PCI TO ISA INTERRUPT PROTOCOL CONVERTER AND SELECTION MECHANISM

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to computer system peripheral connections. More particularly, the present invention relates to interrupt handling for computer systems having multiple peripheral buses.

2. Reservation of Copyright

A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

3. Art Background

In the computer industry, one widely accepted system architecture for stand alone systems has been the AT system design. Computer system motherboards supporting this system type generally include a standardized input/output (I/O) bus for providing inter-system

communications between the various system components. For example, the system's central processing unit (CPU) will utilize this bus for retrieving and writing data to the system's hard disk drive or other storage devices. Such system bus standards include the well-known ISA and EISA standards to which, for example, a hard disk drive may be connected through an Integrated Drive Electronics (IDE) interface.

As computer processor technology has developed, the traditional ISA, EISA and other I/O bus standards have proven inadequate to serve computer systems with advanced processors. There has recently been introduced a new I/O bus standard for use as a primary I/O bus which provides a much greater bandwidth than previously existing bus standards. The Peripheral Component Interconnect (PCI) bus is quickly receiving wide acceptance in the computer industry. The PCI bus standard provides for a high bandwidth and a flexibility that is independent of new processor technologies and increased processor speed. At this time, computer system architects are primarily designing speed sensitive peripherals such as graphics accelerators and SCSI disk drive controllers to be utilized with the PCI bus.

Computer system's designed today that incorporate PCI bus capabilities generally also include a slower, secondary I/O bus such as the ones described above, for compatibility with numerous existing, previously designed peripheral devices. There are a number of differences between the PCI bus standard and the previous I/O bus standards, particularly with respect to the various interrupt protocols.

For example, the EISA/ISA bus architectures provides for 16 interrupt signals, IRQ[0:15] with each designated for different possible uses. The PCI bus on the other hand provides for four shared interrupts, INTR[A:D], with each shared by all slots on the PCI bus.

The different interrupt protocols for the two bus standards described above goes deeper than the number and naming of those designated interrupts. The greater difficulty in reconciling interrupts in a computer system architecture having both a PCI bus and ISA bus is that the PCI interrupts are defined to be active low, level sensitive interrupts while the ISA bus specification defines interrupts to be low-to-high, edge sensitive interrupts. The PC AT compatible interrupt structure, through a master/slave interrupt controller pair, supports sixteen active high edge sensitive interrupts. Out of these possible sixteen interrupts, an AT system dedicates five for bare minimum configurations, an additional six are used in a normal system environment having floppy and IDE drive support, two serial ports, a parallel port and PS/2 mouse support, leaving only five available for use with add-in cards.

The PCI bus specification requires four active low, level sensitive interrupt pins for each slot supported and defines their use as hardware shareable. That means that multiple PCI devices can drive the same interrupt line or that multiple PCI interrupt lines can be driven by different devices but may result in a single interrupt being generated to the system interrupt controller to be serviced by a shared interrupt driver.

Due to the small number of available interrupts in a PC AT system coupled with the addition of PCI based add-in cards, it can be seen that an interrupt sharing technique is clearly needed. It would be advantageous, and is therefore an object of the present invention, to provide a mechanism for integrating dissimilar interrupt protocols into a computer system for enhanced computer system operation.

SUMMARY OF THE PRESENT INVENTION

From the foregoing it can be appreciated that a computer system's performance may be enhanced by providing a mechanism for handling interrupts generated according to differing interrupt protocols. Accordingly, it is an object of the present invention to provide a method and apparatus for integrating interrupts generated in accordance with dissimilar interrupt protocols. Particularly, it is an object of the present invention to provide a level-to-edge conversion mechanism for integrating interrupts compliant with the PCI bus standard in a computer system having a secondary I/O bus and already equipped for handling interrupts generated by agents on the secondary I/O bus. These and other objects of the present invention are provided by introducing interrupt conversion circuitry for processing interrupts generated by agents on the PCI bus.

In one embodiment of the present invention, a computer system is implemented having a PCI bus as a primary I/O bus and an ISA bus as a secondary I/O bus. A programmable logic device is coupled to receive interrupts generated by agents on the PCI bus. The programmable logic device serves to convert the active low, level sensitive interrupts compliant with the PCI standard to the low-to-high, edge sensitive interrupts defined by the ISA bus specification. PCI agent interrupts, once converted to the appropriate interrupt protocol, are then provided to the system's resident interrupt controller for notifying the CPU of a pending interrupt.

One embodiment of the present invention also includes a software (usually BIOS) writeable control register for selecting a particular ISA IRQ interrupt to provide to the system's interrupt controller. The control register provides select control lines to the interrupt conversion logic for selecting a desired IRQ line to be used in routing interrupts from the PCI bus to the interrupt controller. This facilitates the using of a single interrupt signal from the programmable logic device to the interrupt controller.

BRIEF DESCRIPTION OF THE DRAWINGS

The objects, features and advantages of the present invention will be apparent from the following detailed description in which:

FIGURE 1 illustrates a block diagram of a computer system architecture using a hierarchy of buses incorporating one embodiment of the present invention.

FIGURE 2 illustrates a logic diagram indicating the inputs and outputs to the programmable logic to be implemented in one embodiment of the present invention.

FIGURE 3 illustrates a state diagram to be implemented by the programmable logic for converting PCI interrupts to ISA compliant interrupts by the programmable logic implemented in one embodiment of the present invention.

FIGURE 4 illustrates another state diagram which may be implemented by the programmable logic for converting PCI interrupts to ISA compliant interrupts in another embodiment of the present invention.

DETAILED DESCRIPTION OF THE PRESENT INVENTION

A method and apparatus are provided for interrupt handling in a computer system having multiple I/O buses that follow differing interrupt protocols. In this detailed description, one embodiment of a computer system which includes both a PCI bus and an ISA bus is described. It should be understood that this is for the purpose of illustration and is not provided as a limitation in that the present invention may be practiced on other computer systems where it is desirable to utilize multiple buses that follow different interrupt protocols. Throughout this detailed description, numerous specific details are set forth such as particular signal names, bus protocols and logic device types, in order to provide a thorough understanding of the present invention. It will be appreciated by one having ordinary skill in the art that the present invention may be practiced without such specific details. In other instances, well-known components, structures and techniques have not been described in detail in order to avoid obscuring the subject matter of the present invention. In addition, signal names identified in this detailed description are not intended to imply the active state of a given signal (active high or active low) but are simply used to name interface signals for discussion.

Referring now to Fig. 1, a computer system architecture which may incorporate the present invention is illustrated. The computer system architecture of Fig. 1 is illustrated so as to highlight the hierarchical bus organization of the architecture. It can be seen that the

computer system's central processing unit (CPU) 10 communicates with its SRAM cache 15 over a Host bus 20. In one embodiment, the Host bus 20 includes control lines 21, address lines 22 and data lines 23. In other embodiments, the address and data lines may be multiplexed to share common signal paths. A Host bus is usually designed to meet the needs of high performance computing associated with whatever type of processor is implemented for CPU 10.

In some computer system architectures, the computer system's main memory will be directly coupled to the processor Host bus. In the illustrated implementation, the DRAM main memory 18 is coupled to the Host bus through the Host/PCI bus bridge 25. The Host/PCI bus bridge 25 is used as a bridge between the Host bus 20 and the system's PCI bus 30. The PCI bus, as described above, serves as the primary I/O bus for the computer system and was designed to address the growing industry needs for a standardized local bus that is not directly dependent on the speed and size of a computer system's processor bus. The PCI bus 30 as illustrated in **Fig. 1** is shown separated into control lines 31 and address/data lines 32. The Host/PCI bus bridge 25 monitors the Host bus 20 and the PCI bus lines 31 and 32, and determines whether a transaction is designated for the computer system's main memory system 18 or whether a CPU request should be propagated down the hierarchy of buses, the next step being down to the PCI bus 30.

There is shown coupled to the PCI bus 30 in **Fig. 1** three PCI slots 36, 37 and 38 for receiving peripheral devices designed to be compliant with the PCI bus standard. The computer system may, of course, employ any number of PCI slots and still incorporate the present invention. Such add-in cards may include graphics accelerators, disk drive controllers, and speed sensitive peripherals which may take advantage of the abilities of the PCI local bus 30. There is also shown coupled between the address/data lines 32 of the PCI bus and the Host bus a collection of buffers, denoted LBX buffers 34 which are used to buffer data transfers between the two buses 20 and 30. The LBX buffers 34 are implemented to compensate for the varying speeds that the two buses may be operating at and enhance system performance by allowing for concurrency between the Host bus 20 and PCI bus 30, giving each bus greater throughput and decreased latency. In alternative embodiments, the LBX buffers may be replaced by other buffering mechanisms so long as the data transferring functionality between the host bus, main memory and primary I/O bus is maintained.

There is finally shown in **Fig. 1**, a secondary I/O bus 40 which in the illustrated embodiment is a peripheral I/O bus compliant with the ISA bus standard. The teachings of the present invention may of course be extended to computer systems having other secondary I/O buses, such as an EISA bus or other bus having a different interrupt protocol than the PCI bus. Coupled to the ISA bus 40 are four I/O slots 41, 42, 43 and 44 which may be used for receiving various peripherals. On-board peripherals 45 such as floppy disk drives, hard disk drives equipped

with an IDE standard interconnect, and other peripherals normally associated with computer systems are often designed right on the system's motherboard and do not require an I/O slot which are provided for other add-in card upgrades. The on-board peripherals 45 do electrically reside on the ISA bus 40. In general, the I/O bus 40 is a slower bus than the PCI bus 30 but is still included in newer computer systems to maintain compatibility with previously designed peripherals.

Having both an ISA bus and a PCI bus allows personal computer platforms built around the PCI bus as a primary I/O bus to leverage the large ISA product base. The ISA bus provides 24-bit addressing and a 16-bit datapath. An EISA I/O bus allows a personal computer platform built around the PCI bus as a primary I/O bus to leverage the large EISA/ISA product base. Along with compatibility for 16-bit and 8-bit ISA hardware and software, the EISA bus provides for 32-bit addressing and a 32-bit datapath. Of course, other secondary I/O buses may provide similar features.

Coupled between the PCI bus 30 and the ISA bus 40 of the computer architecture of **Fig. 1** is illustrated a system I/O (SIO) component 35. The SIO component 35 provides the bridge between the PCI bus and the ISA bus and may incorporate the logic for interfacing the ISA bus 40 with the PCI bus 30 as well as incorporating support for any necessary DMA controllers and interrupt control logic. For example, if a traditional hard disk drive having an IDE interconnect is present in I/O slot 41 and that IDE drive requests a transaction with

the CPU, the IDE drive will signal an interrupt, IRQ 14, which will be recognized as a primary hard disk drive access by the system. When the computer system recognizes an IRQ 14 interrupt, it will vector to the necessary code for accessing the hard disk drive.

In the computer architecture implemented according to **Fig. 1**, the SIO component 35 includes interrupt controllers, such as the Intel 82C59 interrupt controller, configured to be responsive to interrupts compliant with ISA bus interrupts. The SIO component 35 will resolve interrupts from various sources through programmable arbitration schemes and present a single interrupt to the CPU to act to execute an appropriate interrupt service routine.

As described, agents residing in a PCI slot on the PCI bus share four PCI designated, level sensitive interrupts. In the computer architecture configuration illustrated in **Fig. 1**, each of the PCI slots is coupled so that PCI agents will provide their interrupts to a programmable logic device identified as interrupt PAL 50. The configuration and programming of the interrupt PAL 50 will be described in more detail further herein. Essentially, the function of the interrupt PAL 50 is to convert the active low, level sensitive PCI interrupts into the low-to-high, edge sensitive interrupts required by the ISA bus standard. The interrupt PAL 50 provides the edge sensitive interrupt to the SIO component 35 in response to a PCI interrupt requested by any and all of the PCI agents.

It was noted above that various IRQ signals may be utilized for signaling an ISA interrupt to the SIO component 35. There is thus provided the software (usually BIOS) writeable control register 48 which controls select lines to the interrupt PAL 50 for directing which single IRQ signal to provide to the SIO component 35. In one embodiment the select lines are controlled by an IO write to the lower 2 bits of IO address 73h. Other registers or latches may be used for the storage element 48 and may reside at different IO or memory locations. The internal operation of the control register 48 and the interrupt PAL 50 will be described in more detail further herein.

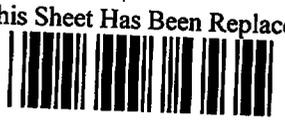
Referring now to Fig. 2, a more detailed diagram of the interrupt PAL 50 as utilized by one embodiment of the present invention is illustrated. In one embodiment of the present invention, the programmable logic may comprise an 85C220 EPLD manufactured by the Intel Corporation. Of course, other programmable logic capable of executing the functionality to be described below may prove suitable for other implementations. The PAL 50 receives all of the shared interrupts from the PCI agents coupled to the PCI bus 30. The combined shared interrupts are OR'ed together by gate logic 55 and the result is provided to state machine 60 by a signal denoted INT. Thus, any PCI agent signaling any interrupt will cause the INT signal to go active to the state machine 60, which is clocked by a PCICLK signal.

The state machine logic 60 in the programmable logic block 50, in addition to receiving the PCICLK timing signal and the logical OR of

the PCI agent interrupts, also receives the INTR signal from the SIO component 35, identified as INTA at the state machine side of the transaction and a system reset signal. In one embodiment, the EPLD is reset on pin 8 by the same signal that resets the entire ISA bus. The INTR signal supplied to the CPU from the SIO component 35 is the interrupt request generated by the SIO which causes the CPU to process an interrupt service routine (ISR). Upon completion of performance of an ISR, the CPU clears the INTR signal by writing an "end of interrupt" command to the SIO's interrupt controller interface.

The state machine logic 60 is also provided with the select control signals SEL0 and SEL1. In the illustrated embodiment, a PCI agent interrupt will be forwarded to the SIO component 35 (**Fig. 1**) through one of the IRQ lines: IRQ9, IRQ10 or IRQ11. The choice of which IRQ signal is provided to the SIO component 35 is determined by the bits set in the select register 48 (**Fig. 1**) which is writeable by the system's configuration software. Of course, other IRQ signals may be used for indicating a PCI interrupt to the SIO component 35. Note that if more than four possible IRQs are provided for, then more than two bits will be required to be set in the SEL register 48 and additional select lines to the PAL 50 will be required. The selected control bits, in addition to being provided to the state machine logic 60, are also provided to the ISA IRQ select multiplexer 52. When the state machine logic 60 sends an active IRQ signal as a control signal to the select multiplexer 52, the multiplexer 52 provides the selected IRQ signal to the SIO component 35 responsive to the select bits and the control

This Sheet Has Been Replaced



THIS SHEET HAS BEEN REPLACED

signal. In one embodiment, one state of these bits is used to encode that no IRQ output to the SIO should occur. This is the reset state of the SEL REG 48, so at RESET the bits must be configured or no IRQ will be processed from PCI agents.

Fig. 3 illustrates a state diagram to be implemented by one embodiment of the state machine logic 60 illustrated in **Fig. 2**. Attached as a Appendix A to this disclosure is a listing of one embodiment of a program for programming the programmable logic device 50 to implement the state diagram shown in **Fig. 3**. Note that each state identifies a three digit control code at its bottom. These correspond to the values of state variables B2, B1 and B0, respectively, in each state as they are utilized throughout the source code listing attached as Appendix A.

The state machine diagram begins at idle state 300 upon reset. The state machine in this example is synchronous, transitioning from state-to-state on a PCICLK rising edge. The state machine remains in the idle state as long as the INT signal is inactive, indicating that no PCI interrupts are being asserted by any PCI agents.

When the INT signal goes active to the state machine 60, it transitions to N1 state 310 on the next PCICLK input. Once the state machine has achieved N1 state 310, it asserts the IRQ active signal to the ISA IRQ select multiplexer 52 for providing the selected IRQ to the SIO component 35. If the INTA signal is already active when the state machine logic is in N1 state 310, the state machine will remain in the N1

state. An active INTA signal when the state machine first achieves the N1 state indicates that the CPU is currently executing a previously requested interrupt service routine. The state machine will not transition from the N1 state 310 to the N2 state 320 until the INTA signal goes inactive, indicating that the CPU has completed handling the previously signaled interrupt.

The state machine may transition from the N1 state 310 back to the idle state 300 if the INT signal goes inactive while in the N1 state. This may indicate that the PCI agent requesting the interrupt no longer requires the interrupt to be handled by the CPU. This is not to be expected but may occur. Additionally, in the illustrated embodiment of the present invention, if both the SEL1 and SEL0 select bits go inactive, the state machine will return to the idle state 300. This may indicate a selection made by the operating system that no PCI interrupts are to be acknowledged. In alternative embodiments, null SEL1 and SEL0 values may be defined to indicate a particular IRQ signal, or a default IRQ signal.

Once the CPU has cleared the INTA signal, if it was active, the state machine transitions from the N1 state 310 to the N2 state 320 on the next PCICLK input. In this state, the IRQ signal continues to be driven from the programmable logic device 50 so long as the INT signal to the state machine logic remains active or the select signals do not go inactive. The state machine remains in the N2 state 320 so long as the INTA signal remains inactive indicating that the CPU is not responding

to any interrupt request. When the INTA signal again goes active, it indicates that the CPU is responding to an interrupt which may be the interrupt generated by the PCI agent, though it is possible that the CPU is responding to a different interrupt. When the INTA signal does go active, the state machine transitions to N3 state 330 where it remains as long as the INTA signal remains active. The state machine continues to provide the IRQ signal to the ISA IRQ select multiplexer 52 while in the N3 state. A deactivation of the SEL1 and SEL0 select controls will cause the machine to return to the idle state 300, otherwise the state machine will remain in the N3 state 330 as long as the INTA signal is active.

When the INTA signal again goes inactive, it indicates that the CPU has completed execution of an interrupt service routine (ISR) which may have been the ISR required by the PCI agent signaling the interrupt, though not necessarily. The state machine will then transition through R1 state 340, R2 state 350 and R2 state 360, residing in each state for one cycle of the PCICLK timing signal. During this period, the IRQ signal will no longer be active and the SIO component 35 will not be receiving an IRQ from the interrupt PAL 50.

From the R3 state 360, the state machine returns to the idle state 300. If the CPU has just executed an ISR for the PCI agent, the procedure should have cleared the PCI agent's interrupt request, then it is likely that the INT signal will be inactive. However, if the CPU had been responding to a different interrupt when the state machine was in

N3 state 330, then it would have been a different ISR which caused the INTA signal to go inactive. In that event, it is expected that the INT signal is still an active input to the state machine 60, in which case the state machine will transition from the idle state 300 to the N1 state 310 to repeat the process described above on the next PCICLK input.

It should be understood that the series of states following the N3 state 330 through the R1, R2 and R3 states 340, 350 and 360, back through the idle state 300 and to the N1 state 310 provides a necessary delay. It is this delay and the switching off and back on of the IRQ signal from the interrupt PAL 50 that converts the PCI compliant level sensitive interrupt received by the interrupt PAL 50 to the low-to-high edge sensitive interrupt driven to the interrupt controller of the SIO component 35 as an ISA IRQ compliant signal.

Fig. 4 illustrates a more general state diagram to be implemented by one embodiment of state machine logic 60. The state diagram illustrated in **Fig. 4** includes the states 300, 310, 320 and 330 described above with respect to **Fig. 3**. In **Fig. 4**, the R1, R2 and R3 states are replaced by delay state 400. The specific implementation of **Fig. 3** required three PCI clock signals of delay before returning to the idle state 300 to ensure a proper placing of the high-to-low edge sensitive interrupt to the SIO component 35. In some embodiments, a different delay period may be required and is encompassed and accounted for by the delay state 400 incorporated in **Fig. 4**.

There has thus been described a mechanism for harmonizing dissimilar interrupt protocols in a computer system having more than one implemented I/O bus. Although the present invention has been described in terms of various embodiments, it will be appreciated by those skilled in the art that various modifications and alterations might be made without departing from the spirit and scope of the present invention. For example, it is possible to implement a programmable logic design using a clock other than the PCICLK. Similarly, an asynchronous implementation is conceivable. The invention should, therefore, be measured in terms of the claims which follow.



CLAIMS

What is claimed is:

1. A computer system comprising:

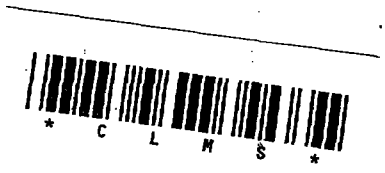
a central processing unit (CPU);

a host bus coupled to the CPU for conveying signals to and from the CPU;

an I/O bus coupled to communicate with the CPU over the host bus, the I/O bus being compliant with a first bus standard for providing a communications path between the computer system and peripheral devices, wherein interrupts propagated over the I/O bus are of a first interrupt type;

an interrupt controller coupled to communicate with the CPU, the interrupt controller being responsive to interrupts of a second interrupt type for signaling interrupts to the CPU;

an interrupt converter coupled to said interrupt controller and coupled to receive interrupts of said first interrupt type generated by agents of said I/O bus, said interrupt converter for providing an interrupt of the second interrupt type to the interrupt controller in response to receiving interrupt of the first interrupt type.



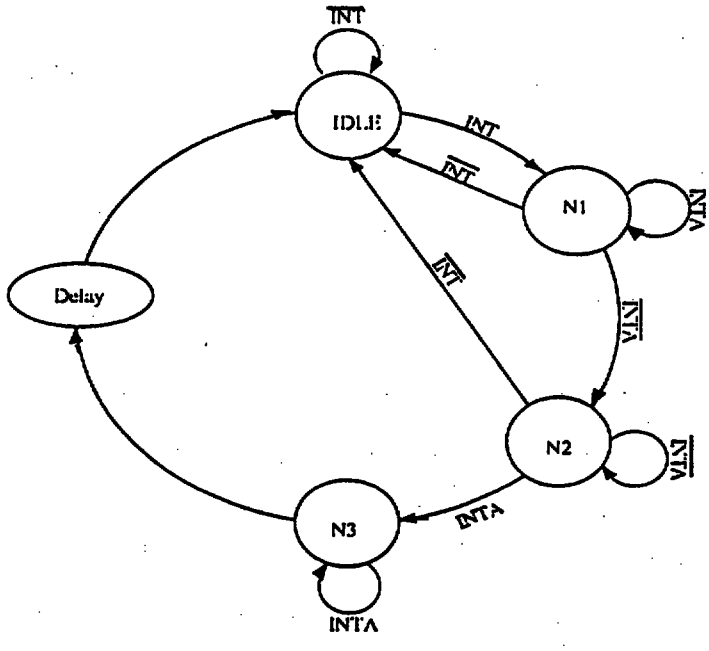
2. The computer system of claim 1 wherein interrupts of the first type are level sensitive interrupts and interrupts of the second type are edge sensitive interrupts.

3. The computer system of claim 2 wherein said interrupt converter comprises:

circuitry for performing a logical OR operation of the interrupt lines from all agents on the I/O bus; and

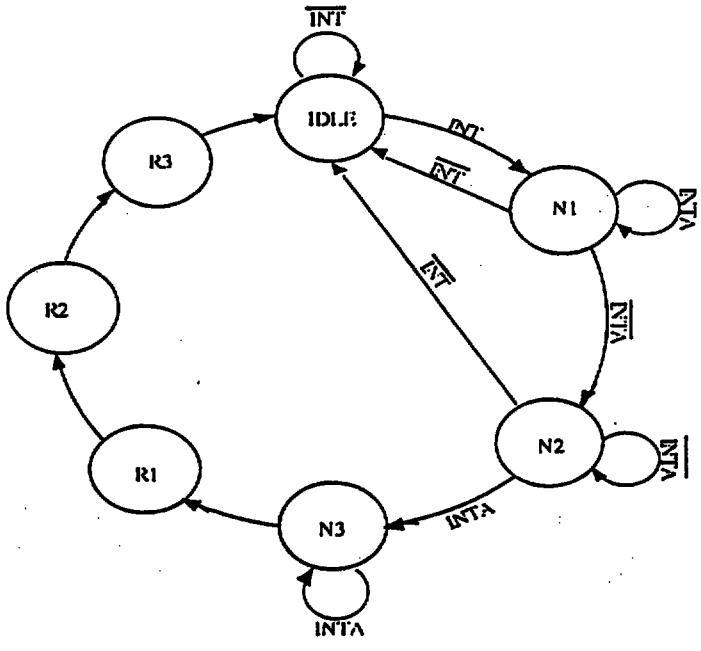
state machine circuitry coupled to and responsive to the circuitry for performing the logical OR operation, the state machine circuitry for outputting an edge sensitive interrupt signal.

4. The computer system of claim 3 wherein the state machine operates according to the following state diagram:



wherein when the state machine is in the N1, N2 and N3 states the state machine circuitry outputs an edge sensitive interrupt signal, when the state machine is in the Delay state the state machine circuitry outputs no interrupt signal, the INT signal comprising an active result of the logical OR of the interrupt lines from all agents on the I/O bus, the INTA signal comprising a signal from the CPU indicating that it is processing an interrupt.

5. The computer system of claim 3 wherein the state machine operates according to the following state diagram:



wherein when the state machine is in the N1, N2 and N3 states the state machine circuitry outputs an edge sensitive interrupt signal, the INT signal comprising an active result of the logical OR of the interrupt lines from all agents on the I/O bus, the INTA signal comprising a signal from the CPU indicating that it is processing an interrupt, the R1, R2 and R3 states comprising successive state machine states where the state machine circuitry outputs no interrupt signal.

6. The computer system of claim 3 wherein said interrupt converter further comprises multiplexing logic coupled to said state



machine circuitry for routing said edge sensitive interrupt signal to a selected interrupt input to said interrupt controller.

7. The computer system of claim 6 further comprising a programmable storage element coupled to said multiplexing logic for controlling said multiplexing logic to direct said edge sensitive interrupt to said selected interrupt input.

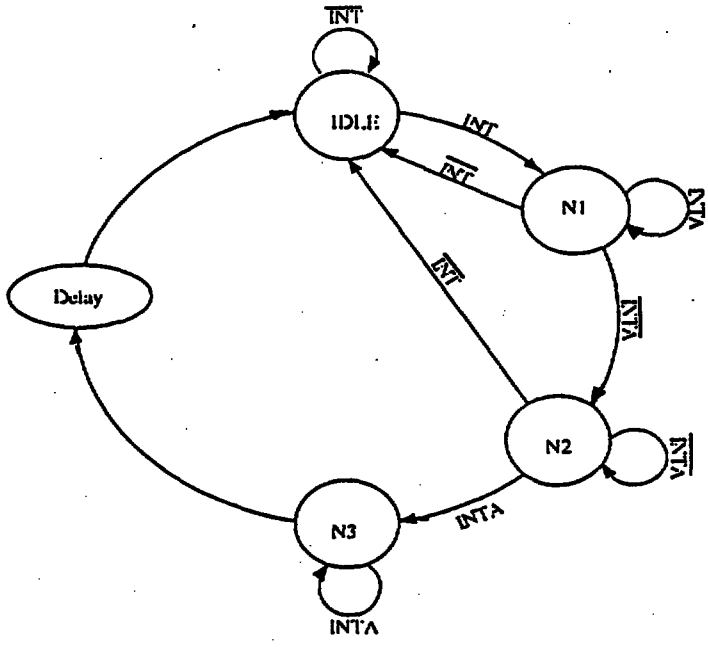
8. The computer system of claim 1 wherein said first bus standard comprises the Peripheral Component Interconnect (PCI) standard.

9. A level sensitive to edge sensitive interrupt converter for use in a computer system having a CPU and peripheral agents that generate level sensitive interrupts wherein said computer system incorporates an interrupt controller responsive to edge sensitive interrupts, the interrupt converter comprising:

circuitry for performing a logical OR operation of interrupt signals generated by all peripheral agents which generate level sensitive interrupt signals; and

logic circuitry coupled to and responsive to the circuitry for performing the logical OR operation for outputting an edge sensitive interrupt signal to said interrupt controller when a level sensitive interrupt signal is detected.

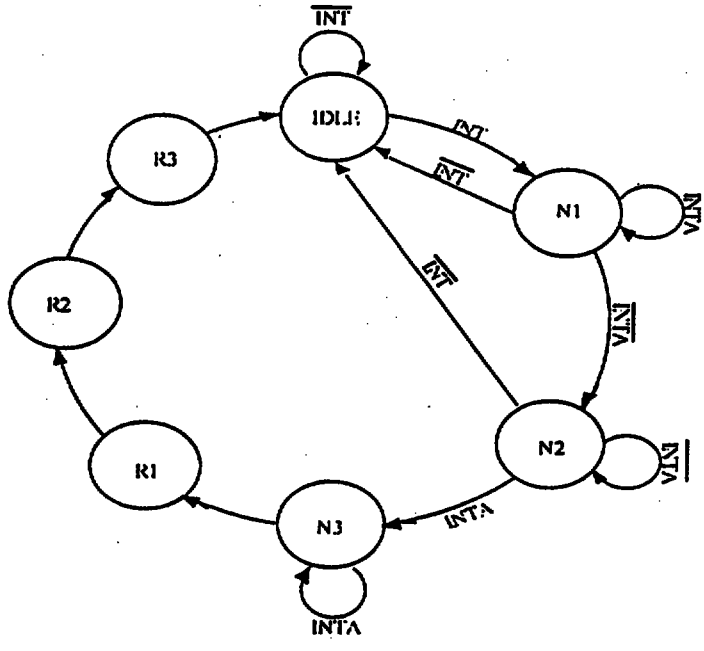
10. The interrupt converter of claim 9 wherein the logic circuitry operates according to the following state machine model:



wherein when the state machine is in the N1, N2 and N3 states the state machine circuitry outputs an edge sensitive interrupt signal, when the state machine is in the Delay state the state machine circuitry outputs no interrupt signal, the INT signal comprising an active result of the logical OR operation of the interrupt lines from all agents on the I/O

bus, the INTA signal comprising a signal from the CPU indicating that it is processing an interrupt.

11. The interrupt converter of claim 9 wherein the logic circuitry operates according to the following state machine model:



wherein when the state machine is in the N1, N2 and N3 states the state machine circuitry outputs an edge sensitive interrupt signal, the INT signal comprising an active result of the logical OR operation of the interrupt lines from all agents on the I/O bus, the INTA signal comprising a signal from the CPU indicating that it is processing an interrupt, the R1, R2 and R3 states comprising successive state machine states where the state machine circuitry outputs no interrupt signal.



12. The interrupt converter of claim 9 wherein said interrupt converter further comprises multiplexing logic coupled to said state machine circuitry for routing said edge sensitive interrupt signal to a selected interrupt input to said interrupt controller.

13. The interrupt converter of claim 12 further comprising a programmable storage element coupled to said multiplexing logic for controlling said multiplexing logic to direct said edge sensitive interrupt to said selected interrupt input.

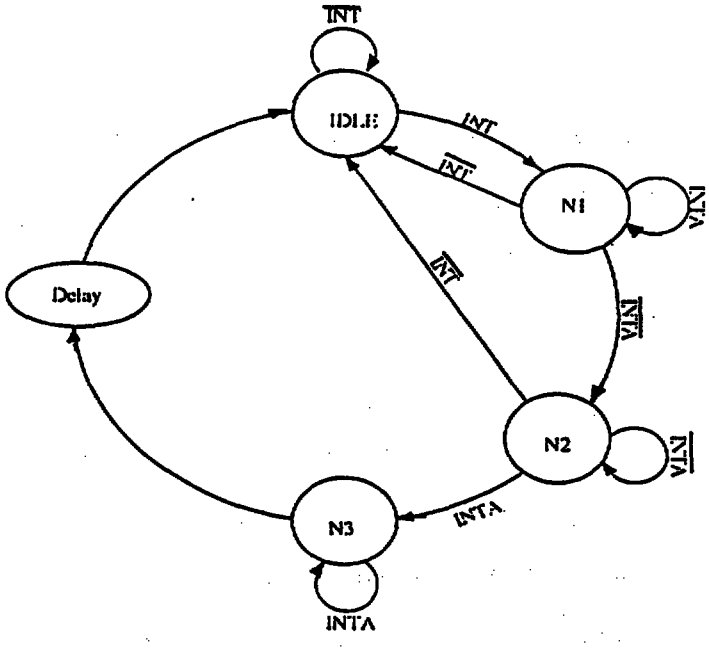
14. A method of processing interrupts in a computer system having a CPU and an interrupt controller responsive to interrupts of a first interrupt type wherein some peripheral agents of said computer system generate interrupts of a second interrupt type, said method comprising the steps of:

detecting interrupts of said second interrupt type;

generating an interrupt signal compliant with said first interrupt type responsive to a detected interrupt of said second interrupt type; and

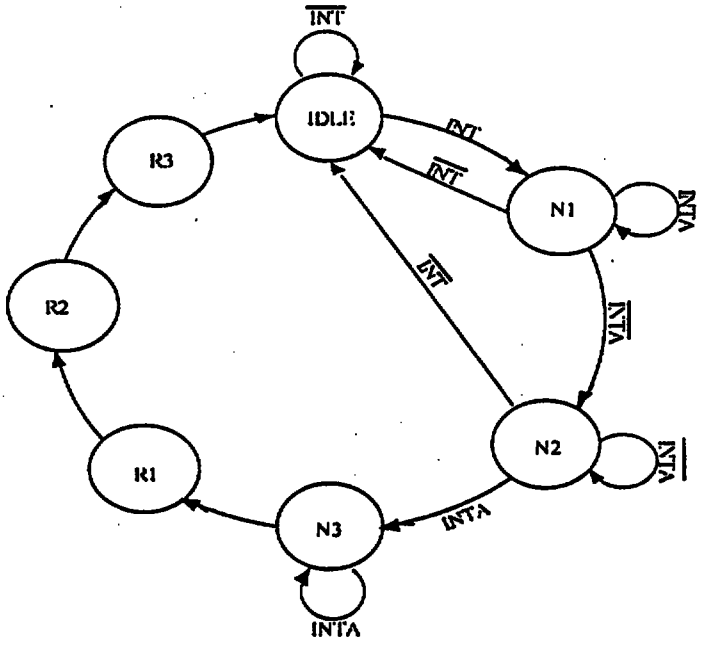
providing said interrupt signal compliant with said first interrupt type to said interrupt controller for interrupt handling.

15. The method of claim 14 wherein said first interrupt type comprises edge sensitive interrupts and said second interrupt type comprises level sensitive interrupts, said step of generating an interrupt signal compliant with said first interrupt type comprises the step of following the following state machine model:



wherein when the state machine is in the N1, N2 and N3 states the state machine model outputs an edge sensitive interrupt signal to the interrupt controller, when the state machine is in the Delay state the state machine circuitry outputs no interrupt signal, the INT signal comprising an active result of the logical OR operation of the interrupt lines from all peripheral agents that generate interrupts of said second interrupt type, the INTA signal comprising a signal from the CPU indicating that it is processing an interrupt.

16. The method of claim 14 wherein said first interrupt type comprises edge sensitive interrupts and said second interrupt type comprises level sensitive interrupts, said step of generating an interrupt signal compliant with said first interrupt type comprises the step of following the following state machine model:



wherein when the state machine is in the N1, N2 and N3 states the state machine model outputs an edge sensitive interrupt signal to the interrupt controller, the INT signal comprising an active result of the logical OR operation of the interrupt lines from all peripheral agents that generate interrupts of said second interrupt type, the INTA signal comprising a signal from the CPU indicating that it is processing an interrupt, the R1, R2 and R3 states comprising successive state machine states where the state machine circuitry outputs no interrupt signal.



17. The method of claim 14 further comprising the step of storing control bits identifying a selected interrupt input to said interrupt controller.

18. The method of claim 17 wherein said providing said interrupt signal step comprises the step of providing said interrupt signal to said selected interrupt input responsive to said control bits.