2228349

COMPUTER BUS WITH VIRTUAL MEMORY DATA TRANSFER CAPABILITY USING VIRTUAL ADDRESS/DATA LINES

BACKGROUND OF THE INVENTION

1. Field of the Invention:

The present invention relates to a method and apparatus for transferring data among various components of a computer system. More particularly, the present invention relates to an improved computer bus with virtual memory capabilities.

2. Art Background:

In a typical computer system, a central processing unit (CPU) reads from, and writes data to, peripheral devices and other components which comprise the data processing system. This communication between devices is frequently accomplished by means of a bus which interconnects all of the components of the computer system. The speed at which the bus transfers data directly limits the overall speed of the computer system. Common constraints on the speed of data transfer between components coupled to the bus are protocol or "handshake" restrictions. These require a predetermined sequence of events to occur within specified time limits prior to an actual exchange of data between the devices coupled to the bus. It is therefore desirable to have a high speed and high bandwidth bus which operates quickly to minimize the computing time required for a particular task. The protocol utilized by the bus should be designed to be as efficient as possible and minimize the time required for data transfer.

In addition, many systems employ virtual memory schemes to increase the capability of the data processing system. As is well known, virtual memory techniques permit a CPU to address more memory then there is physically present in the computer system's main memory. When the CPU needs to read or write data in main memory, it generates a virtual address for the data. This virtual address is translated into a physical address by a memory management unit (MMU). The physical address is then applied to the main memory to read or write data at that address. Since the time required for the CPU to access main memory is typically much less than the access time for a peripheral device (such as an external disk drive), overall speed of program execution is increased.

The MMU accepts a virtual address generated by the CPU and translates it into a corresponding physical address located in main memory. If the translation data is not currently stored in the MMU, then the MMU transfers the proper translation data into its memory from main memory or another peripheral device. MMU's also provide capabilities such as multiple address spaces for separate processes, sharing of memory between processors, and the protection of desired memory locations by permitting only certain operations such as read, write, or execute to such areas. A variety of virtual memory systems are known in the field of computer systems, and are commonplace in state-of-the-art computing equipment. One such type of memory management unit is described in U.S. Patent No. 4,550,368.

Another limitation on a computer bus is the size of the bus itself. Essentially, a bus is a collection of wires connecting the various components of a computer system. In addition to address lines and data lines, the bus will typically contain clock signal lines, power lines, and other control signal lines. As a general rule, the speed of the bus can be increased simply by adding more lines to the bus. This allows the bus to carry more data at a given time. However, as the number of lines increases, so does the cost of the bus. It is therefore desirable to have a bus which operates as quickly as possible while also maintaining a bus of economical size.

SUMMARY OF THE INVENTION

According to the present invention there is provided in a computer system including a number of data processing devices ("agents"), and a controller agent, selectively coupled to a single bus structure, a method of transferring data between a requesting agent and an accessed agent associated with a physical address comprising the steps of:

asserting a bus request signal over a bus request line from a requesting agent to a controller agent;

issuing a bus grant signal over a bus grant line from said controller agent to said requesting agent when said bus is available for use by said requesting agent;

deasserting said bus request signal over said bus request line from requesting agent to said controller agent;

applying a virtual address to data lines in said bus from said requesting agent;

latching said virtual address in a virtual address latch (VAL) means for storing said virtual address;

transmitting said virtual address from said VAL to a memory management unit (MMU);

translating said virtual address into a physical address by means of translation data stored in said MMU;

applying said physical address to address lines in said bus;

asserting a transfer size signal from said requesting agent to an agent associated with said physical address;

asserting a bus select signal from said controller agent to said accessed agent associated with said physical address;

transmitting an acknowledge signal from said

accessed agent associated with said physical address over acknowledge lines in said bus;

applying said data to said data lines;

wherein said data is transferred between said requesting agent and said accessed agent associated with said physical address over said single bus structure.

Also according to the present invention there is provided a computer system with a bus having virtual memory data transfer capability, comprising:

a bus comprised of a first plurality of lines;

a plurality of data processing devices ("agents") selectively coupled to said bus;

a controller means coupled to said bus for controlling access to said bus;

translating means coupled to said bus for translating virtual addresses provided by said agents over said bus into physical addresses;

said first plurality of lines including a second plurality of lines wherein both virtual addresses and data are transmitted over the same second plurality of lines.

An improved high speed bus with virtual memory capability is disclosed. The bus has particular application in computer systems which employ peripheral devices. The bus allows high speed data transfer through the use of a virtual memory scheme. Moreover, the number of lines required to implement the bus is minimised. The amount of time a particular device is required to wait before it can access the bus and complete a data transfer is also minimised. The bus comprises a plurality of lines including address lines, data lines and various command or control lines. A variety of data processing units, referred to as agents, are coupled to the bus. A

controller is provided which controls access to the bus by the agents. A memory management unit (MMU) and a virtual address latch (VAL) are coupled to the bus to implement the virtual address scheme.

During operation of the bus, data is transferred between agents over the bus. A requesting agent asserts a virtual address over the data lines in the bus. This virtual address is converted into a physical address by the MMU. This physical address is applied to address lines in the bus. The requesting agent asserts a predetermined sequence of control signals, and the receiving agent which is storing the data asserts the requested data over the data lines. Also described is a method of preventing deadlock conditions to occur during the operation of the bus. If a requesting agent supplies a virtual address to the MMU and the MMU does not have the necessary translation data to generate a physical address, then the MMU will issue a "rerun" signal. This signal allows the requesting agent to "back off" and temporarily suspend the pending data transfer process. The MMU can then gain control of the bus and fetch the proper translation data. After the MMU has obtained the necessary translation data, the data transfer process is resumed. Without the rerun signal the MMU would be unable to gain control of the bus. This could result in a deadlock condition where no data is transferred and the bus "freezes up". Also described is a scheme whereby certain lines in the bus are tristatable, and are always driven active and inactive (asserted and deasserted by active drive) before they are tri-stated. This is in contrast to prior bus structures wherein lines were deasserted passively by means of pull-up or pull-down resistors. The tri-stating scheme reduces power consumption and allows the bus to be directly driven by CMOS devices.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a schematic representation of a computer system employing the preferred embodiment of the present invention.

Figure 2a diagrammatically illustrates the various sub bus structures comprising the high speed bus of the present invention.

Figure 2b illustrates the structure of the acknowledge lines in the present invention.

Figure 3 is a timing diagram showing the timing protocols for tri-stated control signals used in the present invention.

Figure 4 is a timing diagram showing the driver protocol for tri-stated non-control signal lines used in the present invention.

Figure 5 is a chart showing, in tabular form, the possible encoded acknowledge responses of an accessed agent.

Figure 6 is a chart showing in tabular form, the possible transfer line signals, and their meanings.

Figure 7a is a timing diagram for a CPU read cycle for transferring data over the bus.

Figure 7b is a timing diagram for a CPU write cycle for transferring data over the bus.

Figure 8a is a timing diagram for a DMA device write cycle for transferring data over the bus.

Figure 8b is a timing diagram for a DMA device read cycle for transferring data over the bus.

Figure 9 is a timing diagram for a burst write cycle to memory.

Figure 10 is a timing diagram for a rerun cycle as used in the present invention.

DETAILED DESCRIPTION OF THE INVENTION

An improved high speed bus is described having particular application for use in chip-level computer systems employing virtual memory techniques. In the following description for purposes of explanation specific memory sizes, bit arrangements, numbers, data transfer rates, etc. are set forth in order provide a thorough understanding of the present invention. It will be apparent to one skilled in the art, however, that the present invention may be practiced without these specific details. In other instances, well known circuits and components are shown in block diagram form in order not to obscure the present invention unnecessarily.

Referring first to **Figure 1**, the major components of a computer system employing the present invention are shown. A central processing unit (CPU) 12 communicates with the elements of the computer system over bus 10. Bus 10 includes a plurality of individual lines which carry data and control signals to various components of the computer system. The system shown in **Figure 1** further includes a main memory 16, a video device 18, a Direct Memory Access (DMA) device 20, a co-processor 22, a disk interface 24, and a network interface 25. Additional units such as accelerators or additional memory may be added to the system if desired. For the purposes of this Specification, all data processing and peripheral devices which are coupled to the bus 10 of the present invention are collectively referred to as "agents" and given the general reference numeral 26.

The present invention is intended to allow the various elements of the computer system to communicate with each other. It has been found that the present invention is particularly suited for communication among high performance chip-level devices. It is often desirable, however, for the computer system to communicate with other systems. These systems may not utilize the same communication protocols used in the present bus system. Thus, the computer system may be provided with a separate input/output (I/O) bus 60. This I/O bus 60 may use any one of a number of communication schemes which are well-known in the art. Of course, it will be apparent to those skilled in the art that the present bus system can be used to

communicate with external devices if those devices use or understand the timing protocols of the present invention.

The present invention also comprises a controller/arbiter 14 which is coupled to the bus 10. As will be described more fully below, the controller 14 performs various functions including controlling and allocating access to the bus 10 among the various agents. The controller does this by asserting a bus grant signal over the bus grant line 49. Each agent coupled to the bus has a unique bus request, bus select and bus grant lines coupled between the controller/arbiter 14 and the particular agent. When an agent requires control of the bus, it asserts a signal over its bus request line 49. The controller 14 receives all such requests, prioritizes and stores them, waits until the bus is idle and asserts a bus grant signal over the bus grant line 49 to one of the requesting devices. In the preferred embodiment, the controller uses a round robin priority arbitration scheme. It will be apparent to these skilled in the art, however, that different arbitration schemes can be used as a matter of design choice. For example, the controller 14 could assign bus grants on a first-requested, first-granted basis. Alternatively, each agent could be assigned a unique priority number and the controller 14 could assign bus grants to the requesting agent with the highest priority. The controller also prevents the occurrence of deadlock conditions which arise when two or more agents seek access to the bus 10 at the same time. This accomplished through the use of a "rerun" protocol which is described below.

As will be described, the present invention comprises a bus with a high speed parallel data transfer rate, low power consumption, and a minimum number of lines. Furthermore, the present invention minimizes the time required for a particular agent to gain access to the bus. Although **Figure 1** illustrates specific processor and peripheral units coupled to the bus 10, it will be apparent to those skilled in the art that the structure of the present invention permits a wide variety of processing units and peripheral devices to be interconnected in a variety of systems using the bus structure described herein.

The preferred embodiment of the bus is designed to be used with CMOS technology. As such, certain power specifications and timing protocols are optimized to reduce power consumption and work efficiently with CMOS devices. It will be

apparent to those skilled in the art that those specifications may be changed without departing from the sprint of the present invention.

Throughout the following description, certain bus lines will be described as being "active high" or "active low." This terminology is used to indicate whether a particular bus signal in question is, or is not, asserted and relevant to the bus operation. It will be apparent, however, that it is a matter of design choice whether a given bus line is active high or active low. While the preferred embodiment has been designed to maximize data transfer speed and minimize power consumption, it will be appreciated that changing the design of the bus to change an active high line to active low is within the scope of the present invention.

Similarly, as described more fully below, the preferred embodiment of the bus is completely synchronous. All bus signals change on the rising edge of the clock pulses. The synchronous design is desirable because high data transfer rates require high clock speeds, and it is easier to detect clock pulse edges at high speeds. It will be appreciated that the use of the rising edge of the clock pulses is a matter of design choice and that the present invention could be made completely synchronous with the falling edges of the clock pulses instead.

Referring now to **Figure 2a**, the preferred embodiment of the present invention comprises a bus 10 which includes ninety-six individual lines. As shown, the bus is connected to the various agents 26 which comprise the computer system illustrated in **Figure 1** (including main memory 16, video port 18 and disc drive 24). The bus 10 includes a clock line 40 which distributes a central clock signal among the agents 26. The bus 10 also includes physical address lines 42 and virtual address/data lines 43. In the preferred embodiment, there are twenty-eight physical address lines and thirty-two virtual address/data lines. However, a different number of such lines can be used depending on the size of the data to be transferred. An address strobe line 41 is provided which allows a receiving agent to strobe the address which is present on the physical address lines 42.

During the operation of the bus 10, various protocol signals are asserted at specific times in order to insure accurate data transfer among the agents 26. To this end, the present invention includes transfer size lines 44 which indicate the width (in

bytes) of the data to be transferred, and acknowledge lines 46 which allow an agent accessed by the CPU 12 or other agent to acknowledge a request for data. As will be described, the accessed agent provides these responses in an encoded format. Memory error line 47 is asserted if an agent from whom data is requested detects an error during transfer.

The bus 10 also contains lines for the control of the bus, so that only one agent 27 will drive the bus at any given time. As previously described, the bus request line 48 and bus grant line 49 are unique to each agent. There are also interrupt request lines 50 which allow an agent to interrupt the bus operation. This may be necessary if an agent needs to complete a high priority task and is unable to obtain normal access to the bus because of an ongoing data transfer. In the preferred embodiment, there are seven interrupt request lines. It will be apparent to those skilled in the art, however, that a larger or smaller number of such lines may exist without departing from the scope of the present invention. For example, if it is expected that agents will interrupt the bus only infrequently, there can be a single interrupt request line which is shared by all of the agents. Bus reset line 51 is provided which is used to reset the bus. Reset may be necessary during initial power-up or if an error condition occurs. Finally, power lines 52 supply the voltage and current necessary to operate the bus.

Most of the lines which make up the bus are digital lines which can be driven to either a high level (1) or a low level (0). However, the preferred embodiment also contains control lines driven following a special protocol. In many prior art systems, digital lines may be asserted by being driven active low. When the line is to be de-asserted, the agent ceases driving the line low and pull-up resistors return the line to its normally high state. The speed which the line is returned to its in-active (high) state is a function of the capacitance of the line and the value of the pull-up resistance.

As shown in **Figure 3**, whenever a tri-statable signal is asserted, the driver for that signal first pulls the signal low (asserted). After asserting the signal, the driver will drive the signal high (inactive) before turning off. The tri-statable control signals have passive pull-up resistors which maintain the signals to the inactive level after the drivers turn off. Because of the low power DC specifications of CMOS devices it is

not possible to provide active pull-up resistors which change the signals from the active to inactive levels within a short time period. Therefore, the protocol adopted in the present invention is used. The resistors on the tri-statable lines do not change the state of the signal from active to inactive. They merely maintain the signal at its inactive level. By driving the tri-statable signals, both low and then high, the problem of "floating" signals which is present in many prior art bus systems is avoided.

Another feature of the present invention is the fact that no single line is driven by two agents during the same clock cycle. This avoids the problem of driver overlap between multiple drivers. The timing protocol for this feature is shown in **Figure 4**. At the rising edge of the first clock cycle, the driver for the agent which is driving a particular line turns off. If the line is a tri-statable line it will be maintained to its tri-stated level by its passive resistor. The line will be at its tri-stated level by the end of the first clock cycle. When the rising edge of the second clock cycle is detected, the driver of the second agent will turn on and drive the line to the desired level.

The operation of the bus will now be described with reference to **Figure 1** and **Figures 5** through **10**. There are numerous tasks that can be accomplished with the use of the bus 10. While every such task cannot be covered in detail, the general operation of the bus can be described with reference to several examples. It is to be understood, therefore, that the following examples are given for illustrative purposes only, and are not intended to limit in any way the types of operations that may be performed by the bus 10.

Assume, for example, that the CPU 12 is required to read data stored in main memory 16. A timing diagram for this process is illustrated in Fig 7a. The CPU 12 generates a virtual address corresponding to the data to be read. As shown in **Figure 1**, this virtual address is transmitted directly to the MMU 30. The MMU 30 translates the virtual address to a physical address by means of any one of a number of methods known in the art, and transmits the physical address to the bus 10 over the physical address lines 42. As shown in **Figure 7a**, at the same time the physical address is driven over the physical address lines 42 the CPU 12 also asserts the transfer size lines 44 to their proper levels.

The transfer size lines 44 provide an encoded signal to the memory 16 of the width (in bytes) of the data to be transferred over the bus 10. In the preferred embodiment there are three transfer size lines. **Figure 6** shows the various possible combinations of how the transfer size lines can be driven and their corresponding meaning as to the width of the data transfer. In the preferred embodiment, the bus 10 can accommodate data transfers of 1, 2, 3, 4 or 16 bytes. It will be apparent to those skilled in the art, however, that different encoding schemes and corresponding data transfer sizes can be accomplished as a matter of design choice. The data lines 43 of the present invention are 32 bits wide. Therefore, the bus 10 can transfer up to 4 bytes of data simultaneously. As will be described fully below, if a 16 byte transfer is indicated by the transfer size lines, then the bus 10 executes multiple four-byte transfers.

The controller 14 asserts the bus select line 53. When the controller 14 asserts the address strobe line 41, the memory 16 recognizes that it is being accessed and responds with an acknowledge (ACK) signal over the acknowledge lines 46. As shown in **Figure 2b**, the acknowledge lines 46 are made up of three separate lines: 8-bit acknowledge 55, 32-bit acknowledge 56, and error 57.

An accessed agent (in the present example, memory 16) has four choices of response. It can instruct the CPU 12 (or other accessing agent) to wait, issue a "rerun" response (which is described below), terminate the cycle if an error condition exists, or acknowledge the data transfer while simultaneously indicating the accessed agent's port size. **Figure 5** is a table of the encoded responses used in the preferred embodiment. As shown, the bus 10 can accommodate agents having port sizes of 8, 16, and 32 bits (1, 2, and 4 bytes). It will be apparent to those skilled in the art that a different encoding scheme can be used as a matter of design choice. Moreover, it is within the scope of the present invention to allow for the use of devices having different port sizes - for example 4 bits or 64 bits. In the preferred embodiment, the acknowledge lines are active low.

After asserting the acknowledge lines 46, the memory 16 drives them high in the next clock cycle before tri-stating them as described previously with reference to **Figure 3**. Upon receipt of the acknowledge signal, the controller 14 de-asserts the

address strobe 41 (which is an active low signal). One clock cycle after asserting the acknowledge lines 46, the memory 16 drives the data lines 43 with valid data, and maintains the data valid for one full clock cycle on the data lines 45. After the data lines 43 are asserted, the memory 16 tri-states them, ending the read cycle.

Assume, for sake of example, that the CPU 12 is writing data to memory 16. The timing diagram for this process is shown in Fig 7b. The CPU 12 asserts the physical address lines 42, transfer size lines 44, and applies data to the data lines 43. The controller 14 asserts the bus select line 53 and address strobe 54 on the same clock pulse. Substantially concurrently, the CPU 14 drives the data lines with the data to be written into memory 16. When the address strobe 41, and bus select lines 53, are valid on the rising edge of the same clock cycle, the memory 16 responds with an acknowledge signal as described above. Upon receiving a valid acknowledge signal, the controller de-asserts the address strobe 41, and the CPU 12 de-asserts the data lines 44. As shown in **Figure 1**, the CPU 12 is coupled directly to the MMU 30. Accordingly, in the present embodiment, CPU 12 provides virtual addresses directly to the MMU, thereby by-passing the virtual address latch 28 and bus 10. However, it will be noted that CPU 12 may be directly coupled to bus 10, just as any other agent 26, such that its virtual addresses are applied to the data lines 43 as in the example described below with reference to DMA device 20.

Figure 8a illustrates a timing diagram for a DMA device writing data directly to memory. As shown, the DMA device requests access to the bus by asserting the bus request line 48. As previously discussed, presently request line 48 is an active low line. The controller waits until the bus is inactive, and issues a bus grant signal to the requesting agent according to the arbitration scheme described above. On the immediately following clock cycle, the DMA device de-asserts the bus request and drives a virtual address onto the data lines 43. At the same time, the DMA device indicates the transfer size, as discussed above in relation to **Figure 6**. and asserts the read line.

By driving the virtual address onto the data lines the present invention achieves several advantages over the prior art. First, all of the agents connected to the bus are able to access memory through the MMU. This technique eliminates the

need for a separate set of virtual address lines in the bus. As such, the total number of lines in the bus decreases, reducing complexity and manufacturing costs.

As shown in **Figure 1**, the data/virtual address lines 43 are coupled to the virtual address latch 28. During the first clock cycle after a bus master (for example, CPU 12) receives a bus grant, the master applies the virtual address to the data lines 43. The virtual address latch 28 stores the virtual address value. This virtual address is then transferred into the MMU 30 which converts it to a physical address corresponding to a real address in main memory 18. As shown in **Figure 8a** the physical address is asserted on the physical address lines by the MMU 30 after it is translated. Substantially concurrently with the physical address being asserted by the MMU 30, the controller 14 asserts bus select 53 and address strobe 41. In response to a simultaneously valid address strobe 41, bus select 53, and physical address 42, the slave device restores the write cycle and completes it by asserting ACK 46. Once the slave asserts ACK 46, the controller 14 will de-assert the bus grant signal after the cycle completes. When the bus grant signal is de-asserted, the master device no longer has control of the bus, and stops driving all signals after the next cycle, thus ending the write cycle.

Figure 8b shows a timing diagram for a DMA device read from memory. As illustrated, the DMA device 20 asserts a bus request signal on the bus request line 48. When the bus is inactive, the controller 14 issues a bus grant signal in accordance with the arbitration scheme described above. Subsequently, the DMA device 20 de-asserts the bus request line 48, and applies to the data lines 43 the virtual address of the memory location to which the data is to be written. At the same time, the DMA device asserts the transfer size lines 44, and drives the read line 45 high. After the virtual address is driven onto the data lines, the DMA device 20 tri-states the data lines as discussed above with reference to **Figure 3**. The MMU 30 translates the virtual address into a physical address, and applies this physical address tc the physical address lines 42. When the physical address lines are asserted, the controller 14 asserts address strobe 41 and bus select 53. In response the memory 16 responds with a valid ACK signal over the acknowledge lines 46. After a valid ACK signal is asserted, the DMA device 20 drives the data to be written

onto the data lines 43. Simultaneously, the controller 14 de-asserts the bus grant line 49. On the next clock cycle the DMA device tri-states the data lines, the transfer size lines, and the read latch, thus completing the DMA read cycle.

The present invention can accommodate a high-bandwidth burst transfer mode wherein multiple bytes are written to or read from memory 16 by the CPU 12 or other agent 26. A timing diagram for one such CPU read operation is shown in **Figure 9**. The agent 26 which is transferring data into memory requests control of the bus 10 by asserting a bus request signal over the bus request line 48. When the bus 10 is inactive, the controller 14 issues a bus grant signal in accordance with the arbitration procedures of the present invention as described previously. Once the bus grant is issued, the agent 26 which issued the bus request signal becomes a Bus Master and has control over the operation of the bus 10. On the clock cycle immediately following the bus grant, the Bus Master de-asserts the bus request line 48, and drives the read signal line 45 high. Substantially concurrently, a virtual address is driven onto the data lines 43 by the Bus Master. The Bus Master indicates that the transfer will consist of 16 bytes by asserting the transfer size lines 44 in accordance with the encoding scheme given in **Figure 6** above.

The virtual address is converted into a physical address by the MMU 30 and driven onto the physical address lines 42. At this point the memory must respond with a valid signal on the 32-bit acknowledge line 46 (active low). Upon receipt of a valid acknowledge signal, the Bus Master drives the first 4 bytes onto the data lines 43 to be written into memory 16. The memory 16 responds by driving the 32-bit acknowledge line 56 high and then low. The memory 16 also automatically increments the address into which the next group of four bytes will be transferred. After the 32-bit acknowledge line 46 is driven low, the Bus Master will latch a new group of four data bytes onto the data lines 43 to be written into or read from the new memory address. This process is repeated for each group of four bytes. After four groups of data have been transferred, the controller 14 de-asserts the bus grant line 49. The Bus Master then de-asserts the address strobe 41 and tri-states the read line 45 and transfer size lines 44. The memory tri-states the 32-bit acknowledge line 46, thereby ending the write cycle.

The present invention further implements a "rerun" protocol during which any pending cycle may be aborted and started later. This allows resolution of deadlocks between the CPU and DMA devices and allows agents to have a long read latency without locking out other bus requests.

Deadlocks can occur when an agent is capable of being both a slave agent and a Bus Master and cannot complete a slave access at a given time. The agent 26 must detect this condition and assert a rerun acknowledge by driving the 8-bit acknowledge line 46 and the error line 47 as shown in **Figure 5**. The controller 14 will respond by terminating the cycle by negating the bus select line 53 and repeating the cycle at a later time. Prior to the repeat of the cycle, other bus masters may utilize bus 10. An example of a rerun cycle is shown in **Figure 10**.

The rerun protocol is useful for several reasons. First, it allows agents other than the CPU 12 to access main memory 16 through the MMU 30. Assume that an agent transmits a virtual address to the MMU 30 through the VAL 28. If the corresponding physical address is not currently stored in the MMU 30, translation data for the MMU issues a rerun signal as described above. The requesting agent will then "back off" allowing the MMU 30 to fetch the proper translation data. The MMU 30 executes this transfer over the bus 10. The requesting agent can then retry its operation with the necessary data in the MMU 30 to permit translation of the virtual address into a physical address. Without the rerun protocol, the MMU would be unable to translate the virtual address into a physical address. The MMU 30 would be unable to gain control of the bus. This could lead to a deadlock condition, where the bus 10 would "freeze" the system, preventing data transfer.

The rerun feature also minimizes the amount of the time that a particular agent will be required to wait until it gains access to the bus. This waiting period is known as bus latency time. Bus latency occurs when a particular agent accesses a slave with a long access time, locking out all other agents. This can happen when an accessed agent requires a long period of time to supply requested data to the bus.

Any agent with a bus latency time greater than a predetermined period of time (N) must respond before the time out with a rerun acknowledge. If the accessed agent takes longer than N cycles to respond, then the current (read or write) cycle is

aborted and any pending bus request is serviced. After the pending request is serviced, the original cycle is restarted. During the rerun period, the accessed agent latches any asserted address and data lines, as these signals are not valid after the rerun cycle has been suspended.

In our co-pending application GB 2263378 (divided out of the present application) there is described and claimed a bus system for transferring data between data processing units ("agents").

In our co-pending application GB 2263349 (also divided out of the present application) there is described and claimed a method of transferring data among agents selectively coupled to a computer bus.

* C L M S *

CLAIMS

1. In a computer system including a number of data processing devices ("agents"), and a controller agent, selectively coupled to a single bus structure, a method of transferring data between a requesting agent and an accessed agent associated with a physical address comprising the steps of:

asserting a bus request signal over a bus request line from a requesting agent to a controller agent;

issuing a bus grant signal over a bus grant line from said controller agent to said requesting agent when said bus is available for use by said requesting agent;

deasserting said bus request signal over said bus request line from requesting agent to said controller agent;

applying a virtual address to data lines in said bus from said requesting agent;

latching said virtual address in a virtual address latch (VAL) means for storing said virtual address;

transmitting said virtual address from said VAL to a memory management unit (MMU);

translating said virtual address into a physical address by means of translation data stored in said MMU;

applying said physical address to address lines in said bus;

asserting a transfer size signal from said requesting agent to an agent associated with said physical address;

asserting a bus select signal from said controller agent to said accessed agent associated with said physical address;

transmitting an acknowledge signal from said accessed agent associated with said physical address over acknowledge lines in said bus;

applying said data to said data lines;

wherein said data is transferred between said requesting agent and said accessed agent associated with said physical address over said single bus structure.

2. The method of claim 1 wherein said bus further includes clock line means for providing said agents with a clock signal, said clock line means coupled to each of said agents and to a bus clock, wherein said signals transmitted over said bus change state synchronously with a rising edge of said clock signal.

3. The method of claim 1 wherein said controller agent grants requesting agents access to said bus on a round robin priority basis.

4. The method of claim 1 wherein there are 32 data lines.

5. The method of claim 1 wherein said bus further includes a clock line means for providing said agents with a clock signal, said clock line means coupled to each of said agents and to a bus clock, wherein said signals transmitted over said bus change state synchronously with a portion of the clock signal.

6 . The method of claim 5 wherein only one of said agents drives said bus during any given clock cycle.

7. The method of claim 1 wherein the controller

agent grants access to said bus using a predetermined arbitration scheme.

8. A computer system with a bus having virtual memory data transfer capability, comprising:

a bus comprised of a first plurality of lines;

a plurality of data processing devices ("agents") selectively coupled to said bus;

a controller means coupled to said bus for controlling access to said bus;

translating means coupled to said bus for translating virtual addresses provided by said agents over said bus into physical addresses;

said first plurality of lines including a second plurality of lines wherein both virtual addresses and data are transmitted over the same second plurality of lines.

9. The computer system of claim 8 further comprising a central processing unit (CPU) coupled to said translating means.

10. The computer system of claim 9 further comprising an input/output bus coupled to said central processing unit.

11. The computer system of claim 10 wherein said plurality of data processing devices ("agents") includes a direct memory access (DMA) device.

12. The computer system of claim 11 wherein said plurality of data processing devices ("agents") further includes a main memory.

13. The computer system of claim 8 further comprising a latch for storing said virtual addresses, said latch coupled to said translating means.

14. The computer system of claim 13 wherein said translating means includes means for temporarily suspending translation of a virtual address into a physical address in order to obtain proper translational data over said bus.

15. A computer system with a bus having a virtual memory data transfer capability, comprising:

a bus;

a central processing unit coupled to said bus;

a plurality of data processing devices coupled to said bus;

a memory management unit coupled to said bus for translating virtual addresses into physical addresses;

a controller means coupled to said bus and said plurality of data processing devices for controlling access to said bus;

wherein said bus is comprised of:

clock line means for distributing a central clock signal among said devices;

physical address line means for providing physical addresses to said devices;

address strobe line means for strobing a physical address present on said physical address line means;

transfer size line means for indicating the width of data to be transferred over said bus;

acknowledge line means for allowing said devices to acknowledge a request for data;

memory error line means for allowing said devices to indicate an error detected during data transfer;

bus request line means for permitting a requesting device to request access to said bus;

bus grant line means for indicating to said requesting device that said requesting device has been granted access to said bus;

interrupt request line means for allowing a device to interrupt a bus operation;

bus select line means for selecting a device for the transfer of data;

virtual address/data line means for transferring both virtual addresses and data.

16. A computer system as provided in claim 15, further comprising a virtual address latch coupled to said bus and said memory management unit for receiving and storing virtual addresses.

17. The computer system of claim 16 wherein said translating means includes means for temporarily suspending translation of a virtual address into a physical address in order to obtain proper translational data over said bus.

18. The computer system of claim 17 further comprising an input/output bus coupled to said central processing unit.

19. The computer system of claim 18 wherein said plurality of data processing devices ("agents") include a direct memory access (DMA) device.

20. The computer system of claim 19 wherein said plurality of data processing devices ("agents") further include a main memory.

21. The computer system of claim 20 wherein said plurality of data processing devices ("agents") further include: a video device, a coprocessor, a disk interface, and a network interface.

22. In a computer system including a number of data processing devices and a controller agent, selectively coupled to a single bus structure, a method of transferring data between a requesting agent and an accessed agent associated with a physical address, as claimed in claim 1 and substantially as hereinbefore described.

23. A computer system with a bus having virtual memory data transfer capability, substantially as hereinbefore described with reference to the accompanying drawings.