BUS CONTROLLER WITH CYCLE TERMINATION MONITOR
EINE BUSSTEUERUNG MIT ZYKLUSBEENDIGUNGSÜBERWACHUNGSSCHALTUNG
CONTROLEUR DE BUS AVEC MONITEUR DE FIN DE CYCLE

Designated Contracting States:
DE GB IE

Priority: 15.06.1998 US 97498

Date of publication of application:
28.03.2001 Bulletin 2001/13

Proprietor: SUN MICROSYSTEMS, INC.
Palo Alto, California 94303 (US)

Inventor: GARNETT, Paul, J.
Merseyside WA12 9PW (GB)

Representative: Harris, Ian Richard
D. Young & Co., 21 New Fetter Lane
London EC4A 1DA (GB)

References cited:
This invention relates to a bus controller for a computer system such as a multi-processor system, for example, in which first and second processing sets (each of which may comprise one or more processors) communicate with an I/O device bus via a bridge.

The application finds particular but not exclusive application to fault tolerant computer systems where two or more processor sets need to communicate in lockstep with an I/O device bus via a bridge.

In such a fault tolerant computer system, an aim is not only to be able to identify faults, but also to provide a structure which is able to provide a high degree of system availability and system resilience to internal or external disturbances. In order to provide high levels of system resilience to internal disturbances, such as a processor failure or a bridge failure for example, it would be desirable for such systems automatically to control access to and from a device that might appear to be causing problems.

Automatic access control provides significant technical challenges in that the system has not only to monitor the devices in question to detect errors, but also to provide an environment where the system as a whole can continue to operate despite a failure of one or more of the system components. In addition, the controller must also deal with any outstanding requests issued by components of the computer system and this is typically problematic as bus protocols, such as PCI for example, typically don’t support error termination at all stages of operation.

International patent application WO96/17303 describes a computer system having an ISA bus and a PCI bus that is provided with a PCI to ISA bridge having certain imbedded functions performed by PCI slaves on the bridge. In order to implement the bridge in slow CMOS technology, the PCI control signals are latched on the bridge. Since the PCI slaves on the bridge cannot respond with control signals on the PCI bus fast enough to satisfy the PCI bus protocol due to this latching, a logic device is provided on the bridge. The logic device monitors the unlatched master-slave control signals carried on the PCI bus, and in appropriate situations, drives the control signals on the PCI bus (within the time specified by the PCI bus protocol) that the PCI slaves would normally drive but are unable to within the time necessary to meet the PCI bus protocol.

Accordingly, an aim of the present invention is to address these technical problems.

Particular and preferred aspects of the invention are set out in the accompanying independent and dependent claims.

In accordance with one aspect of the invention, there is provided a bus control mechanism for a computer system that includes a bus, a first component and a second component, wherein the first and second components are interconnected via the bus for performing a data transfer operation, the data transfer operation being initiated by an exchange of request and response signals between the first and second components, and a component that issued a request signal is operable to effect data transfer on receipt of a response signal, the bus control mechanism comprising: a switch selectively operable to disable the bus; a fake response generator selectively operable to generate a fake response signal; and a controller operable to monitor the request and response signals exchanged between the components and, in situations where a corresponding response signal is not issued within a predetermined time following a particular request signal, to cause the switch to disable the bus and to cause the fake response generator to issue a fake response signal to the component that issued the particular request signal for terminating the data transfer operation.

Thus, embodiments of the present invention can provide a bus controller which monitors request and response signals on the bus. Where a response signal is not provided within a predetermined period following a request signal, the controller causes a switch to disable the bus an causes a fake response signal to be supplied to the initiator of the request. As a result, locking up of the system may be prevented. In one particular embodiment, the switch isolates first and second parts of the bus, so that data signals cannot reach the target component.

The component that issued the particular request signal can be operable wherein the component that issued the particular request signal is operable, on receipt of the fake response signal, to transfer data to the bus, which data is thus discarded as a result of the disabling of the bus by the switch. The switch may be a FET.

The computer system can include a clock signal generator, and the terminator includes a counter for counting clock signals between detection of a particular request signal and detection of a corresponding response signal, the request being terminated if a response has not been detected within a predetermined number of clock cycles.

The first component may be a processing set comprising one or more processors, and the second component may be a bridge for directing signals from the processing set to one or more system resources and from the one or more resources to the processing set.
The bus may be a PCI bus. In one embodiment, it may be assumed that one of the first and second components is a functioning component and the other of the components is a malfunctioning component, the terminator being arranged to terminate a request from the functioning component only.

Alternatively, the terminator may be arranged to terminate requests from either component. The request signal may be an IRDY# signal and the response signal may be a TRDY# signal. In accordance with another aspect of the invention, there is provided a computer system comprising: a bus, a first component and a second component interconnected via the bus for performing a data transfer operation, the data transfer operation being initiated by an exchange of request and response signals, wherein a component that initiates a request signal is operable to effect data transfer upon receipt of a response signal; and a bus control mechanism that comprises: a switch selectively operable to disable the bus; and a fake response generator selectively operable to generate a fake response signal; and a controller operable to monitor the request and response signals exchanged between the first and second components and, in situations where a corresponding response signal is not issued within a predetermined time following a particular request signal, to cause the switch to disable the bus an to cause the fake response generator to issue a fake response signal to the component that issued the particular request signal for terminating the data transfer operation.

In accordance with a further aspect of the invention, there is provided a method of controlling a bus of a computer system including a first component and a second component interconnected via the bus for performing a data transfer operation, wherein the data transfer operation is initiated by an exchange of request and response signals, wherein a component which initiated a requested signal is operable to effect data transfer upon receipt of a response signal, the method comprising: monitoring the request signal on the bus; timing a period following the request signal; in the absence of a corresponding response signal within the period, disabling the bus and issuing a fake response signal to the component which initiated the request to thereby terminate the data transfer operation. The method may further comprise disabling the bus if a response to the request has not issued within the predetermined period of time. The method may further comprise issuing a fake response to terminate the request.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

Exemplary embodiments of the present invention will be described hereinafter, by way of example only, with reference to the accompanying drawings in which like reference signs relate to like elements and in which:

- Figure 1 is a schematic overview of a fault tolerant computer system incorporating an embodiment of the invention;
- Figure 2 is a schematic overview of a specific implementation of a system based on that of Figure 1;
- Figure 3 is a schematic representation of one implementation of a processing set;
- Figure 4 is a schematic representation of another example of a processing set;
- Figure 5 is a schematic representation of a further processing set;
- Figure 6 is a schematic block diagram of an embodiment of a bridge for the system of Figure 1;
- Figure 7 is a schematic block diagram of storage for the bridge of Figure 6;
- Figure 8 is a schematic block diagram of control logic of the bridge of Figure 6;
- Figure 9 is a schematic representation of a routing matrix of the bridge of Figure 6;
- Figure 10 is an example implementation of the bridge of Figure 6;
- Figure 11 is a state diagram illustrating operational states of the bridge of Figure 6;
- Figure 12 is a flow diagram illustrating stages in the operation of the bridge of Figure 6;
- Figure 13 is a detail of a stage of operation from Figure 12;
- Figure 14 illustrates the posting of I/O cycles in the system of Figure 1;
- Figure 15 illustrates the data stored in a posted write buffer;
- Figure 16 is a schematic representation of a slot response register;
- Figure 17 illustrates a dissimilar data write stage;
- Figure 18 illustrates a modification to Figure 17;
- Figure 19 illustrates a dissimilar data read stage;
- Figure 20 illustrates an alternative dissimilar data read stage;
- Figure 21 is a flow diagram summarising the operation of a dissimilar data write mechanism;
- Figure 22 is a schematic block diagram explaining arbitration within the system of Figure 1;
- Figure 23 is a state diagram illustrating the operation of a device bus arbiter;
- Figure 24 is a state diagram illustrating the operation of a bridge arbiter,
- Figure 25 is a timing diagram for PCI signals;
- Figure 26 is a schematic diagram illustrating the operation of the bridge of Figure 6 for direct memory access;
- Figure 27 is a flow diagram illustrating a direct memory access method in the bridge of Figure 6;
DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0021] Figure 1 is a schematic overview of a fault tolerant computing system 10 comprising a plurality of CPUsets (processing sets) 14 and 16 and a bridge 12. As shown in Figure 1, there are two processing sets 14 and 16, although in other embodiments there may be three or more processing sets. The bridge 12 forms an interface between the processing sets and I/O devices such as devices 28, 29, 30, 31 and 32. In this document, the term "processing set" is used to denote a group of one or more processors, possibly including memory, which output and receive common outputs and inputs. It should be noted that the alternative term mentioned above, "CPUset", could be used instead, and that these terms could be used interchangeably throughout this document. Also, it should be noted that the term "bridge" is used to denote any device, apparatus or arrangement suitable for interconnecting two or more buses of the same or different types.

[0022] The first processing set 14 is connected to the bridge 12 via a first processing set I/O bus (PA bus) 24. In the present instance a Peripheral Component Interconnect (PCI) bus. The second processing set 16 is connected to the bridge 12 via a second processing set I/O bus (PB bus) 26 of the same type as the PA bus 24 (i.e. here a PCI bus). The I/O devices are connected to the bridge 12 via a device I/O bus (D bus) 22, in the present instance also a PCI bus.

[0023] Although, in the particular example described, the buses 22, 24 and 26 are all PCI buses, this is merely by way of example, and in other embodiments other bus protocols may be used and the D-bus 22 may have a different protocol from that of the PA bus and the PB bus (P buses) 24 and 26.

[0024] The processing sets 14 and 16 and the bridge 12 are operable in synchronism under the control of a common clock 20, which is connected thereto by clock signal lines 21.

[0025] Some of the devices including an Ethernet (E-NET) interface 28 and a Small Computer System Interface (SCSI) interface 29 are permanently connected to the device bus 22, but other I/O devices such as I/O devices 30, 31 and 32 can be hot insertable into individual switched slots 33, 34 and 35. Dynamic field effect transistor (FET) switching can be provided for the slots 33, 34 and 35 to enable hot insertability of the devices such as devices 30, 31 and 32. The provision of the FETs enables an increase in the length of the D bus 22 as only those devices which are active are switched on, reducing the effective total bus length. It will be appreciated that the number of I/O devices which may be connected to the D bus 22, and the number of slots provided for them, can be adjusted according to a particular implementation in accordance with specific design requirements.

[0026] Figure 2 is a schematic overview of a particular implementation of a fault tolerant computer employing a bridge structure of the type illustrated in Figure 1. In Figure 2, the fault tolerant computer system includes a plurality (here four) of bridges 12 on first and second I/O motherboards (MB 40 and MB 42) in order to increase the number of I/O devices which may be connected and also to improve reliability and redundancy. Thus, in the embodiment shown in Figure 2, two processing sets 14 and 16 are each provided on a respective processing set board 44 and 46, with the processing sets boards 44 and 46 bridging the I/O motherboards MB 40 and MB 42. A first, master clock source 20A is mounted on the first motherboard 40 and a second, slave clock source 20B is mounted on the second motherboard 42. Clock signals are supplied to the processing set boards 44 and 46 via respective connections (not shown in Figure 2).

[0027] First and second bridges 12.1 and 12.2 are mounted on the first I/O motherboard 40. The first bridge 12.1 is connected to the processing sets 14 and 16 by P buses 24.1 and 26.1, respectively. Similarly, the second bridge 12.2 is connected to the processing sets 14 and 16 by P buses 24.2 and 26.2, respectively. The bridge 12.1 is connected to an I/O databus (D bus) 22.1 and the bridge 12.2 is connected to an I/O databus (D bus) 22.2.

[0028] Third and fourth bridges 12.3 and 12.4 are mounted on the second I/O motherboard 42. The bridge 12.3 is connected to the processing sets 14 and 16 by P buses 24.3 and 26.3, respectively. Similarly, the bridge 4 is connected to the processing sets 14 and 16 by P buses 24.4 and 26.4, respectively. The bridge 12.3 is connected to an I/O databus (D bus) 22.3 and the bridge 12.4 is connected to an I/O databus (D bus) 22.4.

[0029] It can be seen that the arrangement shown in Figure 2 can enable a large number of I/O devices to be connected to the two processing sets 14 and 16 via the D buses 22.1, 22.2, 22.3 and 22.4 for either increasing the range of I/O devices available, or providing a higher degree of redundancy, or both.

[0030] Figure 3 is a schematic overview of one possible configuration of a processing set, such as the processing set 14 of Figure 1. The processing set 16 could have the same configuration. In Figure 3, a plurality of processors (here four) 52 are connected by one or more buses 54 to a processing set bus controller 50. As shown in Figure 3,
one or more processing set output buses 24 are connected to the processing set bus controller 50, each processing set output bus 24 being connected to a respective bridge 12. For example, in the arrangement of Figure 1, only one processing set I/O bus (P bus) 24 would be provided, whereas in the arrangement of Figure 2, four such processing set I/O buses (P buses) 24 would be provided. In the processing set 14 shown in Figure 3, individual processors operate using the common memory 56, and receive inputs and provide outputs on the common P bus(es) 24.

[0031] Figure 4 is an alternative configuration of a processing set, such as the processing set 14 of Figure 1. Here a plurality of processor/memory groups 61 are connected to a common internal bus 64. Each processor/memory group 61 includes one or more processors 62 and associated memory 66 connected to an internal group bus 63. An interface 65 connects the internal group bus 63 to the common internal bus 64. Accordingly, in the arrangement shown in Figure 4, individual processing groups, with each of the processors 62 and associated memory 66 are connected via a common internal bus 64 to a processing set bus controller 60. The interfaces 65 enable a processor 62 of one processing group to operate not only on the data in its local memory 66, but also in the memory of another processing group 61 within the processing set 14. The processing set bus controller 60 provides a common interface between the common internal bus 64 and the processing set I/O bus(es) (P bus(es)) 24 connected to the bridge(s) 12. It should be noted that although only two processing groups 61 are shown in Figure 4, it will be appreciated that such a structure is not limited to this number of processing groups.

[0032] Figure 5 illustrates an alternative configuration of a processing set, such as the processing set 14 of Figure 1. Here a simple processing set includes a single processor 72 and associated memory 76 connected via a common bus 74 to a processing set bus controller 70. The processing set bus controller 70 provides an interface between the internal bus 74 and the processing set I/O bus(es) (P bus(es)) 24 for connection to the bridge(s) 12.

[0033] Accordingly, it will be appreciated from Figures 3, 4 and 5 that the processing set may have many different forms and that the particular choice of a particular processing set structure can be made on the basis of the processing requirement of a particular application and the degree of redundancy required. In the following description, it is assumed that the processing sets 14 and 16 referred to have a structure as shown in Figure 3. although it will be appreciated that another form of processing set could be provided.

[0034] The bridge(s) 12 are operable in a number of operating modes. These modes of operation will be described in more detail later. However, to assist in a general understanding of the structure of the bridge, the two operating modes will be briefly summarized here. In a first, combined mode, a bridge 12 is operable to route addresses and data between the processing sets 14 and 16 (via the PA and PB buses 24 and 26, respectively) and the devices (via the D bus 22). In this combined mode, I/O cycles generated by the processing sets 14 and 16 are compared to ensure that both processing sets are operating correctly. Comparison failures force the bridge 12 into an error limiting mode (EState) in which device I/O is prevented and diagnostic information is collected. In the second, split mode, the bridge 12 routes and arbitrates addresses and data from one of the processing sets 14 and 16 onto the D bus 22 and/or onto the other one of the processing sets 16 and 14, respectively. In this mode of operation, the processing sets 14 and 16 are not synchronized and no I/O comparisons are made. DMA operations are also permitted in both modes. As mentioned above, the different modes of operation, including the combined and split modes, will be described in more detail later. However, there now follows a description of the basic structure of an example of the bridge 12.

[0035] Figure 6 is a schematic functional overview of the bridge 12 of Figure 1. First and second processing set I/O bus interfaces, PA bus interface 84 and PB bus interface 86, are connected to the PA and PB buses 24 and 26, respectively. A device I/O bus interface, D bus interface 82, is connected to the D bus 22. It should be noted that the PA, PB and D bus interfaces need not be configured as separate elements but could be incorporated in other elements of the bridge. Accordingly, within the context of this document, where a references is made to a bus interface, this does not require the presence of a specific separate component, but rather the capability of the bridge to connect to the bus concerned, for example by means of physical or logical bridge connections for the lines of the buses concerned.

[0036] Routing (hereinafter termed a routing matrix) 80 is connected via a first internal path 94 to the PA bus interface 84 and via a second internal path 96 to the PB bus interface 86. The routing matrix 80 is further connected via a third internal path 92 to the D bus interface 82. The routing matrix 80 is thereby able to provide I/O bus transaction routing in both directions between the PA and PB bus interfaces 84 and 86. It is also able to provide routing in both directions between one or both of the PA and PB bus interfaces and the D bus interface 82. The routing matrix 80 is connected via a further internal path 100 to storage control logic 90. The storage control logic 90 controls access to bridge registers 110 and to a random access memory (SRAM) 126. The routing matrix 80 is therefore also operable to provide routing in both directions between the PA, PB and D bus interfaces 84, 86 and 82 and the storage control logic 90. The routing matrix 80 is controlled by bridge control logic 88 over control paths 98 and 99. The bridge control logic 88 is responsive to control signals, data and addresses on internal paths 93, 95 and 97, and also to clock signals on the clock line(s) 21.

[0037] In the embodiment of the invention, each of the P buses (PA bus 24 and PB bus 26) operates under a PCI protocol. The processing set bus controllers 50 (see Figure 3) also operate under the PCI protocol. Accordingly, the PA and PB bus interfaces 84 and 86 each provide all the functionality required for a compatible interface providing both master and slave operation for data transferred to and from the D bus 22 or internal memories and registers of...
the bridge in the storage subsystem 90. The bus interfaces 84 and 86 can provide diagnostic information to internal bridge status registers in the storage subsystem 90 on transition of the bridge to an error state (EState) or on detection of an I/O error.

The device bus interface 82 performs all the functionality required for a PCI compliant master and slave interface for transferring data to and from one of the PA and PB buses 84 and 86. The D bus 82 is operable during direct memory access (DMA) transfers to provide diagnostic information to internal status registers in the storage subsystem 90 of the bridge on transition to an EState or on detection of an I/O error.

Figure 7 illustrates in more detail the bridge registers 110 and the SRAM 124. The storage control logic 90 is connected via a path (e.g. a bus) 112 to a number of register components 114, 116, 118, 120. The storage control logic is also connected via a path (e.g. a bus) 128 to the SRAM 126 in which a posted write buffer component 122 and a dirty RAM component 124 are mapped. Although a particular configuration of the components 114, 116, 118, 120, 122 and 124 is shown in Figure 7, these components may be configured in other ways, with other components defined as regions of a common memory (e.g. a random access memory such as the SRAM 126, with the path 112/128 being formed by the internal addressing of the regions of memory). As shown in Figure 7, the posted write buffer 122 and the dirty RAM 124 are mapped to different regions of the SRAM memory 126, whereas the registers 114, 116, 118 and 120 are configured as separate from the SRAM memory.

Control and status registers (CSRs) 114 form internal registers which allow the control of various operating modes of the bridge, allow the capture of diagnostic information for an EState and for I/O errors, and control processing set access to PCI slots and devices connected to the D bus 22. These registers are set by signals from the routing matrix 80.

Dissimilar data registers (DDRs) 116 provide locations for containing dissimilar data for different processing sets to enable non-deterministic data events to be handled. These registers are set by signals from the PA and PB buses.

Bridge decode logic enables a common write to disable a data comparator and allow writes to two DDRs 116, one for each processing set 14 and 16.

A selected one of the DDRs can then be read in-sync by the processing sets 14 and 16. The DDRs thus provide a mechanism enabling a location to be reflected from one processing set (14/16) to another (16/14).

Slot response registers (SRRs) 118 determine ownership of device slots on the D bus 22 and to allow DMA to be routed to the appropriate processing set(s). These registers are linked to address decode logic.

Disconnect registers 120 are used for the storage of data phases of an I/O cycle which is aborted while data is in the bridge on the way to another bus. The disconnect registers 120 receive all data queued in the bridge when a target device disconnects a transaction, or as the EState is detected. These registers are connected to the routing matrix 80. The routing matrix can queue up to three data words and byte enables. Provided the initial addresses are voted as being equal, address target controllers derive addresses which increment as data is exchanged between the bridge and the destination (or target). Where a writer (for example a processor I/O write, or a DMA (D bus to P bus access)) is writing data to a target, this data can be caught in the bridge when an error occurs. Accordingly, this data is stored in the disconnect registers 120 when an error occurs. These disconnect registers can then be accessed on recovery from an EState to recover the data associated with the write or read cycle which was in progress when the EState was initiated.

Although shown separately, the DDRs 116, the SRRs 118 and the disconnect registers may form an integral part of the CSRs 114.

EState and error CSRs 114 provided for the capture of a failing cycle on the P buses 24 and 26, with an indication of the failing datum. Following a move to an EState, all of the writes initiated to the P buses are logged in the posted write buffer 122. These may be other writes that have been posted in the processing set bus controllers 50, or which may be initiated by software before an EState interrupt causes the processors to stop carrying out writes to the P buses 24 and 26.

A dirty RAM 124 is used to indicate which pages of the main memory 56 of the processing sets 14 and 16 have been modified by direct memory access (DMA) transactions from one or more devices on the D bus 22. Each page (e.g. each 8K page) is marked by a single bit in the dirty RAM 124 which is set when a DMA write occurs and can be cleared by a read and clear cycle initiated on the dirty RAM 124 by a processor 52 of a processing set 14 and 16.

The dirty RAM 124 and the posted write buffer 122 may both be mapped into the memory 124 in the bridge 12. This memory space can be accessed during normal read and write cycles for testing purposes.

Figure 8 is a schematic functional overview of the bridge control logic 88 shown in Figure 6.

All of the devices connected to the D bus 22 are addressed geographically. Accordingly, the bridge carries out decoding necessary to enable the isolating FETs for each slot before an access to those slots is initiated.

The address decoding performed by the address decode logic 136 and 138 essentially permits four basic access types:
- an out-of-sync access (i.e. not in the combined mode) by one processing set (e.g. processing set 14 of Figure 1) to the other processing set (e.g. processing set 16 of Figure 1), in which case the access is routed from the PA bus interface 84 to the PB bus interface 86;
- an access by one of the processing sets 14 and 16 in the split mode, or both processing sets 14 and 16 in the combined mode to an I/O device on the D bus 22, in which case the access is routed via the D bus interface 82;
- a DMA access by a device on the D bus 22 to one or both of the processing sets 14 and 16, which would be directed to both processing sets 14 and 16 in the combined mode, or to the relevant processing set 14 or 16 if out-of-sync, and if in a split mode to a processing set 14 or 16 which owns a slot in which the device is located; and
- a PCI configuration access to devices in I/O slots.

[0053] As mentioned above, geographic addressing is employed. Thus, for example, slot 0 on motherboard A has the same address when referred to by processing set 14 or by processing set 16.

[0054] Geographic addressing is used in combination with the PCI slot FET switching. During a configuration access mentioned above, separate device select signals are provided for devices which are not FET isolated. A single device select signal can be provided for the switched PCI slots as the FET signals can be used to enable a correct card. Separate FET switch lines are provided to each slot for separately switching the FETs for the slots.

[0055] The SRRs 118, which could be incorporated in the CSR registers 114, are associated with the address decode functions. The SRRs 118 serve in a number of different roles which will be described in more detail later. However, some of the roles are summarized here.

[0056] In a combined mode, each slot may be disabled so that writes are simply acknowledged without any transaction occurring on the device bus 22, whereby the data is lost. Reads will return meaningless data, once again without causing a transaction on the device board.

[0057] In the split mode, each slot can be in one of three states. The states are:
- Not owned;
- Owned by processing set A 14;
- Owned by processing set B 16.

[0058] A slot that is not owned by a processing set 14 or 16 making an access (this includes not owned or un-owned slots) cannot be accessed. Accordingly, such an access is aborted.

[0059] When a processing set 14 or 16 is powered off, all slots owned by it move to the un-owned state. A processing set 14 or 16 can only claim an un-owned slot, it cannot wrest ownership away from another processing set. This can only be done by powering off the other processing set, or by getting the other processing set to relinquish ownership.

[0060] The ownership bits are assessable and settable while in the combined mode, but have no effect until a split state is entered. This allows the configuration of a split system to be determined while still in the combined mode.

[0061] Each PCI device is allocated an area of the processing set address map. The top bits of the address are determined by the PCI slot. Where a device carries out DMA, the bridge is able to check that the device is using the correct address because a D bus arbiter informs the bridge which device is using the bus at a particular time. If a device access is a processing set address which is not valid for it, then the device access will be ignored. It should be noted that an address presented by a device will be a virtual address which would be translated by an I/O memory management unit in the processing set bus controller 50 to an actual memory address.

[0062] The addresses output by the address decoders are passed via the initiator and target controllers 138 and 140 to the routing matrix 80 via the lines 98 under control of a bridge controller 132 and an arbiter 134.

[0063] An arbiter 134 is operable in various different modes to arbitrate for use of the bridge on a first-come-first-served basis using conventional PCI bus signals on the P and D buses.

[0064] In a combined mode, the arbiter 134 is operable to arbitrate between the in-sync processing sets 14 and 16 and any initiators on the device bus 22 for use of the bridge 12. Possible scenarios are:
- processing set access to the device bus 22;
- processing set access to internal registers in the bridge 12;
- Device access to the processing set memory 56.

[0065] In split mode, both processing sets 14 and 16 must arbitrate the use of the bridge and thus access to the device bus 22 and internal bridge registers (e.g. CSR registers 114). The bridge 12 must also contend with initiators on the device bus 22 for use of that device bus 22.

[0066] Each slot on the device bus has an arbitration enable bit associated with it. These arbitration enable bits are cleared after reset and must be set to allow a slot to request a bus. When a device on the device bus 22 is suspected
A PCI bus interface in the processing set bus controller(s) 50 expects to be the master bus controller for the P bus concerned, that is it contains the PCI bus arbiter for the PA or PB bus to which it is connected. The bridge 12 cannot directly control access to the PA and PB buses 24 and 26. The bridge 12 competes for access to the PA or PB bus with the processing set on the bus concerned under the control of the bus controller 50 on the bus concerned.

Also shown in Figure 8 is a comparator 130 and a bridge controller 132. The comparator 130 is operable to compare I/O cycles from the processing sets 14 and 16 to determine any out-of-sync events. On determining an out-of-sync event, the comparator 130 is operable to cause the bridge controller 132 to activate an EState for analysis of the out-of-sync event and possible recovery therefrom.

Figure 9 is a schematic functional overview of the routing matrix 80.

The routing matrix 80 comprises a multiplexer 143 which is responsive to initiator control signals 98 from the initiator controller 138 of Figure 8 to select one of the PA bus path 94, PB bus path 96, D bus path 92 or internal bus path 100 as the current input to the routing matrix. Separate output buffers 144, 145, 146 and 147 are provided for output to each of the paths 94, 96, 92 and 100, with those buffers being selectively enabled by signals 99 from the target controller 140 of Figure 8. Between the multiplexer and the buffers 144-147 signals are held in a buffer 149. In the present embodiment three cycles of data for an I/O cycle will be held in the pipeline represented by the multiplexer 143, the buffer 149 and the buffers 144.

In Figures 6 to 9 a functional description of elements of the bridge has been given. Figure 10 is a schematic representation of a physical configuration of the bridge in which the bridge control logic 88, the storage control logic 90 and the bridge registers 110 are implemented in a first field programmable gate array (FPGA) 89, the routing matrix 80 is implemented in further FPGAs 80.1 and 80.2 and the SRAM 126 is implemented as one or more separate SRAMs addressed by a address control lines 127. The bus interfaces 82, 84 and 86 shown in Figure 6 are not separate elements, but are integrated in the FPGAs 80.1, 80.2 and 89. Two FPGAs 80.1 and 80.2 are used for the upper 32 bits 32-63 of a 64 bit PCI bus and the lower 32 bits 0-31 of the 64 bit PCI bus. It will be appreciated that a single FPGA could be employed for the routing matrix 80 where the necessary logic can be accommodated within the device. Indeed, where a FPGA of sufficient capacity is available, the bridge control logic, storage control logic and the bridge registers could be incorporated in the same FPGA as the routing matrix. Indeed many other configurations may be envisaged, and indeed technology other than FPGAs, for example one or more Application Specific Integrated Circuits (ASICs) may be employed. As shown in Figure 10, the FPGAs 89, 80.1 and 80.2 and the SRAM 126 are connected via internal bus paths 85 and path control lines 87.

Figure 11 is a transition diagram illustrating in more detail the various operating modes of the bridge. The bridge operation can be divided into three basic modes, namely an error state (EState) mode 150, a split state mode 156 and a combined state mode 158. The EState mode 150 can be further divided into 2 states.

After initial resetting on powering up the bridge, or following an out-of sync event, the bridge is in this initial EState 152. In this state, all writes are stored in the posted write buffer 122 and reads from the internal bridge registers (e.g., the CSR registers 116) are allowed, and all other reads are treated as errors (i.e. they are aborted). In this state, the individual processing sets 14 and 16 perform evaluations for determining a restart time. Each processing set 14 and 16 will determine its own restart timer timing. The timer setting depends on a "blame" factor for the transition to the EState. A processing set which determines that it is likely to have caused the error sets a long time for the timer. A processing set which thinks it unlikely to have caused the error sets a short time for the timer. The first processing set 14 and 16 which times out, becomes a primary processing set Accordingly, when this is determined, the bridge moves (153) to the primary EState 154.

When either processing set 14/16 has become the primary processing set, the bridge is then operating in the primary EState 154. This state allows the primary processing set to write to bridge registers (specifically the SRRs 118). Other writes are no longer stored in the posted write buffer, but are simply lost. Device bus reads are still aborted in the primary EState 154.

Once the EState condition is removed, the bridge then moves (155) to the split state 156. In the split state 156, access to the device bus 22 is controlled by the SRR registers 118 while access to the bridge storage is simply arbitrated. The primary status of the processing sets 14 and 16 is ignored. Transition to a combined operation is achieved by means of a sync_reset (157). After issue of the sync_reset operation, the bridge is then operable in the combined state 158, whereby all read and write accesses on the D bus 22 and the PA and PB buses 24 and 26 are allowed. All such accesses on the PA and PB buses 24 and 26 are compared in the comparator 130. Detection of a mismatch between any read and write cycles (with an exception of specific dissimilar data I/O cycles) cause a transition 151 to the EState. The various states described are controlled by the bridge controller 132.

The role of the comparator 130 is to monitor and compare I/O operations on the PA and PB buses in the combined state 158 and, in response to a mismatched signal, to notify the bridge controller 132, whereby the bridge controller 132 causes the transition 151 to the error state 150. The I/O operations can include all I/O operations initiated by the processing sets, as well as DMA transfers in respect of DMA initiated by a device on the device bus.
Table 1 below summarizes the various access operations which are allowed in each of the operational states.

<table>
<thead>
<tr>
<th>E State</th>
<th>D Bus - Read</th>
<th>D Bus - Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary EState</td>
<td>Master Abort</td>
<td>Stored in Post Write Buffer</td>
</tr>
<tr>
<td>Split</td>
<td>Master Abort</td>
<td>Lost</td>
</tr>
<tr>
<td>Combined</td>
<td>Controlled by SRR bits and arbitrated</td>
<td>Controlled by SRR bits and arbitrated</td>
</tr>
<tr>
<td></td>
<td>Allowed and compared</td>
<td>Allowed and compared</td>
</tr>
</tbody>
</table>

As described above, after an initial reset, the system is in the initial EState 152. In this state, neither processing sets 14 or 16 can access the D bus 22 or the P bus 26 or 24 of the other processing set 16 or 14. The internal bridge registers 110 of the bridge are accessible, but are read only.

A system running in the combined mode 158 transitions to the EState 150 where there is a comparison failure detected in this bridge, or alternatively a comparison failure is detected in another bridge in a multi-bridge system as shown, for example, in Figure 2. Also, transitions to an EState 150 can occur in other situations, for example in the case of a software controlled event forming part of a self test operation.

On moving to the EState 150, an interrupt is signaled to all or a subset of the processors of the processing sets via an interrupt line 95. Following this, all I/O cycles generated on a P bus 24 or 26 result in reads being returned with an exception and writes being recorded in the posted write buffer.

The operation of the comparator 130 will now be described in more detail. The comparator is connected to paths 94, 95, 96 and 97 for comparing address, data and selected control signals from the PA and PB bus interfaces 84 and 86. A failed comparison of in-sync accesses to device I/O bus 22 devices causes a move from the combined state 158 to the EState 150.

For processing set I/O read cycles, the address, command, address parity, byte enables and parity error parameters are compared.

If the comparison fails during the address phase, the bridge asserts a retry to the processing set bus controllers 50, which prevents data leaving the I/O bus controllers 50. No activity occurs in this case on the device I/O bus 22. On the processor(s) retrying, no error is returned.

If the comparison fails during a data phase (only control signals and byte enables are checked), the bridge signals a target-abort to the processing set bus controllers 50. An error is returned to the processors.

In the case of processing set I/O bus write cycles, the address, command, parity, byte enables and data parameters are compared.

If the comparison fails during the address phase, the bridge asserts a retry to the processing set bus controllers 50, which results in the processing set bus controllers 50 retrying the cycle again. The posted write buffer 122 is then active. No activity occurs on the device I/O bus 22.

If the comparison fails during the data phase of a write operation, no data is passed to the D bus 22. The failing data and any other transfer attributes from both processing sets 14 and 16 are stored in the disconnect registers 120, and any subsequent posted write cycles are recorded in the posted write buffer 122.

In the case of direct virtual memory access (DVMA) reads, the data control and parity are checked for each datum. If the data does not match, the bridge 12 terminates the transfer on the P bus. In the case of DVMA writes, control and parity error signals are checked for correctness.

Other signals in addition to those specifically mentioned above can be compared to give an indication of divergence of the processing sets. Examples of these are bus grants and various specific signals during processing set transfers and during DMA transfers.

Errors fall roughly into two types, those which are made visible to the software by the processing set bus controller 50 and those which are not made visible by the processing set bus controller 50 and hence need to be made visible by an interrupt from the bridge 12. Accordingly, the bridge is operable to capture errors reported in connection with processing set read and write cycles, and DMA reads and writes.

Clock control for the bridge is performed by the bridge controller 132 in response to the clock signals from the clock line 21. Individual control lines from the controller 132 to the various elements of the bridge are not shown in Figures 6 to 10.

Figure 12 is a flow diagram illustrating a possible sequence of operating stages where lockstep errors are detected during a combined mode of operation.

Stage S1 represents the combined mode of operation where lockstep error checking is performed by the comparator 130 shown in Figure 8.

In Stage S2, a lockstep error is assumed to have been detected by the comparator 130.
In Stage S3, the current state is saved in the CSR registers 114 and posted writes are saved in the posted write buffer 122 and/or in the disconnect registers 120.

Figure 13 illustrates Stage S3 in more detail. Accordingly, in Stage S31, the bridge controller 132 detects whether the lockstep error notified by the comparator 130 has occurred during a data phase in which it is possible to pass data to the device bus 22. In this case, in Stage S32, the bus cycle is terminated. Then, in Stage S33 the data phases are stored in the disconnect registers 120 and control then passes to Stage S35 where an evaluation is made as to whether a further I/O cycle needs to be stored. Alternatively, if at Stage S31, it is determined that the lockstep error did not occur during a data phase, the address and data phases for any posted write I/O cycles are stored in the posted write buffer 122. At Stage S34, if there are any further posted write I/O operations pending, these are also stored in the posted write buffer 122.

Stage S3 is performed at the initiation of the initial error state 152 shown in Figure 11. In this state, the first and second processing sets arbitrate for access to the bridge. Accordingly, in Stage S31-S35, the posted write address and data phases for each of the processing sets 14 and 16 are stored in separate portions of the posted write buffer 122, and/or in the single set of disconnect registers as described above.

Figure 14 illustrates the source of the posted write I/O cycles which need to be stored in the posted write buffer 122. During normal operation of the processing sets 14 and 16, output buffers 162 in the individual processors contain I/O cycles which have been posted for transfer via the processing set bus controllers 50 to the bridge 12 and eventually to the device bus 22. Similarly, buffers 160 in the processing set controllers 50 also contain posted I/O cycles for transfer over the buses 24 and 26 to the bridge 12 and eventually to the device bus 22.

Accordingly, it can be seen that when an error state occurs, I/O write cycles may already have been posted by the processors 52, either in their own buffers 162, or already transferred to the buffers 160 of the processing set bus controllers 50. It is the I/O write cycles in the buffers 162 and 160 which gradually propagate through and need to be stored in the posted write buffer 122.

As shown in Figure 15, a write cycle 164 posted to the posted write buffer 122 can comprise an address field 165 including an address and an address type, and between one and 16 data fields 166 including a byte enable field and the data itself.

The data is written into the posted write buffer 122 in the EState unless the initiating processing set has been designated as a primary CPU set. At that time, non-primary writes in an EState still go to the posted write buffer even after one of the CPU sets has become a primary processing set. An address pointer in the CSR registers 114 points to the next available posted write buffer address, and also provides an overflow bit which is set when the bridge attempts to write past the top of the posted write buffer for any one of the processing sets 14 and 16. Indeed, in the present implementation, only the first 16 K of data is recorded in each buffer. Attempts to write beyond the top of the posted write buffer are ignored. The value of the posted write buffer pointer can be cleared at reset, or by software using a write under the control of a primary processing set.

Returning to Figure 12, after saving the status and posted writes, at Stage S4 the individual processing sets independently seek to evaluate the error state and to determine whether one of the processing sets is faulty. This determination is made by the individual processors in an error state in which they individually read status from the control state and EState registers 114. During this error mode, the arbiter 134 arbitrates for access to the bridge 12.

In Stage S5, one of the processing sets 14 and 16 establishes itself as the primary processing set. This is determined by each of the processing sets identifying a time factor based on the estimated degree of responsibility for the error, whereby the first processing set to time out becomes the primary processing set. In Stage S5, the status is recovered for that processing set and is copied to the other processing set. The primary processing is able to access the posted write buffer 122 and the disconnect registers 120.

In Stage S6, the bridge is operable in a split mode. If it is possible to re-establish an equivalent status for the first and second processing sets, then a reset is issued at Stage S7 to put the processing sets in the combined mode at Stage S1. However, it may not be possible to re-establish an equivalent state until a faulty processing set is replaced. Accordingly, the system will stay in the Split mode of Stage S6 in order to continued operation based on a single processing set. After replacing the faulty processing set the system could then establish an equivalent state and move via Stage S7 to Stage S1.

As described above, the comparator 130 is operable in the combined mode to compare the I/O operations output by the first and second processing sets 14 and 16. This is fine as long as all of the I/O operations of the first and second processing sets 14 and 16 are fully synchronized and deterministic. Any deviation from this will be interpreted by the comparator 130 as a loss of lockstep. This is in principle correct as even a minor deviation from identical outputs, if not trapped by the comparator 130, could lead to the processing sets diverging further from each other as the individual processing sets act on the deviating outputs. However, a strict application of this puts significant constraints on the design of the individual processing sets. An example of this is that it would not be possible to have independent time of day clocks in the individual processing sets operating under their own clocks. This is because it is impossible to obtain two crystals which are 100% identical in operation. Even small differences in the phase of the
clocks could be critical as to whether the same sample is taken at any one time, for example either side of a clock transition for the respective processing sets.

Accordingly, a solution to this problem employs the dissimilar data registers (DDR) 116 mentioned earlier. The solution is to write data from the processing sets into respective DDRs in the bridge while disabling the comparison of the data phases of the write operations and then to read a selected one of the DDRs back to each processing set, whereby each of the processing sets is able to act on the same data.

Figure 17 is a schematic representation of details of the bridge of Figures 6 to 10. It will be noted that details of the bridge not shown in Figure 6 to 8 are shown in Figure 17, whereas other details of the bridge shown in Figures 6 to 8 are not shown in Figure 17, for reasons of clarity.

The DDRs 116 are provided in the bridge registers 110 of Figure 7, but could be provided elsewhere in the bridge in other embodiments. One DDR 116 is provided for each processing set in the example of the multi-processor system of Figure 1 where two processing sets 14 and 16 are provided, two DDRs 116A and 116B are provided, one for each of the first and second processing sets 14 and 16, respectively.

Figure 17 represents a dissimilar data write stage. The addressing logic 136 is shown schematically to comprise two decoder sections, one decoder section 136A for the first processing set and one decoder section 136B for the second processing set 16. During an address phase of a dissimilar data I/O write operation each of the processing sets 14 and 16 outputs the same predetermined address DDR-W which is separately interpreted by the respective first and second decoding sections 136A and 136B as addressing the respective first and second respective DDRs 116A and 116B. As the same address is output by the first and second processing sets 14 and 16, this is not interpreted by the comparator 130 as a lockstep error.

The decoding section 136A, or the decoding section 136B, or both are arranged to further output a disable signal 137 in response to the predetermined write address supplied by the first and second processing sets 14 and 16. This disable signal is supplied to the comparator 130 and is operative during the data phase of the write operation to disable the comparator. As a result, the data output by the first processing set can be stored in the first DDR 116A and the data output by the second processing set can be stored in the second DDR 116B without the comparator being operative to detect a difference, even if the data from the first and second processing sets is different. The first decoding section 136A is operable to cause the routing matrix to store the data from the first processing set 14 in the first DDR 116A and the second decoding section 136B is operable to cause the routing matrix to store the data from the second processing set 16 in the second DDR 116B. At the end of the data phase the comparator 130 is once again enabled to detect any differences between I/O address and/or data phases as indicative of a lockstep error.

Following the writing of the dissimilar data to the first and second DDRs 116A and 116B, the processing sets are then operable to read the data from a selected one of the DDRs 116A/116B.

Figure 18 illustrates an alternative arrangement where the disable signal 137 is negated and is used to control a gate 131 at the output of the comparator 130. When the disable signal is active the output of the comparator is disabled, whereas when the disable signal is inactive the output of the comparator is enabled.

Figure 19 illustrates the reading of the first DDR 116A in a subsequent dissimilar data read stage. As illustrated in Figure 19, each of the processing sets 14 and 16 outputs the same predetermined address DDR-RA which is separately interpreted by the respective first and second decoding sections 136A and 136B as addressing the same DDR, namely the first DDR 116A. As a result, the content of the first DDR 116A is read by both of the processing sets 14 and 16, thereby enabling those processing sets to receive the same data. This enables the two processing sets 14 and 16 to achieve deterministic behavior, even if the source of the data written into the DDRs 116 by the processing sets 14 and 16 was not deterministic.

As an alternative, the processing sets could each read the data from the second DDR 116B. Figure 20 illustrates the reading of the second DDR 116B in a dissimilar data read stage following the dissimilar data write stage of Figure 15. As illustrated in Figure 20, each of the processing sets 14 and 16 outputs the same predetermined address DDR-RB which is separately interpreted by the respective first and second decoding sections 136A and 136B as addressing the same DDR, namely the second DDR 116B. As a result, the content of the second DDR 116B is read by both of the processing sets 14 and 16, thereby enabling those processing sets to receive the same data. As with the dissimilar data read stage of Figure 19, this enables the two processing sets 14 and 16 to achieve deterministic behavior, even if the source of the data written into the DDRs 116 by the processing sets 14 and 16 was not deterministic.

The selection of which of the first and second DDRs 116A and 116B to read can be determined in any appropriate manner by the software operating on the processing modules. This could be done on the basis of a simple selection of one or the other DDRs, or on a statistical basis or randomly or in any other manner as long as the same choice of DDR is made by both or all of the processing sets.

Figure 21 is a flow diagram summarizing the various stages of operation of the DDR mechanism described above.

In stage S10, a DDR write address DDR-W is received and decoded by the address decoders sections 136A and 136B during the address phase of the DDR write operation.
[0118] In stage S11, the comparator 130 is disabled.

[0119] In stage S12, the data received from the processing sets 14 and 16 during the data phase of the DDR write operation is stored in the first and second DDRs 116A and 116B, respectively, as selected by the first and second decode sections 136A and 136B, respectively.

[0120] In stage S13, a DDR read address is received from the first and second processing sets and is decoded by the decode sections 136A and 136B, respectively.

[0121] If the received address DDR-RA is for the first DDR 116A, then in stage S14 the content of that DDR 116A is read by both of the processing sets 14 and 16.

[0122] Alternatively, 116A if the received address DDR-RB is for the second DDR 116B, then in stage S15 the content of that DDR 116B is read by both of the processing sets 14 and 16.

[0123] Figure 22 is a schematic representation of the arbitration performed on the respective buses 22, 24 and 26, and the arbitration for the bridge itself.

[0124] Each of the processing set bus controllers 50 in the respective processing sets 14 and 16 includes a conventional PCI master bus arbiter 180 for providing arbitration to the respective buses 24 and 26. Each of the master arbiters 180 is responsive to request signals from the associated processing set bus controller 50 and the bridge 12 on respective request (REQ) lines 181 and 182. The master arbiters 180 allocate access to the bus on a first-come-first-served basis, issuing a grant (GNT) signal to the winning party on an appropriate grants line 183 or 184.

[0125] A conventional PCI bus arbiter 185 provides arbitration on the D bus 22. The D bus arbiter 185 can be configured as part of the D bus interface 82 of Figure 6 or could be separate therefrom. As with the P bus master arbiters 180, the D bus arbiter is responsive to request signals from the contending devices, including the bridge and the devices 30, 31, etc. connected to the device bus 22. Respective request lines 186, 187, 188, etc. for each of the entities competing for access to the D bus 22 are provided for the request signals (REQ). The D bus arbiter 185 allocates access to the D bus on a first-come-first-served basis, issuing a grant (GNT) signal to the winning entity via respective grant lines 189, 190, 192, etc.

[0126] Figure 23 is a state diagram summarising the operation of the D bus arbiter 185. In a particular embodiment up to six request signals may be produced by respective D bus devices and one by the bridge itself. On a transition into the GRANT state, these are sorted by a priority encoder and a request signal (REQ#) with the highest priority is registered as the winner and gets a grant (GNT#) signal. Each winner which is selected modifies the priorities in a priority encoder so that given the same REQ# signals on the next move to grant. A different device has the highest priority, hence each device has a "fair" chance of accessing DEVs. The bridge REQ# has a higher weighting than D bus devices and will, under very busy conditions, get the bus for every second device.

[0127] If a device requesting the bus fails to perform a transaction within 16 cycles it may lose GNT# via the BACKOFF state. BACKOFF is required as, under PCI rules, a device may access the bus only after GNT# is removed. Devices may only be granted access to D bus if the bridge is not in the not in the EState. A new GNT# is produced at the times when the bus is idle.

[0128] In the GRANT and BUSY states, the FETs are enabled and an accessing device is known and forwarded to the D bus address decode logic for checking against a DMA address provided by the device.

[0129] Turning now to the bridge arbiter 134, this allows access to the bridge for the first device which asserts the PCI FRAME# signal indicating an address phase. Figure 24 is a state diagram summarising the operation of the bridge arbiter 134.

[0130] As with the D bus arbiter, a priority encoder can be provided to resolve access attempts which collide. In this case "a collision" the loser/losers are retried which forces them to give up the bus. Under PCI rules retrieved devices must try repeatedly to access the bridge and this can be expected to happen.

[0131] To prevent devices which are very quick with their retry attempt from hogging the bridge, retried interfaces are remembered and assigned a higher priority. These remembered retries are prioritised in the same way as address phases. However as a precaution this mechanism is timed out so as not to get stuck waiting for a faulty or dead device. The algorithm employed prevents a device which hasn’t yet been retrieved, but which would be a higher priority retry than a device currently waiting for, from being retrieved at the first attempt.

[0132] In combined operations a PA or PB bus input selects which P bus interface will win a bridge access. Both are informed they won. Allowed selection enables latent fault checking during normal operation. EState prevents the D bus from winning.

[0133] The bridge arbiter 134 is responsive to standard PCI signals provided on standard PCI control lines 22, 24 and 25 to control access to the bridge 12.

[0134] Figure 25 illustrates signals associated with an I/O operation cycle on the PCI bus. A PCI frame signal (FRAME#) is initially asserted. At the same time, address (A) signals will be available on the DATA BUS and the appropriate command (write/read) signals (C) will be available on the command bus (CMD BUS). Shortly after the frame signal being asserted low, the initiator ready signal (IRDY#) will also be asserted low. When the device responds, a device selected signal (DEVSEL#) will be asserted low. When a target ready signal is asserted low (TRDY#), data
The bridge is operable to allocate access to the bridge resources and thereby to negotiate allocation of a target bus in response to the FRAME# being asserted low for the initiator bus concerned. Accordingly, the bridge arbiter 134 is operable to allocate access to the bridge resources and/or to a target bus on a first-come-first-served basis in response to the FRAME# being asserted low. As well as the simple first-come-first-served basis, the arbiters may be additionally provided with a mechanism for logging the arbitration requests, and can imply a conflict resolution based on the request and allocation history where two requests are received at an identical time. Alternatively, a simple priority can be allocated to the various requesters, whereby, in the case of identically timed requests, a particular requester always wins the allocation process.

Each of the slots on the device bus 22 has a slot response register (SRR) 118, as well as other devices connected to the bus, such as a SCSI interface. Each of the SRRs 118 contains bits defining the ownership of the slots, or the devices connected to the slots on the direct memory access bus. In this embodiment, and for reasons to be elaborated below, each SRR 118 comprises a four bit register. However, it will be appreciated that a larger register will be required to determine ownership between more than two processing sets. For example, if three processing sets are provided, then a five bit register will be required for each slot.

Figure 16 illustrates schematically one such four bit register 600. As shown in Figure 16, a first bit 602 is identified as SRR[0], a second bit 604 is identified as SRR[1], a third bit 606 is identified as SRR[2] and a fourth bit 608 is identified as SRR[3].

Bit SRR[0] is a bit which is set when writes for valid transactions are to be suppressed. Bit SRR[1] is set when the device slot is owned by the first processing set 14. This defines the access route between the first processing set 14 and the device slot. When this bit is set, the first processing set 14 can always be master of a device slot 22, while the ability for the device slot to be master depends on whether bit SRR[3] is set.

Bit SRR[2] is set when the device slot is owned by the second processing set 16. This defines the access route between the second processing set 16 and the device slot. When this bit is set, the second processing set 16 can always be master of the device slot or bus 22, while the ability for the device slot to be master depends on whether bit SRR[3] is set.

Bit SRR[3] is an arbitration bit which gives the device slot the ability to become master of the device bus 22, but only if it is owned by one of the processing sets 14 and 16, that is if one of the SRR[1] and SRR[2] bits is set.

When the fake bit (SRR[0]) of an SRR 118 is set, writes to the device for that slot are ignored and do not appear on the device bus 22. Reads return indeterminate data without causing a transaction on the device bus 22. In the event of an I/O error the fake bit SRR[0] of the SRR 118 corresponding to the device which caused the error is set by the hardware configuration of the bridge to disable further access to the device slot concerned. An interrupt may also be generated by the bridge to inform the software which originated the access leading to the I/O error that the error has occurred. The fake bit has an effect whether the system is in the split or the combined mode of operation.

The ownership bits only have effect, however, in the split system mode of operation. In this mode, each slot can be in three states:

- Not-owned;
- Owned by processing set 14; and
- Owned by processing set 16

This is determined by the two SRR bits SRR[1] and SRR[2], with SRR[1] being set when the slot is owned by processing set 14 and SRR[2] being set when the slot is owned by processing set B. If the slot is un-owned, then neither bit is set (both bits set is an illegal condition and is prevented by the hardware).

A slot which is not owned by the processing set making the access (this includes un-owned slots) cannot be accessed and results in an abort. A processing set can only claim an un-owned slot; it cannot wrest ownership away from another processing set. This can only be done by power-off the other processing set. When a processing set is powered off, all slots owned by it move to the un-owned state. Whilst it is not possible for a processing set to wrest ownership from another processing set, it is possible for a processing set to give ownership to another processing set.

The owned bits can be altered when in the combined mode of operation but they have no effect until the split mode is entered.

Table 2 below summarizes the access rights as determined by an SRR 118.
In an alternative example, where the SRR for the device is set to 0101, the setting of SRR[2] logic high indicates that the device is owned by processing set B. However, as the device is malfunctioning, SRR[3] is set logic low and the device is not allowed access to the processing set SRR[0] is set high so that any writes to the device are ignored and reads therefrom return indeterminate data. In this way, the malfunctioning device is effectively isolated from the processing set, and provides indeterminate data to satisfy any device drivers, for example, that might be looking for a response from the device.

Table 2 illustrates the operation of the bridge 12 for direct memory access by a device such as one of the devices 28, 29, 30, 31 and 32 to the memory 56 of the processing sets 14 and 16. When the D bus arbiter 185 receives a direct memory access (DMA) request 193 from a device (e.g., device 30 in slot 33) on the device bus, the D bus arbiter determines whether to allocate the bus to that slot. As a result of this granting procedure, the D-bus arbiter knows the slot which has made the DMA request 193. The DMA request is supplied to the address decoder 142 in the bridge, where the addresses associated with the request are decoded. The address decoder is responsive to the D bus grant signal 194 for the slot concerned to identify the slot which has been granted access to the D bus for the DMA request.

The address decode logic 142 holds or has access to a geographic address map 196, which identifies the relationship between the processor address space and the slots as a result of the geographic address employed. This geographic address map 196 could be held as a table in the bridge memory 126, along with the posted write buffer 122 and the dirty RAM 124. Alternatively, it could be held as a table in a separate memory element, possibly forming part of the address decoder 142 itself. The map 182 could be configured in a form other than a table.

The address decode logic 142 is configured to verify the correctness of the DMA addresses supplied by the device 30. In one embodiment of the invention, this is achieved by comparing four significant address bits of the address supplied by the device 30 with the corresponding four address bits of the address held in the geographic addressing map 196 for the slot identified by the D bus grant signal for the DMA request. In this example, four address bits are...
sufficient to determine whether the address supplied is within the correct address range. In this specific example, 32
bit PCI bus addresses are used, with bits 31 and 30 always being set to 1, bit 29 being allocated to identify which of
two bridges on a motherboard is being addressed (see Figure 2) and bits 28 to 26 identifying a PCI device. Bits 25-0
define an offset from the base address for the address range for each slot. Accordingly, by comparing bits 29-26, it is
possible to identify whether the address(es) supplied fall(s) within the appropriate address range for the slot concerned.
It will be appreciated that in other embodiments a different number of bits may need to be compared to make this
determination depending upon the allocation of the addresses.

[0153] The address decode logic 142 could be arranged to use the bus grant signal 184 for the slot concerned to
identify a table entry for the slot concerned and then to compare the address in that entry with the address(es) received
with the DMA request as described above. Alternatively, the address decode logic 142 could be arranged to use the
address(es) received with the DMA address to address a relational geographic address map and to determine a slot
number therefrom, which could be compared to the slot for which the bus grant signal 194 is intended and thereby to
determine whether the addresses fall within the address range appropriate for the slot concerned.

[0154] Either way, the address decode logic 142 is arranged to permit DMA to proceed if the DMA addresses fall
within the expected address space for the slot concerned. Otherwise, the address decoder is arranged to ignore the
slots and the physical addresses.

[0155] The address decode logic 142 is further operable to control the routing of the DMA request to the appropriate
processing set(s) 14/16. If the bridge is in the combined mode, the DMA access will automatically be allocated to all
of the in-sync processing sets 14/16. The address decode logic 142 will be aware that the bridge is in the combined
mode as it is under the control of the bridge controller 132 (see Figure 8). However, where the bridge is in the split
mode, a decision will need to be made as to which, if any, of the processing sets the DMA request is to be sent

[0156] When the system is in split mode, the access will be directed to a processing set 14 or 16 which owns the
slot concerned. If the slot is un-owned, then the bridge does not respond to the DMA request. In the split mode, the
address decode logic 142 is operable to determine the ownership of the device originating the DMA request by ac-
accessing the SRR 118 for the slot concerned. The appropriate slot can be identified by the D bus grant signal. The
address decode logic 142 is operable to control the target controller 140 (see Figure 8) to pass the DMA request to
the appropriate processing set(s) 14/16 based on the ownership bits SRR[1] and SRR[2]. If bit SRR[1] is set, the first
processing set 14 is the owner and the DMA request is passed to the first processing set. If bit SRR[2] is set, the
second processing set 16 is the owner and the DMA request is passed to the second processing set If neither of the
bit SRR[1] and SRR[2] is set, then the DMA request is ignored by the address decoder and is not passed to either of
the processing sets 14 and 16.

[0157] Figure 27 is a flow diagram summarizing the DMA verification process as illustrated with reference to Figure
24.

[0158] In stage S20, the D-bus arbiter 160 arbitrates for access to the D bus 22.

[0159] In stage S21, the address decoder 142 verifies the DMA addresses supplied with the DMA request by ac-
accessing the geographic address map.

[0160] In stage S22, the address decoder ignores the DMA access where the address falls outside the expected
range for the slot concerned.

[0161] Alternatively, as represented by stage S23, the actions of the address decoder are dependent upon whether
the bridge is in the combined or the split mode.

[0162] If the bridge is in the combined mode, then in stage S24 the address decoder controls the target controller
140 (see Figure 8) to cause the routing matrix 80 (see Figure 6) to pass the DMA request to both processing sets 14
and 16.

[0163] If the bridge is in the split mode, the address decoder is operative to verify the ownership of the slot concerned
by reference to the SRR 118 for that slot in stage S25.

[0164] If the slot is allocated to the first processing set 14 (i.e. the SRR[1] bit is set), then in stage S26 the address
decoder 142 controls the target controller 140 (see Figure 8) to cause the routing matrix 80 (see Figure 6) to pass the
DMA request to first processing set 14.

[0165] If the slot is allocated to the second processing set 16 (i.e. the SRR[2] bit is set), then in stage S27 the address
decoder 142 controls the target controller 140 (see Figure 8) to cause the routing matrix 80 (see Figure 6) to pass the
DMA request to the second processing set 16.

[0166] If the slot is unallocated (i.e. neither the SRR[1] bit nor the SRR[2] bit is set), then in step S18 the address
decoder 142 ignores or discards the DMA request and the DMA request is not passed to the processing sets 14 and 16.
[0167] A DMA, or direct vector memory access (DVMA), request sent to one or more of the processing sets causes
the necessary memory operations (read or write as appropriate) to be effected on the processing set memory.

[0168] There now follows a description of an example of a mechanism for enabling automatic recovery from an
EState (see Figure 11).

[0169] The automatic recovery process includes reintegration of the state of the processing sets to a common status
in order to attempt a restart in lockstep. To achieve this, the processing set which asserts itself as the primary processing set as described above copies its complete state to the other processing set. This involves ensuring that the content of the memory of both processors is the same before trying a restart in lockstep mode.

However, a problem with the copying of the content of the memory from one processing set to the other is that during this copying process a device connected to the D bus 22 might attempt to make a direct memory access (DMA) request for access to the memory of the primary processing set. If DMA is enabled, then a write made to an area of memory which has already been copied would result in the memory state of the two processors at the end of the copy not being the same. In principle, it would be possible to inhibit DMA for the whole of the copy process. However, this would be undesirable, bearing in mind that it is desirable to minimise the time that the system or the resources of the system are unavailable. As an alternative, it would be possible to retry the whole copy operation when a DMA operation has occurred during the period of the copy. However, it is likely that further DMA operations would be performed during the copy retry, and accordingly this is not a good option either. Accordingly, in the present system, a dirty RAM 124 is provided in the bridge. As described earlier the dirty RAM 124 is configured as part of the bridge SRAM memory 126.

The dirty RAM 124 comprises a bit map having a dirty indicator, for example a dirty bit, for each block, or page, of memory. The bit for a page of memory is set when a write access to the area of memory concerned is made. In an embodiment of the invention one bit is provided for every 8K page of main processing set memory. The bit for a page of processing set memory is set automatically by the address decoder 142 when this decodes a DMA request for that page of memory for either of the processing sets 14 or 16 from a device connected to the D bus 22. The dirty RAM can be reset, or cleared when it is read by a processing set, for example by means of read and clear instructions at the beginning of a copy pass, so that it can start to record pages which are dirtied since a given time.

The dirty RAM 124 can be read word by word. If a large word size is chosen for reading the dirty RAM 124, this will optimise the reading and resetting of the dirty RAM 124.

Accordingly, at the end of the copy pass the bits in the dirty RAM 124 will indicate those pages of processing set memory which have been changed (or dirtied) by DMA writes during the period of the copy. A further copy pass can then be performed for only those pages of memory which have been dirtied. This will take less time that a full copy of the memory. Accordingly, there are typically less pages marked as dirty at the end of the next copy pass and, as a result, the copy passes can become shorter and shorter. As some time it is necessary to decide to inhibit DMA writes for a short period for a final, short, copy pass, at the end of which the memories of the two processing sets will be the same and the primary processing set can issue a reset operation to restart the combined mode.

The dirty RAM 124 is set and cleared in both the combined and split modes. This means that in split mode the dirty RAM 124 may be cleared by either processing set.

The dirty RAM 124 address is decoded from bits 13 to 28 of the PCI address presented by the D bus device. Erroneous accesses which present illegal combinations of the address bits 29 to 31 are mapped into the dirty RAM 124 and a bit is dirtied on a write, even though the bridge will not pass these transactions to the processing sets.

When reading the dirty RAM 124, the bridge defines the whole area from 0x00008000 to 0x0000ffff as dirty RAM and will clear the contents of any location in this range on a read.

As an alternative to providing a single dirty RAM 124 which is cleared on being read, another alternative would be to provide two dirty RAMs which are used in a toggle mode, with one being written to while another is read.

Figure 28 is a flow diagram summarising the operation of the dirty RAM 124.

In stage S41, the primary processing set reads the dirty RAM 124 which has the effect of resetting the dirty RAM 124.

In stage S42, the primary processor (e.g. processing set 14) copies the whole of its memory 56 to the memory 56 of the other processing set (e.g. processing set 16).

In stage S43, the primary processing set reads the dirty RAM 124 which has the effect of resetting the dirty RAM 124.

In stage S44, the primary processor determines whether less than a predetermined number of bits have been written in the dirty RAM 124.

If more than the predetermined number of bits have been set, then the processor in stage S45 copies those pages of its memory 56 which have been dirtied, as indicated by the dirty bits read from the dirty RAM 124 in stage S43, to the memory 56 of the other processing set. Control then passes back to stage S43.

If, in stage S44, it is determined less than the predetermined number of bits have been written in the dirty RAM 124, then in Stage S45 the primary processor causes the bridge to inhibit DMA requests from the devices connected to the D bus 22. This could, for example, be achieved by clearing the arbitration enable bit for each of the device slots, thereby denying access of the DMA devices to the D bus 22. Alternatively, the address decoder 142 could be configured to ignore DMA requests under instructions from the primary processor. During the period in which DMA accesses are prevented, the primary processor then makes a final copy pass from its memory to the memory 56 of the other processor for those memory pages corresponding to the bits set in the dirty RAM 124.
In stage S47 the primary processor can issue a reset operation for initiating a combined mode.

In stage S48, DMA accesses are once more permitted.

As has been described above in detail, the computer system 10 is provided with various mechanisms to enable the system to survive various different situations that would otherwise cause the system to crash. Some of those mechanisms are concerned with surviving errors associated with I/O device failures, for example, and others relate to errors that can arise when the multi-processors of the system deviate from a lock-step mode.

One final, and hitherto undescribed problem that could occur in a computer system such as that described herein is concerned with a failure of either the processing sets 14, 16 or the bridge 12. The following description illustrates a mechanism for limiting the effect of any such failures.

As shown in Figure 29, the computer system comprises, in part, the bridge 12 provided on the bridge motherboard 42, the processing set 14, 16 (only one of which is shown for illustrative purposes only) and a PCI bus 24, 26 connecting the processing set 14, 16 to the bridge 12. As described above in Figure 3, the preferred processing set architecture comprises a processing set controller 50 to which a plurality of processors (not shown in Figure 29) are connected. The PCI bus interconnecting the processing set controller 50 and the bridge 12 typically comprises a plurality of bus lines, but one line is shown here for clarity.

In normal operation, PCI protocol signals propagate in both directions between the processing set controller 50 and the bridge 12. A problem exists in that the PCI bus protocol is provided with only a rudimentary fault recovery mechanism. Thus, it has been noted that crashes can be caused if either the processing set or the bridge should fail. These crashes typically result because the surviving bridge or processing set continues to attempt to distribute transactions along the PCI bus, and these distributed transactions tend to back-up whereupon either the bridge or the processing set can deadlock.

As shown in Figure 30a, the PCI protocol includes two signals, a first signal IRDY# indicating that an initiator is ready to supply data, and a second signal TRDY# indicating that a target is ready to receive data. It will be understood of course that the initiator refers to the component which is initiating the signals and that the target refers to the component which is to receive the signals.

The IRDY# and TRDY# signal exchange takes place once the initiator address phase has been completed. In the address phase, the initiator addresses the device to which it intends to send data, and the addressed device (the target) responds with a DEVSEL# signal confirming to the initiator that it has been selected by the initiator. In effect, the address phase and the DEVSEL# exchange comprises a hand-shaking process between the initiator and target.

Once the handshaking process has been successfully completed, it is then necessary to determine that the initiator is ready to supply data, and that the target is ready to receive that supplied data. The IRDY# and TRDY# signals are used for this purpose.

Figures 30b and 30d schematically indicate possible IRDY# and TRDY# signals when the PCI bus is operating correctly. As shown in Figure 30b, an IRDY# signal has issued at time "a" indicating that the initiator is ready to supply data. A period of time later, a TRDY# signal is outputted indicating that the target is ready. When both the IRDY# and TRDY# signals are asserted at the same clock edge (i.e. at time "b"), the initiator transfers a datum to the target.

For example, for Programmed Input/Output (PIO) from the processing set 14, 16 via the bridge 12, an initiator ready signal IRDY# is generated on the PCI bus 24 once the address phase of the PCI protocol has been completed, and if everything is working properly a TRDY# signal is subsequently generated on the PCI bus when the bridge has indicated that it is ready to receive data from the processing set.

If a fault has occurred, for example as shown in Figure 30a, then no response is received within a predetermined period of time from the target and an error is assumed to have occurred. Figure 30a illustrates an example of PIO where an error is assumed to have occurred. In this example, the processing set has initiated an IRDY# signal to indicate that the processing set as initiator is ready and this signal has not been answered by a TRDY# signal indicating that the bridge is ready within a predetermined period of time "t". The predetermined period of time "t" is set to be a relatively long time in the order of 40 clocks or thereabouts. In other words, the period of time "t" corresponds to several thousand clock cycles in a computer system operating at 25MHz, for example. For normal PCI bus exchanges, one would expect an IRDY# or TRDY# signal to be answered within 16 or 32 clock cycles. The predetermined period of time "t" may be adjusted to take account of different clock speeds and operating requirements.

In the state shown in Figure 30a, and if nothing was done to remedy this situation, an error might occur as data exchanges behind the depicted data exchange in the queue would back-up until a point is reached at which the system becomes deadlocked and the processing set fails.

Figures 30c and 30d illustrate, in a similar manner to Figures 31a and 31b, a correctly operating PCI bus (Figure 30d) and a PCI bus which appears to have encountered problems (Figure 30c). Figures 30c and 30d, however, differ from Figures 31a and 31b in that they depict DMA from the bridge to the processing set.

For both sets of figures, Figures 30a, 30b and 30c and 30d, it is the clock signal CLOCK that determines the point in time at which the various signals are generated on the bus. For example, as CLOCK is driven high in Figure 30a, the system becomes deadlocked and the processing set fails.

The IRDY# and TRDY# signals are asserted at the same clock edge (i.e. at time "b"), the initiator transfers a datum to the target.
30a (at the broken line in Figure 30a), the initiator is triggered to generate an IRDY# signal on the bus. As the clock signal goes high a predetermined period of time "t" later, so the mechanism is again triggered to determine that an error has occurred.

[0200] We have described above a faulty bridge, but it will be appreciated that a similar situation can occur where a processing set is faulty.

[0201] Returning to Figure 29, there is shown a mechanism 800 by means of which the problematic situation shown in Figures 30a and 30c may be alleviated. As shown in Figure 29, the mechanism 800 comprises a controller 810, a tri-state buffer 820 and a switchable field effect transistor (FET) 830 connected to the bus 24.

[0202] The controller 810 includes a counter which counts the number of clock signals between the issuance of an IRDY# signal and the issuance of a TRDY# signal or between a TRDY# signal and an IRDY# signal where the order is reversed. If the number of counted clock cycles exceeds a predetermined limit, then the controller 810 needs to turn off FET 830 to turn off the PCI bus 24 between the processing set 14, 16 and the bridge 12 before it controls the buffer 820 to abort the requested data transfer to thereby to prevent further data transfer requests. Issuance of TRDY# and IRDY# signals are detected on a sense line 825 connected between the controller 810 and the PCI bus 24.

[0203] If the controller 810 should detect the issuance of a TRDY# signal before the number of counted clock cycles has exceeded the predetermined limit, then the counter is reset and the PCI bus is left on.

[0204] The abortion of the requested data transfer is required as the initiator will expect to receive a TRDY# signal from the target, and thus could crash as data bus transactions back-up behind the request which has not been answered by the selected target. To abort the transaction from the initiator's point of view, the controller 810 controls the tristate buffer 820 by means of a control line 840 to issue a fake response signal to the initiator, whereupon the initiator is satisfied and outputs data that is discarded.

[0205] Figure 29 shows an arrangement wherein it is assumed that the bridge 12 is malfunctioning, and that the processing set 14, 16 is functioning correctly. In the implementation of Figure 29, the malfunctioning bridge is isolated from the processing set and cycles from the processing set 14, 16 are aborted in the address phase because the processing set does not see DEVSEL# asserted. However, it will be appreciated that for other bus protocols the processing set may need to continue to abort bus cycles.

[0206] Figure 31 is a schematic representation of the signals generated on the bus when a transaction is terminated by the mechanism 800. Figure 31 shows a bus where an IRDY# signal has issued during PIO and has not been responded to within the predetermined time limit "t" shown in Figure 30a. At some point in time after it has been determined that an error has occurred (due to the lack of a response to the issued IRDY# signal), an ISOLATE# signal is generated by the controller 810 to turn off the FET 830 thereby isolating the processing set from the bus. The controller 810 then issues a FAKE_ENABLE# signal to cancel the DEVSEL# signal issued during the address phase (i.e. the handshaking phase) between the target and initiator. Issuance of the FAKE_ENABLE# signal also causes a STOP# signal to be generated on the bus. The STOP# signal indicates to the system that the bus cycle has been stopped and serves to cause a FRAME# signal and the issued IRDY# signal to be driven high. The FRAME# signal is one of the basic components of the illustrative PCI protocol described herein and is used to indicate when an access (either PIO or DMA) is occurring. DEVSEL# is driven high and STOP# low which signals a target-abort to the initiator which then negates FRAME# and IRDY# in response.

[0207] It should be noted that the arrangement of Figure 31 is purely illustrative and that the FAKE_ENABLE# and ISOLATE# signals may be separated by a greater or lesser number of clock cycles than the number shown. It may also be necessary for a bus hold circuit to be provided to hold DEVSEL# low until the FAKE_ENABLE# signal is asserted.

[0208] An alternative to the arrangement of Figure 29 is shown in Figure 32. The arrangement of Figure 32 operates in the same manner as the arrangement of Figure 29, except that it is assumed that the bridge is functioning correctly and that it is the processing set 14, 16 which is malfunctioning. In this arrangement, therefore, a FET 850 is positioned between a processing set controller 50 and a controller 870. In this instance, therefore, signals from the malfunctioning processing set are ignored and signals from the bridge 12 are answered with a fake response issued by the controller 870 and a buffer 880.

[0209] A further implementation providing a higher degree of tolerance is shown in Figure 33. As shown in Figure 33, a FET 890 is provided between two controllers 900 and 910 and the controllers are operable, in the same way as the arrangements of Figures 29 and 32 to answer signals from the bridge and processing set with a fake response.

[0210] As mentioned above, it will be appreciated that the PCI bus 24 will normally comprise a number of bus lines (as shown in Figure 3 for example). Accordingly, the arrangement described above with reference to Figures 29, 32 and 33 could enable the generation of fake signals for all relevant bus lines of the PCI bus. In an alternative arrangement, an appropriate mechanism could be provided for each bus line of the PCI bus between the processing set 14, 16 and bridge 12.

[0211] Thus there has been described a bus control mechanism for a computer system that includes a bus, a first component and a second component, wherein the first and second components are interconnected via the bus for
performing a data transfer operation, the data transfer operation being initiated by an exchange of request and response signals, and a component which initiated a request signal is operable to effect data transfer on receipt of a response signal, the bus control mechanism comprising: first means for selectively disabling the bus: second means for generating a fake response signal; and third means for monitoring the request and response signals exchanged between the components, and for controlling the first means to disable the bus and for controlling the second means to issue a fake response signal to the component that issued the request signal for terminating the data transfer operation in situations where the response signal is not issued within a predetermined period following the request signal.

It will be appreciated that although particular embodiments of the invention have been described, many modifications/additions and/or substitutions may be made within the scope of the present invention. For example, although in the specific description two processing sets are provided, it will be appreciated that the specifically described features may be modified to provide for three or more processing sets. Also, it will be appreciated that bus isolation mechanisms such as those described herein can be used for bus protocols other a PCI bus protocol as is used in the particularly described embodiment.

Claims

1. A bus control mechanism for a computer system that includes a bus (24), a first component (50) and a second component (12), wherein the first and second components are interconnected via the bus for performing a data transfer operation, the data transfer operation being initiated by an exchange of request and response signals between the first and second components and a component that issued a request signal is operable to effect data transfer on receipt of a response signal, the bus control mechanism comprising:

   - a switch (830) selectively operable to disable the bus;
   - a fake response generator (820) selectively operable to generate a fake response signal; and
   - a controller (810) operable to monitor (825) the request and response signals exchanged between the components and, in situations where a corresponding response signal is not issued within a predetermined time following a particular request signal, to cause the switch to disable the bus and to cause the fake response generator to issue a fake response signal to the component that issued the particular request signal for terminating the data transfer operation.

2. The bus control mechanism of claim 1, wherein the component that issued the particular request signal is operable, on receipt of the fake response signal, to transfer data to the bus, which data is thus discarded as a result of the disabling of the bus by the switch.

3. The bus control mechanism of claim 1 or claim 2, wherein the computer system includes a clock signal generator and the controller comprises a counter for counting clock signals between detection of the particular request signal and detection of the corresponding response signal, the controller being operable, in the absence of detecting the corresponding response signal within a predetermined number of clock cycles, to cause the switch to disable the bus and to cause the fake response generator to issue a fake response signal to the component that issued the particular request signal for terminating the data transfer operation.

4. The bus control mechanism of any preceding claim, wherein the switch is selectively operable to disable the bus by isolating a first part of the bus connected to the first component from a second part of the bus connected to the second component.

5. The bus control mechanism of claim 4, wherein:

   - the fake response generator is connected to the first part of the bus; and
   - the controller is connected to the switch and to the fake response generator and is operable, in response to detection of the particular request signal from the first component and the absence of the corresponding response signal from the second component within the predetermined time, to cause the switch to disable the bus by isolating the first part of the bus from the second part of the bus and to cause the fake response generator to assert a fake response signal on the first part of the bus for causing the first component to terminate the data transfer operation.

6. The bus control mechanism of claim 5, wherein:
a second fake response generator is connected to the second part of the bus; and
a second controller is connected to the switch and to the further fake response generator and is operable, in
response to detection of a given request signal from the second component and the absence of a resulting
response signal from the first component within a predetermined period, to cause the switch to disable the
bus by isolating the first part of the bus from the second part of the bus and to cause the second fake response
generator to assert a fake response signal on the second part of the bus for causing the second component
to terminate the data transfer operation.

7. The bus control mechanism of claim 4, wherein the switch comprises an FET.

8. The bus controller of any preceding claim, wherein the request signal is an initiate transfer signal, the response
signal is a target ready signal, and the fake response signal is a fake target ready signal.

9. The bus control mechanism of any preceding claim, wherein the bus is a PCI bus.

10. The bus controller of claim 9, wherein the request signal is an IRDY# signal and the response signal is a TRDY#
signal.

11. The bus control mechanism of claim 1, comprising:
a first buffer selectively operable to assert a fake response signal on a first part of the bus;
a second buffer selectively operable to assert a fake response signal on a second part of the bus;
a first controller connected to the first part of the bus and operable to control the switch and the first buffer in
situations where the first component issues the request signal; and
a second controller connected to the second part of the bus and operable to control the switch and the second
buffer in situations where the second component issues the request signal.

12. The bus control mechanism of claim 11, wherein the switch is selectively operable to disable the bus by isolating
the first part of the bus connected to the first component from the second part of the bus connected to the second
component.

13. The bus control mechanism of claim 12, wherein at least one of the components is operable as an initiator to assert
an initiate transfer signal on the bus, and at least the other of the components is operable as a target to assert a
target ready signal, and the controller is operable to sense the initiate transfer signal and the target ready signal,
the controller then being operable to determine a timed period after sensing the initiate transfer signal and being
connected to the switch and to the fake response generator to cause the switch to isolate a part of the bus and to
cause the fake response generator to assert the fake response signal to the initiator for causing termination of the
data transfer operation in the absence of sensing the target ready signal within the timed period.

14. A computer system comprising:
a bus,
a first component and a second component interconnected via the bus for performing a data transfer operation,
the data transfer operation being initiated by an exchange of request and response signals, wherein a com-
ponent that initiates a request signal is operable to effect data transfer upon receipt of a response signal; and
a bus control mechanism according to any preceding claim.

15. A computer system according to claim 14, wherein the computer system is a fault tolerant computer system, and
wherein the first component is a processing set comprising at least one processor, and the second component is
a bus bridge.

16. A method of controlling a bus of a computer system including a first component and a second component inter-
connected via the bus for performing a data transfer operation, wherein the data transfer operation is initiated by
an exchange of request and response signals, wherein a component which initiated a requested signal is operable
to effect data transfer upon receipt of a response signal, the method comprising:
monitoring the request signal on the bus;
timing a period following the request signal;
in the absence of a corresponding response signal within the period, disabling the bus and issuing a fake response signal to the component which initiated the request to thereby terminate the data transfer operation.

17. The method of claim 16, wherein the step of disabling the bus comprises causing a switch to isolate a part of bus connected to a corresponding component that should have issued the response signal within the period.

18. The method of claim 16 or claim 17, wherein the bus is a PCI bus, the request signal is an IRDY# signal and the response signal is TRDY# signal.

Patentansprüche

1. Bussteuerungsmechanismus für ein Computersystem, der einen Bus (24), eine erste Komponente (50) und eine zweite Komponente (12) beinhaltet, wobei die erste und zweite Komponente zur Durchführung einer Datenübertragungsoperation über den Bus miteinander verbunden sind, wobei die Datenübertragungsoperation durch einen Austausch von Anforderungs- und Antwortenignalen zwischen der ersten und zweiten Komponente initiiert wird, und wobei eine Komponente, die ein Anforderungssignal ausgibt, derart betreibbar ist, daß sie den Datentransfer auf den Erhalt eines Antwortenignales hin bewirkt, wobei der Bussteuerungsmechanismus aufweist:

   einen Schalter (830), der wahlweise so betreibbar ist, daß er den Bus deaktiviert,
   einen Generator (820) von imitierten Antworten, der wahlweise derart betrieben werden kann, daß er ein imitiertes bzw. vorgetäuschtes Antwortenignal erzeugt, und
   einen Controller (810), der derart betrieben werden kann, daß er die Anforderungs- und Antwortenignale, die zwischen den Komponenten ausgetauscht werden, überwacht (825) und in Situationen, in denen ein entsprechendes Antwortenignal innerhalb einer vorbestimmten Zeit nach einem besonderen Anforderungssignal nicht ausgegeben wird, veranlaßt, daß der Schalter den Bus deaktiviert, und veranlaßt, daß der Generator von imitierten Antworten ein imitiertes Antwortenignal zu der Komponente ausgibt, die das besondere Anforderungssignal ausgegeben hat, um die Datenübertragungsoperation zu beenden.

2. Bussteuerungsmechanismus nach Anspruch 1, in der die Komponente, die das besondere Anforderungssignal ausgegeben hat, derart betreibbar ist, daß sie mit Erhalt des imitierten Antwortenignales Daten zu dem Bus überträgt, wobei die Daten somit als Ergebnis des Deaktivierens des Busses durch den Schalter abgelegt werden.


5. Bussteuerungsmechanismus nach Anspruch 4, in dem:

   der Generator für die imitierte Antwort mit dem ersten Teil des Busses verbunden ist und
   der Controller mit dem Schalter und dem Generator für die imitierte Antwort verbunden ist und derart betreibbar ist, daß er in Antwort auf die Erfassung des besonderen Anforderungssignales von der ersten Komponente und auf das Ausbleiben des entsprechenden Antwortenignales von der zweiten Komponente innerhalb der vorbestimmten Zeit, den Schalter veranlaßt, den Bus durch Trennen des ersten Teiles des Busses von dem zweiten Teil des Busses zu deaktivieren, und zu veranlassen, daß der Generator für die imitierte Antwort ein imitiertes Antwortenignal auf dem ersten Teil des Busses ausgibt, um die erste Komponente zu veranlassen, die Datenübertragungsoperation zu beenden.

6. Bussteuerungsmechanismus nach Anspruch 5, in dem:
ein zweiter Generator für die imitierte Antwort mit dem zweiten Teil des Busses verbunden ist und
ein zweiter Controller mit dem Schalter und dem weiteren Generator für die imitierte Antwort verbunden ist
und derart betreibbar ist, daß er in Antwort auf die Erfassung eines gegebenen Anforderungssignales von der
zweiten Komponente und das Ausbleiben eines resultierenden Antwortsignals von der ersten Komponente
innerhalb einer vorbestimmten Periode den Schalter veranlaßt, den Bus durch Trennen des ersten Teiles des
Busses von dem zweiten Teil des Busses zu deaktivieren, und zu veranlassen, daß der zweite Generator für
die imitierte Antwort ein imitiertes Antwortsignal auf dem zweiten Teil des Busses ausg, um die zweite Kom-
ponente zu veranlassen, die Datenübertragungsoperation zu beenden.


8. Buscontroller nach einem der vorherigen Ansprüche, in dem das Anforderungssignal ein Signal für die Initiierung
der Übertragung ist, das Antwortsignal ein Signal für die Bereitschaft des Ziels ist und das imitierte Antwortsignal
ein imitiertes Signal für die Bereitschaft des Ziels ist.


10. Buscontroller nach Anspruch 9, in dem das Anforderungssignal ein IRDY#-Signal und das Antwortsignal ein
TRDY#-Signal ist.

11. Bussteuerungsmechanismus nach Anspruch 1, der aufweist:

- einen ersten Pufferspeicher, der derart wahlweise betrieben werden kann, daß er ein imitiertes Antwortsignal
  auf einem ersten Teil des Busses ausgeben kann,
- einen zweiten Pufferspeicher, der wahlweise derart betrieben werden kann, daß er ein imitiertes Antwortsignal
  auf einem zweiten Teil des Busses ausgeben kann,
- einen ersten Controller, der mit dem ersten Teil des Busses verbunden ist und derart betreibbar ist, daß er
  den Schalter und den ersten Pufferspeicher in Situationen steuert, in denen die erste Komponente das An-
  forderungssignal ausgibt, und
- einen zweiten Controller, der mit dem zweiten Teil des Busses verbunden ist, und derart betreibbar ist, daß
  er den Schalter und den zweiten Pufferspeicher in Situationen steuert, in denen die zweite Komponente das
  Anforderungssignal ausgibt.

12. Bussteuerungsmechanismus nach Anspruch 11, in dem der Schalter wahlweise derart betreibbar ist, daß er den
Bus durch Trennen des ersten Teiles des Busses, der mit der ersten Komponente verbunden ist, von dem zweiten
Teil des Busses, der mit der zweiten Komponente verbunden ist, deaktiviert.

13. Bussteuerungsmechanismus nach Anspruch 12, in dem zumindest eine Komponente als ein Initiator betreibbar
ist, um ein Signal für die Initiierung der Übertragung auf dem Bus geltend zu machen, und zumindest die andere
Komponente als Ziel betreibbar ist, um ein Signal für die Bereitschaft des Ziels geltend zu machen, und wobei
der Controller derart betreibbar ist, daß er das Signal für die Initiierung der Übertragung und das Signal für die
Bereitschaft des Ziels erfaßt, wobei der Controller dann so betreibbar ist, daß er eine zeitlich abgestimmte Periode
nach der Erfassung des Signales für die Initiierung der Übertragung bestimmt und mit dem Schalter und dem
Generator für die imitierte Anwort verbunden ist, um den Schalter zu veranlassen, einen Teil des Busses zu iso-
lösen, und den Generator für die imitierte Antwort zu veranlassen, ein imitiertes Antwortsignal zu dem Initiator
auszugeben für die Veranlassung der Beendigung der Datenübertragungsoperation beim Ausbleiben der Erfas-
sung des Signales für die Bereitschaft des Ziels innerhalb der zeitlich abgestimmten Periode.

14. Computersystem, das aufweist:

- einen Bus,
eine erste Komponente und eine zweite Komponente, die über den Bus miteinander verbunden sind, für das
  Durchführen einer Datenübertragungsoperation, wobei die Datenübertragungsoperation durch einen Aus-
tausch von Anforderungs- und Antwortsignalen initiiert wird, wobei eine Komponente, die ein Anforderungs-
signal initiiert, derart betreibbar ist, daß sie die Datenübertragung mit Erhalt eines Antwortsignales bewirkt, und
- einen Bussteuerungsmechanismus nach einem der vorherigen Ansprüche.

15. Computersystem nach Anspruch 14, in dem das Computersystem ein fehlertolerantes Computersystem ist, und
in dem die erste Komponente eine Verarbeitungsgruppe ist, die zumindest einen Prozessor aufweist, und die zweite Komponente eine Busbrücke ist.

16. Verfahren zum Steuern eines Busses eines Computersystemes, das eine erste Komponente und eine zweite Komponente umfaßt, die über den Bus miteinander verbunden sind, für das Durchführen einer Datenübertragungsfunktion, wobei die Datenübertragungsfunktion durch einen Austausch von Anforderungs- und Antwortsignalen initiiert wird, in dem eine Komponente, die ein Anforderungssignal initiiert, derart betreibbar ist, daß sie mit Erhalt eines Antwortsignalen die Datenübertragung bewirkt, wobei das Verfahren aufweist:

  Überwachen des Anforderungssignales auf dem Bus,
  Einstellen einer Zeitperiode, die auf das Anforderungssignal folgt,
  beim Ausbleiben eines entsprechenden Antwortsignales innerhalb der Periode Deaktivieren des Busses und
  Ausgeben eines imitierten Antwortsignales an die Komponente, die die Anforderung initiierte, um hierdurch
die Datentransferfunktion zu beenden.

17. Verfahren nach Anspruch 16, in dem der Schritt des Deaktivierens das Veranlassen umfaßt, daß ein Schalter
   einen Teil des Busses, der mit einer entsprechenden Komponente, die das Antwortsignal innerhalb der Periode
   ausgegeben haben sollte, isoliert.

18. Verfahren nach Anspruch 16 oder Anspruch 17, in dem der Bus ein PCI-Bus ist, das Anforderungssignal ein IRDY#-
   Signal ist und das Antwortsignal ein TRDY#-Signal ist.

Revendications

1. Mécanisme de commande de bus pour un système informatique qui comprend un bus (24), un premier composant
   (50) et un second composant (12), dans lequel les premier et second composants sont interconnectés par le bus
   pour effectuer une opération de transfert de données, l’opération de transfert de données étant amorcée par un
   échange de signaux de demande et de réponse entre les premier et second composants et un composant qui a
   délivré un signal de demande peut être mis en oeuvre pour effectuer un transfert de données à la réception d’un
   signal de réponse, le mécanisme de commande de bus comportant :

   un commutateur (830) pouvant être actionné sélectivement pour invalider le bus ;
   un générateur (820) de réponse falsifiée pouvant être actionné sélectivement pour générer un signal de ré-
   ponse falsifiée ; et
   un régisseur (810) pouvant fonctionner de façon à contrôler (825) les signaux de demande et de réponse
   échangés entre les composants et, dans des situations dans lesquelles un signal de réponse correspondant
   n’est pas délivré dans un temps prédéterminé suivant un signal de demande particulier, pour amener le com-
   mutateur à invalider le bus et amener le générateur de réponse falsifiée à délivrer un signal de réponse falsifiée
   au composant qui a délivré le signal de demande particulier pour faire cesser l’opération de transfert de don-
   nées.

2. Mécanisme de commande de bus selon la revendication 1, dans lequel le composant qui a délivré le signal de
   demande particulier peut être mis en oeuvre, à la réception du signal de réponse falsifiée, de façon à transférer
   des données au bus, lesquelles données sont ainsi rejetées comme étant un résultat de l’invalidation du bus par
   le commutateur.

3. Mécanisme de commande de bus selon la revendication 1 ou la revendication 2, dans lequel le système informa-
   tique comprend un générateur de signal d’horloge et le régisseur comporte un compteur destiné à compter des
   signaux d’horloge entre la détection du signal de demande particulier et la détection du signal de répon’se corres-
   pondant, le régisseur pouvant être mis en oeuvre, en l’absence d’une détection du signal de réponse correspondant
   dans un nombre prédéterminé de cycles d’horloge, pour amener le commutateur à invalider le bus et à amener le
   générateur de réponse falsifiée à délivrer un signal de réponse falsifiée au composant qui a délivré le signal de
   demande particulier pour faire cesser l’opération de transfert de données.

4. Mécanisme de commande de bus selon l’une quelconque des revendications précédentes, dans lequel le com-
   mutateur peut être actionné sélectivement pour invalider le bus en isolant une première partie du bus connectée
   au premier composant d’une seconde partie du bus connectée au second composant.

23
5. Mécanisme de commande de bus selon la revendication 4, dans lequel :

le générateur de réponse falsifiée est connecté à la première partie du bus ; et
le régisseur est connecté au commutateur et au générateur de réponse falsifiée et peut être mis en oeuvre,
en réponse à la détection du signal de demande particulier provenant du premier constituant et de l'absence
du signal de réponse correspondant provenant du second constituant dans le temps prédéterminé, pour ame-
ner le commutateur à invalider le bus en isolant la première partie du bus de la seconde partie du bus et
amener le générateur de réponse falsifiée à imposer un signal de réponse falsifiée sur la première partie du
bus pour amener le premier composant à arrêter l'opération de transfert de données.

6. Mécanisme de commande de bus selon la revendication 5, dans lequel :

un second générateur de réponse falsifiée est connecté à la seconde partie du bus ; et
un second régisseur est connecté au commutateur et à l'autre générateur de réponse falsifiée et peut être
mis 'en oeuvre, en réponse à la détection d'un signal de demande donné provenant du second composant et
en l'absence d'un signal de réponse résultant provenant du premier composant dans une période prédéter-
minée, à amener le commutateur à invalider le bus en isolant la première partie du bus de la seconde partie
du bus et à amener le second générateur de réponse falsifiée à imposer un signal de réponse falsifiée sur la
seconde partie du bus pour amener le second composant à arrêter l'opération de transfert de données.

7. Mécanisme de commande de bus selon la revendication 4, dans lequel le commutateur comprend un transistor
du type FET.

8. Régisseur de bus selon l'une quelconque des revendications précédentes, dans lequel le signal de demande est
un signal de transfert d'initialisation, le signal de réponse est un signal de cible prête, et le signal de réponse
falsifiée est un signal falsifié de cible prête.

9. Mécanisme de commande de bus selon l'une quelconque des revendications précédentes, dans lequel le bus est
un bus PCI.

10. Contrôleur de bus selon la revendication 9, dans lequel le signal de demande est un signal IRDY# et le signal de
réponse est un signal TRDY#.

11. Mécanisme de commande de bus selon la revendication 1, comportant :

un premier tampon pouvant être mis en oeuvre sélectivement pour imposer un signal de réponse falsifiée sur
une première partie du bus ;
un second tampon pouvant être mis en oeuvre sélectivement pour imposer un signal de réponse falsifiée sur
une seconde partie du bus ;
un premier régisseur connecté à la première partie du bus et pouvant fonctionner de façon à commander le
commutateur et le premier tampon dans des situations dans lesquelles le premier composant délivre le signal
de demande ; et
un second régisseur connecté à la seconde partie du bus et pouvant fonctionner de façon à commander le
commutateur et le second tampon dans des situations dans lesquelles le second composant délivre le signal
de demande.

12. Mécanisme de commande de bus selon la revendication 11, dans lequel le commutateur peut être mis en oeuvre
sélectivement pour invalider le bus en isolant la première partie du bus connectée au premier composant de la
seconde partie du bus connectée au second composant.

13. Mécanisme de commande de bus selon la revendication 12, dans lequel au moins l'un des composants peut être
mis en oeuvre en tant qu'initiateur pour imposer un signal de transfert d'initialisation sur le bus, et au moins l'autre
des composants peut être mis en oeuvre en tant que cible pour imposer un signal de cible prête, et le régisseur
peut être mis en oeuvre pour détecter le signal de transfert d'initialisation et le signal de cible prête, le régisseur
pouvant ensuite être mis en oeuvre pour déterminer une période minutée après la détection du signal de transfert
d'initialisation et étant connecté au commutateur et au générateur de réponse falsifiée pour amener le commutateur
to isoler une partie du bus et amener le générateur de réponse falsifiée à imposer le signal de réponse falsifiée à
l'initiateur pour provoquer la cessation de l'opération de transfert de données en l'absence de la détection du signal
14. Système informatique comportant :

- un bus,
- un premier composant et un second composant interconnectés par le bus pour effectuer une opération de transfert de données, l'opération de transfert de données étant amorcée par un échange de signaux de demande et de réponse, dans lequel un composant qui amorce un signal de demande peut être mis en œuvre pour effectuer un transfert de données à la réception d'un signal de réponse ; et
- un mécanisme de commande de bus selon l'une quelconque des revendications précédentes.

15. Système informatique selon la revendication 14, dans lequel le système informatique est un système informatique à tolérance de pannes, et dans lequel le premier composant est un ensemble de traitement comprenant au moins un processeur, et le second composant est un pont de bus.

16. Procédé de commande d'un bus d'un système informatique comprenant un premier composant et un second composant interconnectés par le bus pour effectuer une opération de transfert de données, dans lequel l'opération de transfert de données est amorcée par un échange de signaux de demande et de réponse, dans lequel un composant qui a amorcé un signal demandé peut être mis en œuvre pour effectuer un transfert de données à la réception d'un signal de réponse, le procédé comprenant :

- le contrôle du signal de demande sur le bus ;
- le minutage d'une période suivant le signal de demande ;
- en l'absence d'un signal de réponse correspondant dans la période, l'invalidation du bus et la délivrance d'un signal de réponse falsifiée au composant qui a amorcé la demande pour faire cesser ainsi l'opération de transfert de données.

17. Procédé selon la revendication 16, dans lequel l'étape d'invalidation du bus comprend le fait d'amener un commutateur à isoler une partie du bus connectée à un composant correspondant qui aurait dû délivrer le signal de réponse dans la période.

18. Procédé selon la revendication 16 ou la revendication 17, dans lequel le bus est un bus PCI, le signal de demande est un signal IRDY# et le signal de réponse est un signal TRDY#.
FIG. 2
FIG. 7
FIG. 11
LOCKSTEP ERROR S2

DATA PHASE? S31

YES

TERMINATE CYCLE S32

STORE DATA PHASES IN DISCONNECT REGISTERS S33

NO

STORE ADDRESS AND DATA PHASE IN POSTED WRITE BUFFER S34

NEXT I/O CYCLE? S35

YES

EVALUATE ERROR STATE S4

NO

FIG. 13
FIG. 14
FIG. 27

ARBITRATE
D BUS
S20

VERIFY
DMA
ADDRESS?
S21

FALSE

COMBINED
MODE?
S23

NO

YES

DMA REQUEST
TO BOTH
PROCESSING
SETS
S24

CHECK
OWNERSHIP
S25

A

B

DMA REQUEST
TO PROCESSING
SET A
S26

DMA REQUEST
TO PROCESSING
SET B
S27

IGNORE DMA
REQUEST
S22
PROCESSING SET
14, 16

BRIDGE MOTHERBOARD
42

FIG. 29
FIG. 30A

CLOCK
IRDY#
TRDY#

FIG. 30B

CLOCK
IRDY#
TRDY#

FIG. 30C

CLOCK
TRDY#
IRDY#

FIG. 30D
FIG. 32

PROCESSING SET 14, 16
BRIDGE MOTHERBOARD 42

PROCESSING SET BUS CONTROLLER 50

CONTROLLER 870

BRIDGE 12