Description

The present invention relates to data transfer on a computer bus using the bus address lines.

General purpose computer systems are usually designed to transfer data between various subsystems over a shared data path known as a bus. The various subsystems, such as a central processor, cache controller, and input/output (I/O) subsystems such as keyboard, video, and mass storage controllers may all be attached to a single system bus. The system main memory may also be attached to the system bus. In some systems, main memory is connected directly to the central processor, with the system bus being reserved for I/O data transfer through a Direct Memory Access (DMA) device or other I/O controller.

Since a large amount of data is transferred between numerous subsystems on a single bus, the bus itself can often become a performance bottleneck for the overall system. Designing a system bus to be able to handle the heavy load placed upon it is a difficult task. The difficulty of this task is increased in many systems because the precise nature of the equipment to be included in the system may not be known in advance to the bus designer. Different subsystems, made by different manufacturers and having different performance specifications, may all be attached to a single system bus at the same time.

The system bus must be designed to perform correctly regardless of which subsystems are attached to it. In practice, this usually means that a bus is designed to a "lowest common denominator", in which certain aspects of the bus design are chosen to sacrifice performance in order to assure correct operation with all subsystems.

The nature of subsystems to be attached to the system bus can vary. Some subsystems, such as keyboard I/O, usually transfer only a single character or word at one time. Other subsystems, such as video device controllers or DMA controllers attached to mass storage subsystems, usually transfer data in blocks of words. In order to support efficient transfer of block data, many system buses include a block transfer mode, also called a "burst mode" on some systems, which is designed for the express purpose of transferring blocks of consecutive data words. When a block transfer is in effect, some of the standard bus control transactions are not used, which decreases bus overhead and increases the data transfer rate.

Although block transfer modes allow blocks of data to be more efficiently transferred on a bus, the overall system bus performance is limited by the fact that all of the subsystems will be attached to the bus at all times. Thus, bus performance limitations, which are included to allow some low performance subsystems to operate properly, prevent potentially higher performance subsystems from being able to transfer data at a higher rate. In traditional computer system bus designs it is not possible to have different subsystems transfer data over a single bus at different rates. It is also not possible to transfer data using word width greater than those used for normal data transfers.

United States Patent Specification 4,213,176 describes a system according to the pre-characterising part of claim 1, in which in one mode of operation data is transferred between devices by using both data and address lines. However, on each such line the data is transferred at the normal rate.

It is therefore an object of the invention to provide a system in which data is transferable at different rates on the same bus on either the data lines or on both the data lines and the address lines.

This object is achieved by the characterising part of claim 1.

In order that the invention may be fully understood a preferred embodiment thereof will now be described, by way of example only, with reference to the accompanying drawings in which:

FIGURE 1 is a block diagram showing a master subsystem and a slave subsystem connected to a computer system bus;

FIGURE 2 is a flowchart illustrating operation of a high speed data transfer mode;

FIGURES 3a, 3b and **3c**, considered together are a flowchart illustrating the operation of a bus master device;

FIGURES 4a and **4b**, considered together, are a flowchart illustrating operation of a bus slave device; and

FIGURE 5 is a timing diagram illustrating bus signal sequencing for a sample data transfer.

The preferred embodiment of the invention which will now be described is based upon the Micro Channel bus which is incorporated in PS/2 microcomputer products available from IBM (IBM, PS/2 and Micro Channel are trademarks of the International Business Machines Corporation). Only the control signals necessary for illustrating this preferred embodiment of the present invention are included in this description; additional aspects of the bus upon which this description is based are available in published literature.

Figure 1 shows a computer system bus **10** with two attached subsystems **12, 14**. For convenience, the bus signals are broken down into addressing signals ADDR, data signals DATA, and control signals CTRL. The addressing signals ADDR define memory locations within the memory map of the system. For systems such as the PS/2 family which have separate address spaces for system memory and I/O devices, the address signals ADDR also include a signal (not shown) indicating whether the current bus address is a memory address

or an I/O address.

In a high speed block data transfer according to the present embodiment, two devices are involved. One device **12** coupled to the bus will be considered the bus master device, with the other device referred to as the bus slave device **14**. Consistent with traditional usage, the bus master **12** is the subsystem which is in charge of the transfer. The slave **14** generates signals and may terminate the transfer, but its control logic is generally simpler than that of the master **12**.

In many systems, any given subsystem may be capable of operating either as the master or the slave. In particular, subsystems which are capable of operating as a bus master are often capable of operating as a slave instead. Many simpler devices, such as memory subsystems, are only capable of operating as slaves.

The high speed data transfer which will now be described is referred to herein as a streaming data transfer. This terminology is used to distinguish this new transfer mechanism from previous ("normal") block mode transfers which operate on different principles.

The discussion of **Figures 2, 3, 4,** and **5** makes the assumption, in all cases, that the bus master can perform the described high speed transfer process. If the master cannot perform the high speed transfer, either by design or because of something in the master's current operational state, data transfer is performed in the normal mode used by the Micro Channel bus.

Figure 2 is a high level block diagram illustrating the data transfer process according to the present embodiment. First, a normal bus data transfer cycle begins **20**. At this point, no decision has been made as to whether this cycle will be a normal data transfer or a streaming data transfer. After the cycle begins, the bus master determines whether the slave device supports streaming data transfers **22**. If not, a normal bus cycle is completed **24**. If the slave does support streaming data, and assuming that the master does also, a check is made **26** to see whether the transfer is complete. If it is not, data is transferred **28** and control refers to step **26**. During the time that control is looping between steps **26** and **28**, the remaining devices attached to the bus can see only that the original bus cycle begun in step **20** has not yet completed. Therefore, they ignore any changing of signals which occurs during the block data transfer.

Once the transfer is complete, as tested in step **26**, the normal bus cycle completes **24**. The remaining devices on the bus then operate in normal fashion again. It is possible that only a single data transfer will be made at the higher rate, in which case little or nothing is gained. However, the maximum number of data items which may be transferred can be quite large. In many systems, the maximum number will be determined by a bus timeout period. Since a data cycle is begun in step **20**, but has not yet completed, a typical bus timeout controller will think that the slave device has not responded at all.

Figure 3 and **4** are flowcharts illustrating operation of the bus master **12** and slave **14**, respectively. Although steps are shown sequentially, some are actually performed in parallel. For example, steps **34** and **38** in **Figure 3** are actually performed simultaneously. Steps **40, 44,** and **46** are also performed together. Such implementation details are performed in the same manner as previously done on the Micro Channel bus.

Figure 3 is a detailed flowchart illustrating operation of a bus master **12** as used with the preferred embodiment. If the master **12** does not support streaming data transfers, it will simply operate in the normal mode. The flowchart of **Figure 3** illustrates the flow of control within a bus master **12** which does support streaming data transfers.

At the beginning of a bus cycle, the master asserts the address and select signals **30**. The select signal, SELECT, is a signal indicating whether the address placed on the bus is a memory address or an I/O port address. Only a single address must be asserted for a streaming data transfer. As will be described, the master **12** and the slave **14** both automatically increment an internal address each time a data item is transferred. This is somewhat analogous to prior art block transfers, in which a starting address and a count of the number of data items to be transferred are the only address related signals required by a DMA device. However, in prior art DMA transfers, the DMA controller typically asserts an address for each data item transferred. In the presently described system, both the master **12** and slave **14** increment addresses internally, so that the address bus is not used for addresses after the start of the streaming data transfer. As described below, in the presently described system no block length is defined in advance. Instead, either the master **12** or the slave **14** may terminate the transfer.

Shortly after the address is asserted, a READ/WRITE signal is asserted **32**. This indicates whether data will be transferred from the master **12** to the slave **14** (a write cycle), or from the slave device **14** to the master **12** (a read cycle).

Next, an address latch signal, ADL, signal is asserted **34**. This signal indicates that the address currently on the address lines of the bus **10** is valid, and typically causes the slave device **14** to write the address into its internal buffers. Bus definitions typically specify a minimum delay period between the time that the addresses become valid and their latching with the ADL signal.

The bus master **12** determines the data width at which it is capable of communicating **36**. A master **12** may

not be capable of transferring data at the maximum width allowed by the bus **10**. For example, on a 32 bit data bus, a given device acting as a bus master for this cycle may only be capable of transferring 8 or 16 bits at a time. In most bus masters, the maximum width of the master does not change, and will actually have been determined at some fixed time in the past. However, the results of that determination should be available at step **36**.

If the current data transfer is a write, thereby transferring data from the master to the slave, the master **12** then asserts (places on the data bus) data **38**. Such data may be asserted simultaneously with ADL. The master **12** also asserts BE(0..3) **39**, which is a 4 bit signal indicating which bytes of the data bus will be used to transmit data during a normal cycle. This signal has an additional use during certain block transfers as described below. The master then begins the data cycle **40**, by transitioning the bus clock. In the preferred embodiment, as illustrated in **Figure 5**, the clock signal is driven from high to low to start the data cycle.

At the time the data cycle begins, the master **12** detects whether the slave **14** is ready to transfer data **42**. This is done by sampling a control line driven by the slave **14** which indicates whether the slave **14** is ready. If the slave **14** is not ready, control passes to step **80** (Figure 3c). If the slave **14** is ready, the master **12** proceeds to determine **44** the data width which can be handled by the slave. Width is sensed on control lines reserved for this purpose, which are driven by the currently addressed slave during each normal bus clock cycle. If the master word width is greater than that supported by the slave **14**, the master **12** will need to abort or complete this cycle and resend the same data in smaller segments which are acceptable by the slave.

In a preferred embodiment, slave data width is indicated by a device size signal DS(16.32) combined with MSDR, which gives a 3 bit signal. The possible values of DS(16.32) and MSDR, and the meanings corresponding to each permutation of values, are shown in Table I. The master **12** will transfer data at the lesser of the master's word width and the slave's word width.

Table I

MSDR	DS(16.32)	Data Width
1	00	32 bits
1	01	16 bits
1	10	Reserved
1	11	8 bits
0	00	64 bits
0	1x	Reserved
0	x1	Reserved

8, 16, and 32 bit transfers are all done on the data lines. 64 bit transfers require that data be transferred on the address bus as well, as is described below.

Up to and including step **44**, all of the actions performed by the master **12** are those which are performed in every bus cycle. That is, no special actions have yet been undertaken in support of a streaming data transfer.

In step **46**, an SDR signal is tested by the master **12** to see if it is valid. The SDR signal indicates whether the slave device **14** supports streaming data transfers. If it does not, control is transferred to step **47** (**Figure 3b**). If the master was reading during this bus cycle it fetches the data from the data bus **47**, and completes the data cycle **48**. Completing the data cycle is accomplished by driving the clock signal from low to high. The READ/WRITE signal should also be driven high if it is still low. READ/WRITE may be driven high at any time after the SDR signal has been tested. At this point, the master **12** has completed a normal bus cycle.

In step **46**, if the slave **14** does in fact support streaming data transfers, control passes to step **49**. If the master **12** has only a single data item to transfer, it will typically complete a normal data cycle by forcing control of the transfer to step **47** regardless of the state of the slave's SDR signal.

When the streaming data transfer is initiated, the master **12** first tri-states the address lines **49**. In the case of a 64 bit transfer, the master then signals **50** that the address lines will be used for data. This signalling is preferably done by driving all of the BE(0..3) lines high. This step is not done unless both the master **12** and slave **14** support 64 bit transfers. If this is a write cycle, the master will put the additional data onto the address lines before the first transfer is made. The master next determines the maximum transfer rate which the slave device supports **51**. In a preferred embodiment, the slave indicates this rate using the same SDR signal which indicates whether the slave supports streaming data transfer.

Table II

SDR(0.1)	Operation & Speed
00	SD at 20 MHz (50ns)
10	SD at 16 MHz (62.5ns)
01	SD at 10 MHz (100ns)
11	Normal Transfer Cycle

In a preferred embodiment, the signal SDR is a 2 bit signal, with the four possible values indicated in Table II. The bit pairs 00, 10, and 01 indicate that the slave **14** can support streaming data transfers at 20 MHz, 16 MHz, and 10MHz respectively. These clock speeds correspond to high speed clock cycle times of 50ns, 62.5ns, and 100ns, respectively.

If the SDR signal set by the slave **14** is 11, then the slave does not support streaming data transfer at all, and the master **12** will cause only the normal transfer cycle to be used as described above. This is the value tested for by the master **12** in step **46**. If the value of SDR is 11, the SDR signal is considered to be not valid.

Once the transfer rate has been determined **51**, the master **12** drives a high speed clock (step **52**), referred to as SD STROBE. This is a free running clock which is a completely separate signal from the normal bus clock.

As described above, in the case of a normal bus cycle the normal bus clock is driven low in step **40**, and driven high in step **48** to complete the bus cycle. In a streaming data transfer the normal bus clock remains low during the entire transfer.

The next step for the master **12** is to strobe the data **54** (Figure **3b**). This means that the master **12** waits for the next cycle of SD STROBE. In a preferred embodiment, the high to low transitions of SD STROBE actually cause the data strobing referred to in step **54**.

At the time that the data is strobed, the slave **14** may not yet be ready to transfer data. The slave indicates this on a signal line which is reserved for this purpose, and which is sensed by the master (step **56**) at the time SD STROBE transitions high-to-low. If the slave **14** is not ready, then the attempted data transfer must be made again. Therefore, if the master **12** is writing to the slave **14**, the same data is again asserted **58** on the data lines, and control returns to step **54**.

Once the slave **14** successfully transfers the data, indicated by the yes branch from step **56**, the master **12** tests **60** to see whether the SDR signal driven by the slave **14** is still valid. SDR is not valid when SDR(0.1)=11, as shown in Table II. If it is not valid, this shows that the slave **14** is terminating the data transfer, and the master **12** transfers control to step **62**, which causes the high speed clock (SD STROBE) to stop. The slave **14** is required to indicate its termination of the data transfer before the last data item is transferred. Thus, after the slave **14** terminates the transfer, one last data item is transferred when the normal data cycle completes in step **48**.

If the SDR signal remains valid in step **60**, control transfers to step **64**. If the master **12** is reading data from the slave **14**, if fetches the data **64** from the bus and latches it into its internal buffers. The master next increments its internal address **66**. After incrementing its address, if the master is writing data to the slave, it asserts the next data item **68**.

Next, in step **70** the master checks to see if this is the master's penultimate (next to last) cycle. If it is not, there are at least two more data items to be transferred, and control returns to step **54**.

If this is the master's next to last cycle, the master asserts the last cycle indicator **72**. In the preferred embodiment, indication of the last cycle is done by raising the READ/WRITE signal which is originally asserted in step **32**. The data is then strobed **74**, and a check is made **76** to see if the slave is ready. If not, the same data is again asserted if the master is writing to the slave **78**, and control returns to step **74**. As shown in **Figure 5**, this determination of slave **14** readiness is made by waiting for the SDR signals to transition high.

When the slave is ready in step **76**, the high speed clock (SD STROBE) is stopped **62**, data is fetched **47** if the master is reading from the slave, and the data cycle is completed **48**. The last item of data to be transferred is always transferred on the normal cycle completion, regardless of whether the streaming data transfer was terminated by the master **12** or the slave **14**.

Returning to step **42**, if the slave **14** was not ready, control passed to step **80**, figure **3c**. In step **80** the master **12** waits for a specified minimum delay and then determines the slave's **14** data width **82**. During this delay the slave **14** must assert a valid data width signal, even if it is not otherwise ready. This delay allows a bus converter, used to attach a second bus to the bus **10** as known in the art, to function properly. The use of this delay was previously included in the Micro Channel bus design.

At the same time as the slave's 14 data width is determined 82, the SDR signal is again tested 84 for validity. SDR must also be asserted during step 80, even if the slave 14 is not otherwise ready. If SDR is now valid, the master 12 waits until the slave 14 is ready 86, and control returns to step 49 (figure 3a) for a high speed streaming data transfer.

If the slave 14 does not support streaming data transfers, step 84 will result in an invalid SDR signal, and in step 87 the master 12 resets the READ/WRITE signal. When the slave 14 is ready to complete the normal bus transfer (step 88), control returns to step 47, figure 3b, and the normal transfer is completed.

Figure 4 illustrates the sequence of steps performed by the slave device 14 during a data transfer. Prior to a new data cycle, the master places the appropriate address on the address lines, and these are decoded 90 by the slave. The slave 14 then asserts 92 the DS(16.32) portion of its device size. As shown in Table 1, DS(16.32) = 00 and MSDR = 0 for 64-bit transfers.

If the slave 14 is not ready to transfer data at the time the address is latched 90, it signals not ready 96 on a line (CHRDY in Figure 5) reserved for this purpose.

The next action taken by the slave 14 depends on whether or not it is capable of supporting streaming data transfers determined in step 98. If the slave is so capable, it asserts in step 99 the streaming data rate which it can handle. This rate is selected form those shown in Table 2. If the slave 14 can handle 64 bit transfers, it asserts the signal MSDR at the same time. MSDR can be returned to the tri-state value anytime after it is sampled when the normal bus clock falls, unless the slave 14 was not ready. The slave tests whether it is ready 100 and, when ready, signals so in step 101. The data cycle starts 102 when the master 12 drives the bus clock. (step 40 in Figure 3) A test is made 104 to see if, during the forthcoming data cycle, the master 12 is preparing to transfer its last data item. If not, the slave 14 determines in step 106 whether or not it is ready to transfer the next data item, and, if not, signals not ready 108 to the master 12. If the slave 14 is ready, and if this a read cycle, the slave asserts data on to the bus in step 110. If this is a 64 bit read, data is asserted on the address lines as well as the data lines. 64 bit transfers are indicated by the master 12 in step 50 of Figure 3.

Next, the data is strobed 112, and, if this a write from the master 12 to the slave 14, the slave 14 fetches the data from the bus 114 and loads it into its internal buffers. Again, if this is a 64 bit transfer, data is read from the address lines as well as the data lines. The slave 14 then increments its internal address 116, and determines (step 118, figure 4b) whether this is the slave's penultimate cycle. This is a determination made of the status of the slave device 14; therefore in some transfers, the slave device 14 will determine the size of the streaming data transfer. This situation can occur, for example, in a system in which the slave is a buffered I/O device with a limited buffer size. During a read (slave to master) cycle, the slave will know how much data it has to transmit, and can terminate the transfer when all the data has been transmitted. In a write (master to slave), the slave may need to terminate a transfer in order to prevent buffer overflow.

In any event, if the slave 14 is not prepared to terminate the transfer after the next data item, control returns to step 104. If the slave is in fact preparing to terminate the transfer, it checks to see whether it is ready 120, and, if not, signals not ready 122. When the slave is ready, and if the data transfer is a read, it asserts its last data item 124.

The slave 14 then resets its SDR rate to 11 in step 126. This is interpreted by the master 12 as a slave termination of transfer as described in connection with Figure 3 (step 60). The data cycle then completes 128, and if this was a write transfer the slave 14 fetches the data from the bus 130 and latches it into its buffers.

If the master 12 terminates the transfer, the yes branch will be taken from step 104, transferring control to step 132. As described in connection with Figure 3, the master termination event is indicated by raising the READ/WRITE signal. If the slave 14 is not ready 132 it simply waits. When it is ready, the slave responds by resetting the SDR rate to 11 in step 134. If this is read transfer, the slave places its data on the bus 136, and waits for the data cycle to complete 128. As in the usual case, if this is a write transfer, data is fetched from the bus 130 after the cycle completes.

If the slave 14 is not ready in step 132, it is not necessary to signal this fact as is done in steps 96 and 122. Instead, when the master 12 terminates the transfer and raises the READ/WRITE signal, the slave 14 signals that it is ready by resetting the SDR rate (step 134). This hand shaking protocol prevents the master 12 from completing the cycle until the slave 14 is ready to transfer the last data item.

Returning to step 98, if in fact the slave 14 does not support streaming data transfers, control branches to step 144. The slave 14 waits until it is ready 144, and when ready, signals such fact 142. The normal data cycle starts 140, and if this is a read cycle the slave 14 asserts data 136. The data cycle then completes 128 as described above.

Figure 5 is a timing diagram illustrating most of the operations described in connection with Figures 3 and 4 with a sample transfer. The sample transfer described is a four word streaming data transfer. Several options, such as read and write transfers, and master and slave terminated transfers, are illustrated.

The address signals of the bus **10** are shown as ADDR. For the timing diagram of **Figure 5**, ADDR includes the memory/I/O select line (SELECT). ADDR is 32 bits wide, not including SELECT. The signals S0 and S1 are control lines driven by the master **12** to indicate whether this is a read or write transfer. If S0 is low, the transfer is a read. If S1 is low, it is a write transfer. S0 and S1 may not both be low at the same time.

The signal BE(0..3) is used to indicate which bytes of the 32 bit data bus are transmitting the data. It is also used by the master **12** to indicate that a 64 bit transfer is occurring. The signal ADL is the signal used to latch the address, and the CMD is the normal bus clock. SD STROBE is the high speed clock used in streaming data transfers. The signals ADDR, S0, S1, BE(0..3), ADL, CMD, and SD STROBE are all driven by the bus master **12**.

The signal DATA contains 8, 16, or 32 bits of parallel data, whichever data width is supported by both the master **12** and slave **14**. The DATA lines are driven by the master **12** during a write transfer, and by the slave **14** during a read transfer.

The signal CHRDY is used by the slave **14** to indicate whether or not it is ready to transfer data as described in connection with **Figure 4**. The 2 bit signal SDR(0.1) indicates whether or not the slave **14** supports streaming data transfers, and the speed of such transfers if they are supported, as defined in Table II. MSDR indicates whether the slave **14** supports 64-bit transfers. The 2 bit signal DS(16.32), when combined with MSDR as defined in Table I, indicates the slave word width. The signals CHRDY, SDR(0.1), MSDR, and DS(16.32) are all driven by the slave **14**.

Initially, the address lines are not guaranteed to contain valid values **200**. To initiate a transfer, the master **12** drives valid address signals **202** onto the lines ADDR. Next, the appropriate read or write signals S0, S1 are driven low **204**. The slave **14** drives its data width onto DS(16.32) **205** when it recognizes that it has been addressed **205**. After the ADDR signals have been valid for at least a defined minimum time, ADL is driven low **206** to latch the address into the slave's buffers. When the slave **14** detects the transition of ADL, it sets the SDR signals **208** to show the transfer rate it supports. The SDR signals are typically driven high **210** from a tri-state condition when the slave device **14** detects that it is being addressed. The tri-state to high transition is not controlled by the transition of ADL, as was the case with the DS signals.

If this is a write transfer, the master **12** next places information **212** on DATA, which was previously tri-stated. The signal BE(0..3) is driven **213** to the appropriate values to indicate which bytes of DATA will be used for this transfer. The master **12** then begins the data cycle by driving CMD low **214**. Preferably, SD STROBE is driven low **216** at the same time, and ADL is driven high **218**. Steps **40, 44, 46, 50,** and **52** shown in **Figure 3** are thus occurring simultaneously, in that the master **12** is starting the SD STROBE, and determining the data width and transfer rate of the slave simultaneously, at the time CMD transitions low **214**.

After CMD is driven low **214**, the slave **14** can allow DS(16.32) to become undefined **219**. If this is a read (slave to master) transfer, the slave drives the first data item (D0) onto the data lines **220** in response to the high-to-low transition **214** of CMD.

64 bit transfers must be supported by both the master **12** and the slave **14**. If this is a 64 bit transfer, the master **12** tri-states the address lines **221** after CMD goes low **214**. This is followed by driving all of BE(0..3) high **222** to indicate to the slave **14** that a 64 bit transfer will occur. In response to the BE(0..3) transition **222**, data is placed on the address lines **223**. The master **12** puts this data on ADDR if the transfer is a write, and the slave **14** does so if it is a read. The handshaking of the transitions ADDR to BE(0..3) and BE(0..3) to ADDR prevents collisions between address and data signals on ADDR.

The SD STROBE signal is now free running. The "strobe data" steps of **Figures 3** and **4** occur on the high-to-low transitions **224, 225, 226,** and **228** of SD STROBE. After each such transition, the device which is driving data onto the bus places the next data item on DATA, and ADDR for 64 bit transfers. For so long as the slave can keep up with the transfer, a new item will be placed in DATA and ADDR for each cycle of SD STROBE.

In the example of **Figure 5**, the slave determines during the second high-to-low transition **224** of SD STROBE that it will be unable to transfer data item D1 at the next cycle. This is the determination made in step **106** of **Figure 4**. Since the slave **14** is not ready to transfer the next data item, it drives the signal CHRDY low **230**. When the signal SD STROBE next makes a high-to-low transition **225**, the master **12** detects that the slave **14** has indicated that it is not ready. This detection is made in step **56** of **Figure 3**.

The signal SD STROBE continues running, even though no data is being transferred. If this is a write cycle, the master **12** continues placing the same data onto the bus. If it is a read cycle, the master ignores whatever data happens to be on the bus, as this data is assumed to be invalid. **Figure 5** shows CHRDY being held low by the slave **14** for only a single clock cycle, but it may be held low longer if necessary. When the slave **14** is ready to transfer the next data item, control in **Figure 4** passes to step **110**, and CHRDY is allowed to go high **232**. At the next high-to-low transition **226** of SD STROBE, the master **12** detects that the data has properly transferred, and continues its normal function.

A master terminated transfer is indicated when the master **12** drives the appropriate signal, S0 or S1, high

234. This is preferably done simultaneously with a high-to-low transition 228 of SD STROBE. This transition 234 is detected in step 104 of Figure 4, and the slave 14, when ready, drives the signals SDR(0.1) high 236. This is shown in step 134 of Figure 4. Once the slave 14 has indicated ready 236, the master 12 drives CMD high 238 to complete the data cycle. This is shown in step 48 of Figure 3 and step 128 of Figure 4. The last data item, D3 in Figure 5, is transferred on the positive-going transition 238 of CMD.

After the slave 14 detects the low-to-high transition 238 of CMD, it tri-states SDR(0.1) 240, DATA 242, and ADDR 243 if this was a read transfer, in order to prevent bus collisions with other devices.

If the slave 14 terminates the transfer, it indicates such fact by a low-to-high transition 244 of SDR(0.1). The master 12 senses this transition asynchronously, so that the timing of the transition 244 need not be synchronized with SD STROBE. The master 12 senses this transition in step 60 of Figure 3. In response to the end of transfer indication 244 from the slave 14, the master 12 drives S0 or S1 high 246, and also drives CMD high 238. As before, the final data item D3 is transferred on the low-to-high transition 238 of CMD. Also, DATA, ADDR, and SDR(0.1) are tri-stated after CMD goes high.

The remaining devices on the bus are completely unaffected by the transfer of Figure 5 because CMD has remained low throughout the entire transfer. When ADL was driven low 206, the remaining devices, which were not addressed, became idle. They are awakened from their idle state only when CMD transitions from low to high 238. To the remaining devices, the streaming data transfer appears simply as a stretched-out normal transfer.

The events which occur during a normal transfer are those shown to the left of the first dashed line 250, and those shown to the right of the second dashed line 252. In a normal transfer, CMD goes low 214, D0 is placed on the data bus by the master 12 or by the slave 14, and that same D0 is transferred when CMD transitions low-to-high 238. As described above, the remaining devices on the bus are unaffected by any events occurring between the dashed lines.

The details of the bus interfaces needed to implement the master and slave as described above are well within the skill of a person skilled in the art. The interfaces can utilize programmable logic arrays or other programmable devices to reduce parts count.

As will be appreciated by those skilled in the art, the streaming data transfer described above allows a very high transfer rate for block data. It allows both high performance and low performance devices to operate normally on the same bus. Those devices for which a high speed block transfer is suitable, such as mass storage controllers and video controllers, can take advantage of streaming data transfer, while other devices can be built for less cost. Since data is transferred on the address lines as well as the data lines, the effective transfer rate is doubled without requiring any higher performance from any of the subsystem involved.

Claims

1. A system for transferring data in a digital computer, comprising a bus (10) having address lines, data lines and control lines, a master subsystem (12) connected to the bus, a slave subsystem (14) connected to the bus, and a control line in the bus for communicating a transfer mode signal which indicates whether data transfer between the master subsystem (12) and the slave subsystem (14) is to take place in a first mode or a second mode, and wherein data is transferred between the master subsystem (12) and the slave subsystem (14) on the data lines when the system is operating in the first mode, and data is transferred between the master subsystem (12) and the slave subsystem (14) on both the data and address lines when the system is operating in the second mode,

   characterised in that

   a first clock line is provided in the bus (10) for communicating a first clock signal (CMD, figure 5) and a further clock line is provided in the bus (10) for communicating a high speed clock signal (SD STROBE), and in that when the system is operating in the first mode the data is transferred under the control of the first clock signal and when the system is operating in the second mode the data is transferred under the control of the high speed clock signal.

2. A system as claimed in claim 1, wherein the high speed clock signal (SD STROBE) is free running compared to the first clock signal (CMD), and in that in the second mode the data is transferred during a single cycle of the first clock signal.

3. A system as claimed in Claim 1 or 2, wherein, prior to transferring data in the second mode, the master subsystem communicates an address on the address lines to the slave subsystem.

234. This is preferably done simultaneously with a high-to-low transition 228 of SD STROBE. This transition 234 is detected in step 104 of Figure 4, and the slave 14, when ready, drives the signals SDR(0.1) high 236. This is shown in step 134 of Figure 4. Once the slave 14 has indicated ready 236, the master 12 drives CMD high 238 to complete the data cycle. This is shown in step 48 of Figure 3 and step 128 of Figure 4. The last data item, D3 in Figure 5, is transferred on the positive-going transition 238 of CMD.

After the slave 14 detects the low-to-high transition 238 of CMD, it tri-states SDR(0.1) 240, DATA 242, and ADDR 243 if this was a read transfer, in order to prevent bus collisions with other devices.

If the slave 14 terminates the transfer, it indicates such fact by a low-to-high transition 244 of SDR(0.1). The master 12 senses this transition asynchronously, so that the timing of the transition 244 need not be synchronized with SD STROBE. The master 12 senses this transition in step 60 of Figure 3. In response to the end of transfer indication 244 from the slave 14, the master 12 drives S0 or S1 high 246, and also drives CMD high 238. As before, the final data item D3 is transferred on the low-to-high transition 238 of CMD. Also, DATA, ADDR, and SDR(0.1) are tri-stated after CMD goes high.

The remaining devices on the bus are completely unaffected by the transfer of Figure 5 because CMD has remained low throughout the entire transfer. When ADL was driven low 206, the remaining devices, which were not addressed, became idle. They are awakened from their idle state only when CMD transitions from low to high 238. To the remaining devices, the streaming data transfer appears simply as a stretched-out normal transfer.

The events which occur during a normal transfer are those shown to the left of the first dashed line 250, and those shown to the right of the second dashed line 252. In a normal transfer, CMD goes low 214, D0 is placed on the data bus by the master 12 or by the slave 14, and that same D0 is transferred when CMD transitions low-to-high 238. As described above, the remaining devices on the bus are unaffected by any events occurring between the dashed lines.

The details of the bus interfaces needed to implement the master and slave as described above are well within the skill of a person skilled in the art. The interfaces can utilize programmable logic arrays or other programmable devices to reduce parts count.

As will be appreciated by those skilled in the art, the streaming data transfer described above allows a very high transfer rate for block data. It allows both high performance and low performance devices to operate normally on the same bus. Those devices for which a high speed block transfer is suitable, such as mass storage controllers and video controllers, can take advantage of streaming data transfer, while other devices can be built for less cost. Since data is transferred on the address lines as well as the data lines, the effective transfer rate is doubled without requiring any higher performance from any of the subsystem involved.

Claims

1. A system for transferring data in a digital computer, comprising a bus (10) having address lines, data lines and control lines, a master subsystem (12) connected to the bus, a slave subsystem (14) connected to the bus, and a control line in the bus for communicating a transfer mode signal which indicates whether data transfer between the master subsystem (12) and the slave subsystem (14) is to take place in a first mode or a second mode, and wherein data is transferred between the master subsystem (12) and the slave subsystem (14) on the data lines when the system is operating in the first mode, and data is transferred between the master subsystem (12) and the slave subsystem (14) on both the data and address lines when the system is operating in the second mode,

   characterised in that

   a first clock line is provided in the bus (10) for communicating a first clock signal (CMD, figure 5) and a further clock line is provided in the bus (10) for communicating a high speed clock signal (SD STROBE), and in that when the system is operating in the first mode the data is transferred under the control of the first clock signal and when the system is operating in the second mode the data is transferred under the control of the high speed clock signal.

2. A system as claimed in claim 1, wherein the high speed clock signal (SD STROBE) is free running compared to the first clock signal (CMD), and in that in the second mode the data is transferred during a single cycle of the first clock signal.

3. A system as claimed in Claim 1 or 2, wherein, prior to transferring data in the second mode, the master subsystem communicates an address on the address lines to the slave subsystem.

* C L M S *

4. A system as claimed in Claim 1, 2 or 3, further comprising a transfer rate line in the bus for communicating a transfer rate signal, wherein the high speed clock signal operates at a rate determined by the transfer rate signal when the transfer mode signal indicates the second mode.

Patentansprüche

1. Ein System für die Übertragung von Daten in einem Digital-Computer, einen Bus (10) mit Adreßleitungen, Datenleitungen und Steuerleitungen umfassend, ein an den Bus angeschlossenes Master-Subsystem (12), ein an den Bus angeschlossenes Slave-Subsystem (14) und eine Steuerleitung in dem Bus zur Übertragung eines Übertragungsmodus-Signals, das anzeigt, ob die Datenübertragung zwischen dem Master-Subsystem (12) und dem Slave-Subsystem (14) in einem ersten Modus oder einem zweiten Modus stattfinden soll, und bei dem Daten zwischen dem Master-Subsystem (12) und dem Slave-Subsystem (14) auf den Datenleitungen übertragen werden, wenn das System im ersten Modus arbeitet, und Daten zwischen dem Master-Subsystem (12) und dem Slave-Subsystem (14) sowohl auf den Datenals auch auf den Adreßleitungen übertragen werden, wenn das System im zweiten Modus arbeitet,
dadurch gekennzeichnet, daß
in dem Bus (10) eine erste Taktleitung für die Übertragung eines ersten Taktsignals (CMD, Figur 5) bereitgestellt wird, und eine weitere Taktleitung in dem Bus (10) zur Übertragung eines Hochgeschwindigkeits-Taktsignals (SD STROBE) bereitgestellt wird, und dadurch, daß, wenn das System in dem ersten Modus arbeitet, die Daten unter Steuerung durch das erste Taktsignal übertragen werden, und wenn das System in dem zweiten Modus arbeitet, die Daten unter Steuerung durch das Hochgeschwindigkeits-Taktsignal übertragen werden.

2. Ein System nach Anspruch 1, bei dem das Hochgeschwindigkeits-Taktsignal (SD STROBE) im Vergleich mit dem ersten Taktsignal (CMD) frei läuft, und daß im zweiten Modus die Daten während eines einzigen Zyklus des ersten Taktsignals übertragen werden.

3. Ein System nach Anspruch 1 oder 2, bei dem, bevor Daten im zweiten Modus übertragen werden, das Master-Subsystem dem Slave-Subsystem auf den Adreßleitungen eine Adresse mitteilt.

4. Ein System nach Anspruch 1, 2 oder 3, desweiteren eine Übertragungsrate-Leitung in dem Bus umfassend, für die Mitteilung eines Übertragungsrate-Signals, bei dem das Hochgeschwindigkeits-Taktsignal mit einer Rate arbeitet, die vom Übertragungsrate-Signal bestimmt wird, wenn das Übertragungsmodus-Signal den zweiten Modus anzeigt.

Revendications

1. Système pour transférer des données dans un ordinateur numérique, comprenant un bus (10) ayant des lignes d'adresse, des lignes de données et des lignes de commande, un sous-système maître (12) connecté au bus, un sous-système esclave (14) connecté au bus, et une ligne de commande dans le bus pour communiquer un signal de transfert de mode qui indique si le transfert de données entre le sous-système maître (12) et le sous-système esclave (14) doit avoir lieu dans un premier mode ou un second mode, et dans lequel les données sont transférées entre le sous-système maître (12) et le sous-système esclave (14) sur les lignes de données lorsque le système fonctionne dans le premier mode, et les données sont transférées entre le sous-système maître (12) et le sous-système esclave (14) sur les lignes de données et les lignes d'adresse lorsque le système fonctionne dans le second mode,
caractérisé en ce que :
une première ligne d'horloge est fournie dans le bus (10) pour communiquer un premier signal d'horloge (CMD, figure 5) et une ligne d'horloge supplémentaire est fournie dans le bus (10) pour communiquer un signal d'horloge à grande vitesse (SONDE SD), et en ce que lorsque le système fonctionne dans le premier mode, les données sont transférées sous la commande du premier signal d'horloge, et lorsque le système fonctionne dans le second mode, les données sont transférées sous la commande du signal d'horloge à grande vitesse.

2. Système tel que revendiqué dans la revendication 1, dans lequel le signal d'horloge à grande vitesse (SONDE SD) s'exécute librement comparé au premier signal d'horloge (CMD), et, dans le second mode,

* C L M S *

les données sont transférées durant un seul cycle du premier signal d'horloge.

3. Système tel que revendiqué dans la revendication 1 ou 2, dans lequel, avant de transférer les données dans le second mode, le sous-système maître communique une adresse sur les lignes d'adresse au sous-système esclave.

4. Système tel que revendiqué dans la revendication 1, 2 ou 3, comprenant en outre une ligne à vitesse de transfert dans le bus pour communiquer un signal de vitesse de transfert, dans lequel le signal d'horloge à grande vitesse opère à une vitesse déterminée par le signal de vitesse de transfert lorsque le signal de mode de transfert indique le second mode.

* C L M S *