGS. 8a & 8b illustrate best case and worst case frames for data transfers in accordance with the present invention; and

[0022] FIG. 9 illustrate timing diagrams for host controller-hub transactions and hub-agent transactions.

DETAILED DESCRIPTION

[0023] A method and apparatus for communicating between a host and a peripheral (agent) is described, where the agent communicates at a different speed and/or protocol than the host. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be evident, however, to one skilled in the art that the present invention may be practiced in a variety of bus systems, especially serial buses, without these specific details. In other instances well known operations, steps, functions and devices are not shown in order to avoid obscuring the invention.

[0024] Parts of the description will be presented using terminology commonly employed by those skilled in the art to convey the substance of their work to others skilled in the art, such as device or controller drivers, bus or host controllers, hubs, bus agents or agents, and so forth. Also, parts of the description will also be presented in terms of operations performed through the execution of programming instructions or initiating the functionality of some electrical component(s) or circuitry, using terms such as, performing, sending, processing, packaging, scheduling, transmitting, configuring, and so on. As well understood by those skilled in the art, these operations take the form of electrical or

magnetic or optical signals capable of being stored, transferred, combined, and otherwise manipulated through electrical components.

[0025] Various operations will be described as multiple discrete steps performed in turn in a manner that is most helpful in understanding the present invention. However, the order of description should not be construed as to imply that these operations are necessarily performed in the order that they are presented, or even order dependent. Lastly, repeated usage of the phrases “in one embodiment,” “in an embodiment,” “an alternative embodiment,” or “an alternate embodiment” does not necessarily refer to the same embodiment, although it may.

[0026] FIG. 1a illustrates a block diagram of a bus using a protocol in accordance with the present invention. Bus 100 includes a system 102 having a host controller 110 which is coupled to hub 120 which is in turn coupled to agent 130. Host controller 110 has an associated device driver 105 that executes on system 102. Examples of agents include cameras, compact disc players, speakers, microphones, video display devices, scanners, and joy-sticks and mice, among other devices. System 102 can include any digital system capable of digital communication, especially laptop computers, desktop computers, servers, set-top boxes, entertainment systems, and game machines. Consequently, this invention can be practiced with a variety of digital devices using digital communication.

[0027] Two arrows 101a and 101b are drawn in FIG. 1a to provide a frame of reference as to the direction of communication among the host, hub and agent. The direction from the agent to the hub and on to the host is referred to as the upstream direction or upstream (in). The direction from the host to the hub and on to the agent is referred to as the downstream direction or downstream (out).

[0028] Host controller driver 105 facilitates communications or transactions between along bus 100 (e.g., on behalf of an application executing on system 102) by processing packets of information destined for agent 130 and scheduling the packets for transmission by controller 110. Host controller 110 sends data to and receives data from agent 130 data via hub 120. Agent 130 communicates at a different or lower data rate (agent data rate) than the data rate of host controller 110 (host data rate). While only one agent is shown coupled to hub 120, it should be apparent that additional agents (not shown) can be attached to hub 120 or to other hubs (not shown). These additional agents may communicate at the host data rate or the agent data rate. Furthermore, while agent 130 is shown coupled to hub 120, it may be coupled to hub 120 through at least one conventional repeater type hub that operates at the agent data rate. A conventional repeater type hub repeats signals it receives on its upstream side on its downstream ports, and vice versa. The conventional repeater type hub may in turn have one or more agents 130 attached to it.

[0029] Host controller 110 and agent 130 have a master-slave relationship which means that the host initiates typically all data transactions between the host and an agent; i.e., the agent only responds to requests from the host but never initiates a transaction. Hub 120 has store-and-forward buffers (not shown) that allow hub 120 to temporarily store downstream information received from host controller 110

and destined for agent **130**, and to temporarily store upstream information received from agent **130** and destined for host controller **110**.

[0030] Since agent **130** and host controller **110** communicate at different data rates, it is desirable to enhance the effective throughput on the bus by providing a protocol that would allow host controller **110** to both (1) communicate at its higher data rate and (2) not have to wait for responses from agent **130** before engaging in another transaction. The protocol of the present invention allows host controller **110** to take advantage of the store-and-forward characteristic of hub **120** to allow the host controller **110** to communicate at its higher data rate and to engage in another transaction instead of waiting for a response from agent **130**, if a response is required. The protocol of the present invention also provides robustness and reliability to transactions performed between controller **110** and hub **120**. Additionally, the host controller and/or hub of the present invention allow increased effective throughput on the bus and provide increased responsiveness to agents (not shown) that communicate at the host data rate and that are attached to hub **120** or other hubs.

[0031] FIG. 1*b* illustrates a process **150** showing a method in accordance with this invention for communicating with an agent having a lower (or different) data rate, than the data rate of a host controller. The agent may also have a different protocol than the host controller. Process **150** can be used to effect a variety of information transfers between host controller **110** and agent **130**. For ease of understanding process **150** will only be described here with regards to a bulk out transfer. However, process **150** can be used with other information transfers described herein, below. In a bulk out transfer data is transferred from host controller **110** to agent **130** via hub **120**. The bulk out transfer is defined according to an embodiment of this invention as an asynchronous transfer type. However, it should not be concluded from this definition that any bulk and/or out transfer need be asynchronous.

[0032] At step **152** in process **150** a start split transaction is performed. The start split transaction communicates downstream information from host controller **110** to hub **120**. Some of the downstream information communicated to hub **120** is temporarily buffered in hub **120**. The buffers in hub **120** largely behave in a first-in-first-out (FIFO) manner, and are described in greater detail below. Some time after the downstream information is buffered, hub **120** performs a hub-agent transaction (not shown) with agent **130** based on some of the buffered downstream information. The relative timing of the hub-agent transaction need not be described herein because one of ordinary skill in the art would recognize that this is an application or implementation detail for which there are many possibilities. The hub-agent transaction may result in upstream information being buffered in hub **120**. Some time after the downstream information is buffered, at step **154** a complete split transaction is performed. The complete split transaction communicates buffered upstream information from hub **120** to host controller **110**. The relative timing of the complete split transaction need not be described herein because one of ordinary skill in the art would recognize that this is an application or implementation detail for which there are many possibilities.

[0033] A benefit of the split transaction protocol is that it allows controller **110** to initiate communication (start-split

transaction) with agent **130**, engage in another function, or engage in another communication with another agent (low data rate or high data rate agent), and then return to complete the communication that was initiated earlier with the low data rate agent. By communicating using split-transactions, controller **110** communicates at high data rates without speed-shifting and does not sit idle while waiting for hub **120** to communicate with agent **130**. The time that would have been spent being idle can be used to communicate with another agent. In an alternative embodiment in accordance with the present invention, controller **110** may engage in speed-shifting with some agents while engaging in split-transaction communication with other agents.

[0034] The start split and the complete split transactions (split transactions) described above may be used to implement a variety of transfer types (e.g., read or write) for communicating data between controller **110** and agent **130**. In an embodiment of this invention four transfer types (or transfer requests) are defined: bulk out/in, control out/in, interrupt, isochronous. It should be apparent to one of ordinary skill in the art that the spirit and scope of this invention includes other embodiments with fewer, more or different transfer types. Each of the transfer types provides different levels of robustness, reliability, synchronization, asynchronous operation, error detection and correction of the communication flow, and other characteristics that should be apparent to one of ordinary skill in the art. For example, bulk out/in provides large asynchronous data transfers from controller **110** to agent **130** or in the opposite direction. Control out/in also provides asynchronous data transfer from controller **110** to agent **130** or in the opposite direction, but the data is typically control information used to control the operation of elements (e.g., a tape drive) in agent **130** or system **100**. Interrupt provides a periodic data transfer from controller **110** to agent **130** or in the opposite direction. If the transfer is not successful, controller **110** may try again in an embodiment in accordance with this invention. Isochronous transfer provides a data transfer once every predetermined time interval. According to an embodiment of the present invention, the transfer may occur at any time during the time interval. If the transfer is not successful, controller **110** will not repeat the transfer. In an alternative embodiment in accordance with the present invention, the isochronous transfer may provide for repeat transfers.

[0035] The split transactions may include a number of phases depending on the transfer type being implemented. Each of the split transactions may have up to three phases: token, data, and handshake. However, depending on the transfer being performed, some transactions may have fewer phases. In an embodiment of the present invention, bulk and control can use the same phases in each of their respective split transactions. The phases for each of the transfer types described above are shown in Table 1, below. Presence of an "X" in a cell of the table indicates that the split transaction for the transfer type has the phase indicated at the top of the column in which the cell resides. While in this embodiment the token and data phases are separate for each of the transfer types, in alternative embodiments the token and data phases may be combined. It should be apparent that in alternative embodiments transfer types may have fewer, more, or even different phases than those shown in Table 1 without departing from the scope and spirit of the present invention.

TABLE 1

Transfer Type	Start-Split Transaction			Complete-Split Transaction		
	Token	Data	Handshake	Token	Data	Handshake
Bulk-Control Out	X	X	X	X		X
Bulk-Control In	X		X	X	X	X
Interrupt Out	X	X		X		X
Interrupt In	X			X	X	
Isochronous Out	X	X				
Isochronous In	X			X	X	

[0036] FIG. 1c illustrates in greater detail a process 160 showing a start split transaction for a bulk out transfer in accordance with an embodiment of this invention. At step 162 a token packet including hub identification information, agent and endpoint identification information, transfer type, indicator for specifying direction of transfer (in or out), and data rate identification is sent from host controller 110 to hub 120. Hub identification information, and agent and endpoint identification information, and direction are together commonly referred to here as transaction addressing information. The agent identification information identifies the particular agent with which the host is attempting to communicate. The endpoint identification information identifies a particular portion in the agent with which the host is attempting to communicate. Examples of endpoints include: left speaker and right speaker of a speaker hub, or speaker and microphone of telephone handset. The transfer type in the transaction addressing information is not limited to the types described herein (e.g., bulk out, interrupt, isochronous, control), but can include other types known in the art without departing from the spirit and scope of this invention. Data rate identification specifies the data rate with which the hub-agent transaction described in connection with process 150 above will be performed. For an embodiment in which the hub-agent transaction is performed in accordance with the USB standard, data rate identification will specify either 12 Mb/s (full-speed) or 1.5 Mb/s (low-speed). At step 164, a data packet is sent from host controller 110 to hub 120. At step 166, a first acknowledgement is received by host controller 110 from hub 120, if the data packet was decoded properly by hub 120. The first acknowledgement indicates whether the data was decoded properly by hub 120 or whether hub 120 wants to be tried later (e.g., hub 120 had full buffers and was not able to accept the data).

[0037] FIG. 1d illustrates in greater detail a process 170 showing a complete split transaction for a bulk out transfer in accordance with an embodiment of this invention. At step 172, a second token packet including transaction addressing information is sent from the host to the hub. At step 174, a second acknowledgement is received by host controller 110 from hub 120, where the second acknowledgement can either (1) include handshake information received by hub 120 from agent 130 during the hub-agent transaction described above in connection with FIG. 1b or (2) indicate that hub 120 does not yet have information based on the hub-agent transaction to forward to host controller 110 (e.g., the hub-agent transaction has not yet been completed). The handshake information indicates whether (1) agent 130 properly received data during the hub-agent transaction (ACK), (2) agent 130 indicated that it is not able to operate

normally (STALL), or (3) agent 130 indicated that it wanted to be tried later (NAK). While the first and second acknowledgements and the handshake information have been described as specifying certain indicators, it should be apparent to one of ordinary skill in the art that these acknowledgements and handshakes and other ones described herein may represent other indications. Additionally, acknowledgements and handshakes different from or additional to the ones described herein may be added in an alternative embodiment without departing from the spirit and scope of the invention.

[0038] While the above description has generally been presented in the context of agent 130 and hub 120 communicating at a lower data rate than the data rate between hub 120 and host controller 110, those skilled in the art will appreciate that the present invention may be practiced to bridge a lower data rate to a higher data rate instead, or even equal data rates but different protocols.

[0039] While in FIG. 1 only one hub was shown in between the agent and the host there can be multiple hubs between any particular agent and the host. While only six transfer types have been described, those skilled in the art will appreciate that other types can be used without departing from the scope or spirit of this invention.

[0040] Each of FIGS. 2a, 2b, 3a, 3b, 4a, 4b, 5a, 5b, 6a, 6b, 7a & 7b illustrates a state machine diagram for performing a transfer using a host controller and a hub in accordance with this invention. Figures with an "a" as a suffix show the state machine diagram for a host controller; the state machine may be performed on host controller 110 described above in connection with FIG. 1a. Figures with a "b" as a suffix show the state machine diagram for a hub; the state machine may be performed on hub 120 described above in connection with FIG. 1a. The state machines illustrated in these figures show processes having multiple steps. It should be apparent that some of the steps may be divided into more numerous steps or combined into fewer steps without departing from the scope and spirit of this invention. The state machines are not described in great detail because their operation should be apparent to one of ordinary skill in the art. For ease of understanding, the processes performed by the state machines are not described separately. Since the processes performed by the state machines operate in tandem (i.e., each process has steps whose execution depends on the occurrence of events or steps in the other process), the description of the processes are interleaved to facilitate understanding.

[0041] FIGS. 2a & 2b illustrate state machine diagrams for a host controller and a hub, respectively, performing a transfer in accordance with this invention, specifically a split out bulk/control transfer. Process 200 and process 260 show the state machine for a host controller and a hub, respectively. Process 200 includes start split transaction, having a token phase (XOUT) and a data phase (DATAx), which may be repeated up to three times by the host controller when timeouts occur during the phases of a transaction attempt between the host controller and the hub. In response to the start split transaction, process 260 will either propagate through states that will accept the data (ACK), respond to a host controller retry after a communication failure of a hub handshake to the host controller (ACK, don't accept data), request a host controller retry due to lack of space to hold the

start transaction information (NAK), or timeout up to three times. Process 200 shows the host controller response to a complete split transaction (XIN) when the transaction between the hub and agent was successfully processed (Advance), resulted in a NAK from the agent (NAK), received an indication that the agent was not able to function normally (STALL), was not yet completed by the hub or agent (NYET), or the XIN or its response had some communication failure and resulted in up to three timeouts between the host controller and hub. In response to the complete split transaction, process 260 will either indicate that the transaction between the hub and the agent has not finished (NYET) or will provide an indication (ACK, NAK, STALL) of what transpired during the transaction between the hub and the agent.

[0042] FIGS. 3a & 3b illustrate state machine diagrams for a host controller and a hub, respectively, performing another transfer in accordance with this invention, specifically a split out interrupt transfer. Process 300 and process 360 shows the state machine for a host controller and a hub, respectively. Process 300 includes a start split transaction, having a token phase (XOUT) and a data phase (DATAx), which is not repeated by the host controller because according to an embodiment the interrupt out transfer is time sensitive and need not be repeated if it is not successful on the first try. In response to the start split transaction, process 360 will either accept the data (ACK), or do nothing. Process 300 shows the host response to a complete split transaction (XIN) when the transaction between the hub and agent was successfully processed (Advance), received an indication that the agent was not able to function normally (STALL), was not yet completed by the hub or agent (NYET), or the XIN or its response had some communication failure and resulted in up to three timeouts between the host controller and hub. In response to the complete split transaction, process 360 will either indicate that the transaction between the hub and the agent has not finished (NYET) or will provide an indication (ACK, NAK, STALL, NYET) of what transpired during the transaction between the hub and the agent.

[0043] FIGS. 4a & 4b illustrate state machine diagrams for a host controller and a hub, respectively, performing another transfer in accordance with this invention, specifically a split out isochronous transfer. Process 400 and process 460 shows the state machine for a host controller and a hub, respectively. Process 400 includes a start split transaction, having a token phase (XOUT) and a data phase, neither of which is repeated. Process 400 does not include a complete split transaction according to an embodiment. Process 460 allows the hub to agent transaction data to be subdivided into multiple sections to minimize buffering required in the hub; the host controller can send the next section of data just before the hub needs to send it to the agent. In this case each split start transaction is marked with ALL, BEGIN, MIDDLE or END so that the hub can detect when a communication failure has caused it to not receive a data section in the correct sequence. In response to the start split transaction, process 460 will accumulate a data payload having one data section (ALL), two data sections (BEGIN, END), or three or more data sections (BEGIN, MIDDLE . . . MIDDLE, END).

[0044] FIGS. 5a & 5b illustrate state machine diagrams for a host controller and a hub, respectively, performing

another transfer in accordance with this invention, specifically a split in bulk/control transfer. Process 500 and process 560 shows the state machine for a host controller and a hub, respectively. Process 500 includes start split transaction, having a token phase (XOUT), which is repeated up to three times by the host controller when timeouts occur during transaction attempts between the host controller and the hub. In response to the start split transaction, process 560 will either acknowledge and accept the transaction (ACK, accept xact), respond to a host controller retry after a communication failure of a hub handshake to the host controller (ACK, ignore xact), or request a host controller retry due to a lack of space to hold the start transaction information. Process 500 shows the host controller response to a complete split transaction (XIN) when the transaction between the hub and agent was successfully processed (Advance), received an indication that the endpoint is unable to function normally (STALL), was not yet completed by the hub or agent (NYET), ignores data received from the hub because it is corrupted, or the XIN or its response had some communication failure and resulted in up to three timeouts between the host controller and hub. In response to the complete split transaction, process 560 will either indicate that the transaction between the hub and the agent has not finished (NYET) or will provide an indication (NAK, STALL) of what transpired during the transaction between the hub and the endpoint, or send the data received from the agent by the hub to the host controller.

[0045] FIGS. 6a & 6b illustrate state machine diagrams for a host controller and a hub, respectively, performing another transfer in accordance with this invention, specifically a split in interrupt transfer. Process 600 and process 660 shows the state machine for a host controller and a hub, respectively. Process 600 includes a start split transaction, having a token phase (XOUT). In response to the start split transaction, process 660 will accept the transaction (accept xact). Process 600 shows the host response to a complete split transaction (XIN) when the transaction between the hub and agent was successfully processed (Advance), received an indication that the endpoint is not able to function normally (STALL), was not yet completed by the hub or agent (NYET), received a NAK from the agent (NAK), retries token phase if data received is an agent retry of the previous transaction request (ignore data), or the XIN or its response had some communication failure and resulted in up to three timeouts before the host controller gives up communicating with the agent. In response to the complete split transaction, process 660 will indicate that a timeout occurred when the hub was communicating with the agent (NYET), or the hub didn't received the start transaction for this request and had no corresponding response information (STALL) or provide an indication (NAK, STALL) of what transpired during the transaction between the hub and the agent, or send the data to the host controller.

[0046] FIGS. 7a & 7b illustrate state machine diagrams for a host controller and a hub, respectively, performing another transfer in accordance with this invention, specifically a split in isochronous transfer. Process 700 and process 760 shows the state machine for a host controller and a hub, respectively. Process 700 includes a start split transaction, having a token phase (XOUT). In response to the start split transaction, process 760 will accept the transaction (accept xact). Process 700 shows the host response to a complete split transaction (XIN) when the transaction between the hub

and agent was successfully processed and all the data has been returned (TAadvance) or there is more data to return (DAadvance) or an error occurred (record error) due to a communication failure between the host and hub (timeout or bad cyclic redundancy check) or the agent (NAK) or a hub problem with the agent (NYET). In response to the complete split transaction (XIN), process 660 will indicate that data received from the agent had a bad cyclic redundancy check (NAK) or the agent didn't respond (NYET) or the hub had no information about this complete-split, or send the data to the host controller either indicating all the data has been returned (DATAO) or more data is to be returned (MDATA).

[0047] It is useful at this point to summarize the description of the above protocol before describing the remaining apparatus and methods of the present invention. The protocol described above allows a host controller to transfer data to or receive data from an agent via a hub. The protocol allows the host controller to engage in a first transaction (start split transaction) in which a transfer request is communicated to the hub. After the host controller performs the first transaction, it may engage in an intermediate transaction with the same hub or another hub without waiting for the hub and agent to perform the transfer request (i.e., engage in the transfer of data to or from the agent). The intermediate transaction may include a transfer request for the agent, another agent on the same hub as the agent, or another agent on yet another hub. After the hub engages in the transfer of data to or from the agent, the host controller performs a complete split transaction (or second transaction) to get the result (e.g., data or handshake sent from the agent to the hub) of the transfer performed between the hub and the agent. By allowing the host controller the capacity to engage in an intermediate transaction, instead of waiting for the hub to perform the transfer request (or third transaction) with the agent, the effective throughput of a bus using a protocol in accordance with this invention can be significantly greater than buses which involve speed-shifting or which require the host controller to wait for the hub to perform the transfer with the agent before initiating another transfer.

[0048] While the above protocol defines the sequence of transactions involved in communicating data across a bus, it does not explicitly describe the timing for the transactions or transfers which will result in data being sent to or received from an agent. However, timing of transfers for agents is important because agents typically require data to be sent to or received from the host controller on a periodic (e.g., isochronous or interrupt) or asynchronous (e.g., bulk or control) basis. Additionally, while the above protocol allows a host controller to perform an intermediate transaction (or even multiple intermediate transactions) between the start split and the complete split transaction, the above protocol does not explicitly describe how transfer requests are stored in the hub and how the hub and agent perform transfer requests without requiring the host controller to wait for the transfer request to be performed before engaging in an intermediate transaction. The issues not addressed by the above description of the protocol, namely the timing of transfer requests and the processing (i.e., buffering and performance) of transfer requests by a hub, are addressed by the following descriptions of methods and apparatus in accordance with the present invention. The present invention includes a method and apparatus for scheduling transfers of data to and from an agent and a method and apparatus

for processing transfer requests at a hub. The method and apparatus for scheduling transfers of data is described first and then the method and apparatus for buffering and performing transfer requests is described second.

[0049] Referring again to FIG. 1, the process of scheduling transfers of data first begins when system 102 performs configuration for agents attached to the bus of the system. Configuration can occur upon system initialization or boot-up or when an agent is attached to the bus after initialization. The scheduling of data transfers depends on the transfer type associated with an agent (or an endpoint of an agent) and the amount of data involved in the transfer. The manner in which agents having associated periodic transfers, such as isochronous and interrupt, are handled is described first, below, and then the manner in which agents having associated asynchronous transfers, such as bulk and control, are handled is described next.

[0050] During the process of configuration, each endpoint of each agent informs the system of the endpoint's associated maximum data payload size and transfer type (e.g., isochronous in/out, interrupt, bulk, control). The manner in and device by which each agent informs the system is well understood by those of ordinary skill in the art and need not be described here. The maximum data payload size is the largest amount of data that a transfer to or from an agent will entail. The system relays the data payload size and the transfer type to the host controller. The manner in which the system relays the size and transfer type to the host controller is well understood by those of ordinary skill in the art and need not be described here. The host controller uses the aforementioned two pieces of information associated with each endpoint to generate a budget list for the periodic transfers of the endpoints. In alternate embodiment, a software driver such as the host controller driver or even a hardware driver can generate the budget list and perform the scheduling operations described below. The budget list gives the earliest time that a transfer (sending or receiving a data packet) may occur as well as the latest time that a result associated with the transfer may be available. The earliest time that a transfer can occur depends on the amount of time taken up by each of the previous transfers assuming that the previous transfers happened under the best of circumstances (defined below). The latest time that a result associated with a transfer may be available depends upon the amount of time taken up by each of the previous transfers assuming that previous transfers happened under the worst circumstances (defined below) and the time required for the transfer under the worst circumstances. The earliest time that a transfer can occur is important because the host controller needs to have the transfer request buffered at the hub before that time so that as soon as the hub finishes the transfer request that is ahead of the buffered request the hub can turn its attention to the buffered transfer request. The latest time that a result associated with the transfer may be available is important because after the latest time it is substantially certain that the result will be available for the host controller to retrieve. If the host controller were to try to retrieve the result from the hub before the latest time, meaning that the result is not available yet, a less efficient protocol employing multiple retrievals would be required.

[0051] Transfers happen under the best of circumstances when each transfer involves the maximum data payload and there is substantially no bit-stuffing. Transfers happen under

the worst of circumstances when each transfer involves maximum data payload and there is maximum bit-stuffing. Bit-stuffing occurs according to an embodiment of the present invention because the signals on the bus obey non-return-to-zero (NRZ) signaling. According to an embodiment of the present invention, bit-stuffing may increase the size of the maximum data payload by 16%. While in an embodiment, the best circumstances and the worst circumstances have been defined in terms of bit-stuffing, it should be apparent that in alternative embodiments the circumstances can be described in terms of other things that can expand or decrease the size (or time) of transfers, or generate delays.

[0052] The generation of a budget list in accordance with the present invention will now be described. While one way of generating a budget list is described herein, it should be apparent that the spirit and scope of the present invention encompasses other possible ways. A budget list is a list of the allowable periodic transactions that can occur in a particular frame template. A frame is a continuously occurring period of time defined as part of the bus specification that is sufficient to provide for one or more transactions. In an embodiment, a 1 millisecond time period is defined for a frame. A different budget list is constructed for each frame that has a different set of allowable periodic transactions. A frame template is a description of a particular periodically repeating frame that provides for some maximum number of transactions, each transaction having some maximum data payload. A frame contains some number of transactions of some actual data payload, while a frame template describes a potential budgeted frame. Each budget list has an associated best case information and an associated worst case information. The best case information describes the situation in which each transaction in the frame template occurs under the best circumstances. In an alternative embodiment, transfers rather than transactions are represented by the best case information and worst case information.

[0053] FIG. 8a illustrates a diagram for a best case frame template 800 in accordance with the present invention. Block 805 represents the transfer associated with a first endpoint, where the transfer happens under the best case. Block 810 represents the transfer associated with a second endpoint, where the transfer happens under the best case. The remaining blocks 815-835 represent similar best case transfers for other endpoints configured by the system. The worst case information describes the situation in which each transfer in the frame template occurs under the worst circumstances. FIG. 8b illustrates a diagram for a worst case frame template 850 in accordance with the present invention. Block 855 represents the transfer associated with the first endpoint, where the transfer happens under the worst case. Block 860 represents the transfer associated with the second endpoint, where the transfer happens under the worst case. The remaining blocks 865-885 represent similar worst case transfers for other endpoints configured by the system. It should be apparent that the relative sizes of the blocks are only meant for purposes of illustration.

[0054] The significance of best case frame template 800 and the worst case frame template 850 will now be explained by examining blocks 805, 810, 855 and 860. Blocks 805 and blocks 855 represent the best case and worst case transfer, respectively, for the first endpoint. The earliest that the transfer for the first endpoint can finish is T_a . The latest that

the transfer for the first endpoint can finish is T_b . The foregoing suggests that the transfer associated with the second endpoint can begin as early as time T_a or begin as late as time T_b . Blocks 810 and 860 represent the best case and worst case transfer times, respectively, for the second endpoint. The earliest that the transfer for the second endpoint can end is T_c . The latest that the transfer for the second endpoint can finish is T_d . The foregoing suggests that the information necessary for the second transaction must be available to the hub before time T_a and any result due to the first transfer is substantially certain to be available after time T_b . Furthermore, it should be apparent that scheduling of a transfer depends on the time required to perform each of the previous transfer requests. Additionally, it should be apparent that the scheduling of the complete-split transaction will depend on the time to perform the present transfer request, if any, associated with a start-split transaction.

[0055] Frame template 800 and frame template 850 are frame templates that are representative of a frame between the hub and the agent (hub-agent or classic frame). According to an embodiment of the present invention, the host controller and the hub have another frame that is a fraction of the size (microframe) of the classic frame. More specifically, in an embodiment, eight microframes are equivalent to a single classic frame. Since the host controller and the hub communicate using microframes and since a transfer request before reaching the hub starts as a start split transaction, it is useful to describe the time for performing the start split transaction in terms of microframes. Similarly it is useful to describe the time for performing the complete split transaction in terms of microframes. With regards to the start time for a start split transaction, according to an embodiment, a start split transaction should occur one microframe before the microframe in which the transfer between the hub and agent can occur according to the best case frame. With regards to the start time for a complete split transaction, according to an embodiment, a complete split transaction should occur one microframe after the microframe in which the transfer of the associated start split transaction can finish according to the worst case frame. According to an embodiment of the present invention, the host controller uses the best case frame and the worst case frame for each frame template to create a start-split vector and a complete-split vector for each endpoint in each frame template. The start-split vector contains a value indicative of the microframe in which the start split transaction for an endpoint should occur during a classic frame. According to an embodiment, start-split vector has values ranging from -1 to 6. The complete-split vector contains a value indicative of the microframe in which the complete split transaction for an endpoint should occur during a classic frame. According to an embodiment, the complete-split vector has values ranging from 1 to 8.

[0056] When host controller 110 is tasked by host controller driver 105 to perform a transfer in a particular frame for a particular endpoint, host controller 110 examines the best case vector that is associated with the endpoint and the frame template that corresponds to the particular frame. The value in the start-split vector for a given endpoint and frame template indicates the microframe at which a start split transaction is to be performed. Similarly, host controller 110 examines the complete-split case vector to determine the microframe at which a complete split transaction is to be performed.

[0057] FIG. 9 includes a sequence of hub-agent frames (or classic frames) 900 and a host controller-hub frame 950. Frames 900 include frames 910-930. Frame 930 has a transfer 901 which is scheduled to start in microframe B and is scheduled to end in microframe B. In this illustration, transfer 901 is an isochronous out transfer, but it should be apparent that other transfers can be given a similar representation. Frame 950 includes frame 960 which has sub-frames 961 and 963. While transfer 901 is scheduled to occur in hub 120 in microframe A and end in microframe B (i.e., in the hub-agent time frame represented by frames 900), the associated start-split transaction occurs in microframe A and the associated complete-split transaction occurs in microframe C. In the host controller-hub time frame represented by frame 950, the associated start-split transaction occurs at time 951 a sometime in sub-frame 961 and the associated complete-split transaction occurs at time 963a sometime in sub-frame 963. Transfer 951 represents transfer 901 in the host controller-hub time frame. Other transfers and transactions can also occur in sub-frames 961 and 963. It should be apparent from FIG. 9, that sub-frames 962-3 are available for other transfers and transactions with other agents.

[0058] While the above description has generally been presented in the context of periodic transactions, the invention is not limited to periodic transaction but also includes asynchronous transfers such as the bulk and control transfers described above. According to an embodiment of the present invention, ninety percent of a classic frame is set aside for periodic transfers. Asynchronous transfers will occur at the earliest time there is space available for them on the bus between the hub and the agent. The hub will accept a transaction during a microframe when it has space available to hold the start-split transaction until it can be issued on the classic bus (i.e., between the hub and an agent). Then the hub will issue the transaction during a classic frame when it has no other pending periodic transactions. Once the hub has accepted a start-split, the host controller will later issue a complete-split transaction to retrieve the results from the hub for that transaction on the classic bus. The host controller driver issues an asynchronous split-transaction to the hub whenever it has no other periodic transactions to issue during a microframe. The buffering provided by the hub for bulk/control transactions is distinct and separate from the buffering provided by the hub for periodic transactions.

[0059] A method and apparatus for buffering periodic transfer requests and processing them in accordance with this invention will now be described. FIG. 1e shows in greater detail hub 120 of FIG. 1a. According to an embodiment, hub 120 of the present invention includes a host-controller-hub (or high-speed hub) controller 180, a hub-agent (or low-speed) hub controller 181, a memory 182, a control unit 183, a clock 183a, a repeater 184, a routing logic 185, and ports 185-189. The controller 180 performs split transactions between hub 120 and host controller 110. Whenever a start-split transaction is received by controller 180 from the host controller 110, controller 180 records the current microframe number in memory 182 as a timestamp for the transaction. An alternate embodiment would record the microframe number as a timestamp in the state records to separate one group of transactions received by the hub during one microframe from those transactions received during another microframe. Controller 181 performs transfers between the hub and one or more agents. The transfers

performed by controller 181 are received during start split transactions performed between controller 180 and host controller 110. Memory 182 is coupled to both high-speed hub controller 180 as well as low-speed hub controller 181. Memory includes a pipeline with a plurality of stages. According to an embodiment, the pipeline has five stages. Each stage of the pipeline corresponds to a microframe as defined above. Each stage of the pipeline has a plurality of transaction states (or transaction records). According to an embodiment, a stage of the pipeline has 19 or fewer transaction states or records. Each stage stores records representative of transfers to be performed. A record may have several fields, such as device and endpoint address, direction of transfer, transfer type, status, and data reference. Status indicates whether the transfer is not ready (prevent the low-speed hub controller from performing it), pending (waiting for execution) or ready (has been performed by the low-speed hub controller). The data reference is a pointer to a starting address in memory for data received (i.e., in transfer) from the agent or data to be sent to the agent (e.g., out transfer).

[0060] Hub 120 includes control unit 183 and clock 183a coupled to control unit 183. According to an embodiment, clock 183a generates a microframe indication; i.e., that one-eighth of the time-duration of the classic frame has elapsed. Control unit 183 is coupled to the memory and monitors the records in the stages and prevents the performance of a transfer by the low-speed hub controller 181 if the time at which the transfer is to begin execution at the low-speed hub controller 181 is past the time the transfer was scheduled to be performed or to have completed performance. Specifically, according to an embodiment, a record whose associated time stamp indicates that it was received by controller 180 from host controller 110 one microframe earlier, is set to a pending status to allow the low-speed hub controller 181 to issue the transaction on the classic bus. A record whose associated timestamp indicates that it was received more than three microframes before the current microframe indication but has not yet been performed by low speed hub controller 181 is marked as old to prepare it for a subsequent corresponding complete-split transaction by the high speed hub controller 180. A record whose associated timestamp indicates that it was received more than four microframes before the current microframe indication but has not yet been performed or is currently being performed by low-speed hub controller 181 is aborted and removed from the pipeline. Clock 183a is also coupled to the low-speed hub controller 181. Controller 181 sequences in order through the periodic transaction records that are marked as pending. According to an embodiment, controller sequences 181 through the pending records received in the earliest microframe before proceeding to the records received in the next earliest received microframe and so on. According to an embodiment, controller 181 proceeds to the next earliest received microframe when control unit 181 receives a microframe indication. When control unit 181 receives a microframe indication it changes the status of the transfers in the next earliest received microframe from not ready to pending, and flushes out the transfers that are stale as described above. To perform a transfer controller 181 transfers data to routing logic 185 which gives controller 181 access to one of the ports 186-189. Routing logic 185 gives an agent access to either repeater 184 or controller 181 to allow the transfer of data

at a high data rate or a low data rate depending on the agent's configured data rate. Repeater **184** is coupled to controller **180** and routing logic **185**. Repeater **184** repeats signals received from (or destined to) controller **180** and destined to (or received from) a high speed agent coupled to one of ports **186-189**.

[0061] While sequencing through the pending records received in the earliest microframe, controller **181** operates in accordance with the following rules. First, after performing a transfer between controller **181** and the agent, controller **181** will retrieve the next transaction record and perform the next pending isochronous or interrupt transfer. Second, if there is no pending isochronous or interrupt transfers, controller **181** will perform a pending bulk/control transfer. Third, if there is no pending bulk/control transfer, controller **181** waits for the high-speed hub controller **180** to indicate a pending bulk/control or isochronous or interrupt transfer.

[0062] When the low-speed hub controller performs a transfer a result is stored in the memory **182**. The result may be either a handshake or data received from the agent. Transfers which have been performed have their status changed by the low-speed hub controller **181** to ready and their records are made available to the high-speed hub controller **180**. When a complete split transaction is received by the high-speed hub controller **180**, the high-speed hub controller **180** examines the record of the most recently performed transfer with the same addressing information and sends a response back to the host controller **110** based on the result of the most recently performed transfer. If the complete split transaction is not inquiring about a transaction with the same addressing information, the high-speed hub controller **180** will respond back with a NYET.

[0063] Thus, a method and apparatus for scheduling transfers between a host controller and a hub has been described. Additionally, a method and apparatus for buffering and performing transfers at a hub has been described. Although the present invention has been described with reference to specific exemplary embodiments, it will be evident to one of ordinary skill in the art that various modifications and changes may be made to these embodiments without departing from the broader spirit and scope the invention as set forth in the claims. Accordingly, the specification and drawings are to be regarded in an illustrative rather than a restrictive sense.

1. A method for communicating data using a hub, the method comprising:

buffering a single transfer request received at a hub during a transaction between the hub and a host controller;

wherein the single transfer request is to be performed between the hub and an agent to generate a result; and

determining whether a transfer inquiry received at the hub from the host controller corresponds to the result.

2. The method of claim 1, wherein the single transfer request is received at a first data rate or a first protocol.

3. The method of claim 1, wherein the single transfer request is performed at a second data rate or a second protocol.

4. The method of claim 3, wherein

the hub includes storage for buffering transfer requests, and

at any instant in time transfer requests buffered in the storage take up an aggregate storage space that is less than an information capacity of three communication frames between the hub and the agent.

5. The method of claim 3, wherein

at any instant in time transfer requests buffered in the storage take up an aggregate storage space that is less than an information capacity of a communication frame between the hub and the agent.

6. The method of claim 1, wherein the single transfer request is associated with a time stamp and the step of performing includes,

comparing the time stamp with a time indication, and

performing the single transfer request when the time stamp is within a predetermined time range relative to the time indication.

7. The method of claim 1, wherein the single transfer request is associated with a time stamp and the step of performing includes,

comparing the time stamp with a time indication, and

preventing the host controller from performing the single transfer request when the time stamp is outside a predetermined time range relative to the time indication.

8. The method of claim 1, wherein the step of buffering includes,

storing a transaction record corresponding to the single transfer request.

9. The method of claim 6, wherein the step of buffering includes,

setting a status field for the single transfer request to pending.

10. The method of claim 7, wherein the step of buffering includes,

defining a data reference field if the single transfer request requires data to be received at the hub from the agent or sent to the agent.

11. The method of claim 1, further comprising:

sending from the hub to the host controller a response based on the result when the result corresponds to information received in the transfer inquiry.

12. The method of claim 1, wherein the step of determining includes,

comparing a transaction record corresponding to the single transfer request with information received in the transfer inquiry.

13. An input/output device for communicating data, comprising:

a memory adapted to buffer a single transfer request received during a transaction between a host controller and the device;

a hub controller that performs the single transfer request between the device and an agent to generate a result;

wherein the device receives a transfer inquiry from the host controller; and

a control unit that determines whether the transfer inquiry corresponds to the result.

14. The device of claim 13, wherein the single transfer request is received at a first communication speed or a first protocol.

15. The device of claim 13, wherein the single transfer request is performed at a second communication speed or a second protocol.

16. The device of claim 13, wherein

at any instant in time transfer requests buffered in the memory take up an aggregate storage space that is less than the information capacity of a communication frame between the hub controller and the agent.

17. The device of claim 16, wherein the aggregate storage space is less than three-fourths of the information capacity of the communication frame between the hub controller and the agent.

18. The device of claim 13, wherein the memory has a plurality of stages including a first stage storing a first transfer request and an adjacent second transfer request, an interval during which transfer requests are not received at the hub separates receipt of the first transfer request and the

second transfer request, and the hub controller performs the first transfer request and the second transfer request.

19. The device of claim 18, wherein performance of the first transfer request by the hub controller and performance of the second transfer request by the hub controller is divided by a delay less than the interval.

20. The device of claim 16, wherein the memory stores a transaction record corresponding to the single transfer request.

21. The device of claim 16, wherein the memory stores a status field corresponding to the single transfer request and the status field is indicative of a pending status.

22. The device of claim 16, wherein the memory stores a transaction record at an address indicative of a pending status.

23. The device of claim 16, wherein the memory stores a data reference when the single transfer request requires data to be received at the hub from the agent or sent to the agent.

24. The device of claim 16, wherein the hub controller sends from the device to the host a response based on the result.

23-46. (Cancelled)

* * * * *