MEMORY TRANSACTIONS ON A LOW PIN COUNT BUS

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is related to U.S. Patent application no. _____, filed _____, entitled I/O TRANSACTIONS ON A LOW PIN COUNT BUS, U.S. Patent application no. _____, filed _____, entitled DIRECT MEMORY ACCESS (DMA) TRANSACTIONS ON A LOW PIN COUNT BUS, and U.S. Patent application no. _____, filed _____, entitled BUS MASTER TRANSACTIONS ON A LOW PIN COUNT BUS. Each of the above identified applications are assigned to the same assignee as the present application.

FIELD OF THE INVENTION

The present invention relates to computer systems. More particularly, the present invention relates to communicating between different types of devices on a low pin count bus in a computer system.

BACKGROUND

Conventional legacy-based computer systems include a variety of peripheral and memory devices that communicate with the system's chip-set or processor via an Industry Standard Architecture (ISA) bus or an Expansion bus (X-bus). The system chip-set or processor must include a large amount of pins (e.g., approximately 50-70 pins) and associated circuitry to support the ISA bus or X-bus signals that are used to interface the chip-set or processor with legacy-based peripheral devices including input/output (I/O) or I/O controller devices such as parallel port controllers, serial port controllers, super I/O controllers, floppy disk controllers, keyboard controllers, and memory devices such as non-volatile memory devices that store, for example, basic input-output services (BIOS) information.

The large number of pins needed to support the ISA bus and X-bus standards generally increases overall system cost. For example,

larger packages are required for a chip-set or processor. Larger pin-count packages generally cost more than smaller pin-count packages, and are generally more susceptible to manufacturing quality and reliability problems during package assembly and system integration processes. Furthermore, larger pin-count packages require a greater amount of surface area on a printed circuit board (PCB) on which a computer system may be implemented. Therefore, it would be desirable to replace the ISA bus or X-bus in a computer system with a bus that would support the same types of peripheral devices, but utilize a relatively smaller number of pins or signal lines.

The ISA bus and X-bus standards also limit the amount of memory or I/O address space to 24 bits or approximately 16 MBytes of memory or I/O address space. As some processors and chip-sets can support 32 bits or 64 bits of address space, aliasing has previously been required to translate the processor address space to the ISA address space. Aliasing can produce significant overhead such that the performance of the computer system may be negatively impacted. Therefore, it would be desirable to replace the ISA bus and X-bus standards with a bus that may support an address space that is compatible with the processor or chip-set address space, and that potentially is unlimited.

The ISA bus and X-bus standards have a further disadvantage of being asynchronous. This generally increases design complexity for components that need to generate and respond to ISA bus and X-bus signals. Therefore, it would be desirable to replace the ISA bus and X-bus standards with a bus that may support synchronous communication between components so as to generally reduce the complexity of the bus interface circuitry.

SUMMARY OF THE INVENTION

A system is described having a bus coupled to a host and a memory device. The bus may include a plurality of general purpose signal lines to carry time-multiplexed address, data, and control

information. The memory device may store system start-up information and communicate this information with the host over the bus.

Other features and advantages of the present invention will be apparent from the accompanying drawings and from the detailed description that follows.

BRIEF DESCRIPTION OF THE DRAWINGS

The features and advantages of the present invention are illustrated by way of example and not limitation in the figures of the accompanying drawings in which like references indicate similar elements and in which:

Figure 1 is one embodiment of a computer system having a low pin count (LPC) bus;

Figure 2 is one embodiment of an LPC interface;

Figure 3 is one embodiment of a state machine diagram for implementing memory and I/O transactions;

Figure 4 is one embodiment of an LPC memory read transaction;

Figure 5 is one embodiment of an LPC memory write transaction;

Figure 6 is another embodiment of a state machine diagram for implementing memory and I/O transactions;

Figure 7 is another embodiment of an LPC memory read transaction;

Figure 8 is another embodiment of an LPC memory write transaction;

Figure 9 is one embodiment of a memory device selection scheme;

Figure 10 is one embodiment of an LPC interface for a memory device; and

Figure 11 is one embodiment of a timing diagram for a low power mode of operation.

DETAILED DESCRIPTION

A computer system with a low pin count (LPC) bus is disclosed. The LPC bus may replace the ISA bus or X-bus in a computer system while providing a communication mechanism between a host, such as a processor or chip-set, and peripheral devices, such as I/O or I/O controller devices and memory devices. For one embodiment, the LPC bus may include general purpose signal lines that carry substantially all time-multiplexed address, data, and control information to implement memory, I/O, direct memory access (DMA), and bus master transactions between the host and the peripheral devices.

An intended advantage of the LPC bus is that it may require significantly fewer signal lines (e.g., approximately 6-8 signal lines) to perform all of the functions previously performed by the approximately 50-70 signals lines of the ISA bus and X-bus standards. Since the number of signals lines for the LPC bus may be greatly reduced over the ISA bus and X-bus standards, the number of pins required on a computer system's chip-set or processor to support the LPC bus may be significantly reduced. Similarly, an LPC interface included within each peripheral device may use an equally small number of pins to interface with the chip-set or processor via the LPC bus. This may result in lower packaging costs for the component manufacturer and lower system costs for the computer system manufacturer. Additionally, this may result in higher quality and more reliable packaging and system integration. The LPC bus may also be used to replace other system busses to reduce pin counts on packages coupled to the busses.

Another intended advantage of the LPC bus is that memory and/or I/O address space is not limited to a particular number of address lines in a communication bus; rather, the LPC bus may serially carry as many address signals as are required to address any address space. Thus, conventional aliasing of memory or I/O address space may be avoided.

Another intended advantage of the LPC bus is that the LPC interface logic and/or software that may be used to interface a chip-set or processor with a peripheral device may be software compatible with existing computer systems employing ISA bus or X-bus standards. That is, no special driver or configuration is required by operating system software or application programs to run on an LPC bus system vs. ISA bus or X-bus systems.

Still another intended advantage of the LPC bus is that it may be a synchronous bus enabling LPC interface circuitry within host and peripheral devices to generally require less complex circuitry than asynchronous bus interface designs.

The LPC bus and associated supporting circuitry may also be included in other systems including printers, digital cameras, and communication systems (e.g., modems, cellular phone systems, pagers, and digital answering systems).

Figure 1 is one embodiment of computer system 100 including a host 102 coupled to peripheral devices including I/O or I/O controller devices 104 and 106, and memory device 108 via an LPC bus 124. Host 102 may be a chip-set, processor, or any other control logic that includes intelligence for interfacing with LPC bus 124. Host 102 may also include the necessary logic to interface with an optional ISA bus 126 and an optional PCI or Host bus 128. Host 102 communicates address, data, and control information with memory 110 over bus 130 that may be a local bus or other memory bus. Memory 130 may be the main or primary memory for system 100 such as dynamic random access memory (DRAM), static random access memory (SRAM), or any other type of volatile or nonvolatile memory that may store, for example, operating system software, application code, or program data.

I/O devices 104 and 106, may be I/O controller devices that are be addressable in I/O address space and control access to peripheral components such as floppy disk drives, keyboards, and the like. I/O devices 104 and 106 may each be, for example, a floppy disk controller,

a serial port controller, a parallel port controller, an infra-red (IR) controller, a keyboard controller, audio controller device, or video controller device. I/O device 104 may a Super I/O device that includes I/O controllers 120 and 122 that may each be, for example, a floppy disk controller, a serial port controller, a parallel port controller, an infra-red (IR) controller, a keyboard controller, audio controller device, or video controller device.

Memory device 108 is a peripheral device that is addressable in memory space. For example, memory device 108 may include any type of volatile or nonvolatile memory such as DRAM, SRAM, flash memory, electrically programmable read only memory (EPROM), electrically erasable programmable read only memory (EEPROM), magnetic storage media, or optical storage media. Memory device 108 may store system start-up code, manageability code, operating system data, application code, program data, or function as a scratch pad for host 102 or I/O devices 104 and 106. System start-up code may include the necessary software to configure or boot components within system 100, and may include BIOS information.

Each of host 102 and the peripheral devices 104, 106, and 108 include LPC interfaces 118, 112, 114, and 116, respectively, for generating LPC signals and for responding to LPC signals on LPC bus 124. These LPC interfaces may enable system 100 to implement memory, I/O, DMA, and bus master transactions over LPC bus 124. Each LPC interface has either input or output pin(s) to receive or transmit signals LAD[3:0], LCLK, and LFRAME#. Note that a pound symbol "#" following a signal name indicates that the signal is an active low signal. For alternative embodiments, active low signals may be altered to be active high signals.

The LAD[3:0] signal lines are general purpose signal lines that carry time-multiplexed address, data, and control information over LPC bus 124 and between host 102 and peripheral devices 104, 106, and 108. Whether LAD[3:0] is carrying address, data, or control information

at any given time may be determined by the operating context of system 100 as will be described in greater detail below. While LAD[3:0] is illustrated as a four-bit bus, it may include any number of signal lines to carry out an intended purpose of carrying time-multiplexed address, data, and control information. A four-bit nibble-sized bus may be advantageous to reduce the number of pins required on each of the LPC interfaces 112-118 and reduce the number of pins and package sizes of host 102 and peripheral devices 104, 106, and 108.

LCLK is a clock signal that may be generated by LPC interface 118 or by other circuitry within host 102 (not shown). Alternatively, LCLK may be generated by a system clock (not shown) including a crystal or other timing circuitry. For one embodiment, LCLK may be the PCI clock from PCI bus 128.

LFRAME# is generated by LPC interface 118 of host 102 to indicate the start and/or end of LPC transactions or cycles. Each of LPC interfaces 112, 114, and 116, may monitor or sample LFRAME# to determine when an LPC transaction may begin or end. For example, when LFRAME# is asserted (e.g., low) by host 102, LPC interfaces 112, 114, and 116 of peripheral devices 104, 106, and 108, respectively, monitor LAD[3:0] to determine whether the transaction or cycle is for them. Advantageously, when a transaction is not intended for a particular peripheral device, that peripheral device may enter a lower power mode and may de-couple its state machines from the bus and/or gate its clocks. If LFRAME# is asserted for more than one LCLK cycle, then LPC interfaces 112, 114, and

116 may only react to the start control information on LAD[3:0] during the last LCLK cycle that LFRAME# is active. Table 1 summarizes exemplary definitions of the start control information on LAD[3:0] while LFRAME# is asserted.

Table 1

LAD[3:0]	Function
0000	Start of transaction
0001	Reserved
0010	Grant for bus master 0
0011	Grant for bus master 1
0100	Reserved
1101	Memory read
1110	Memory write
1111	Stop/Abort transaction

LFRAME# is used to abort or end LPC transactions. For example, LPC interfaces 112, 114, and 116 of peripheral devices, 104, 106, and 108, respectively, continue to monitor LFRAME# during LPC transactions. If LFRAME# is asserted during an LPC transaction, the peripheral devices 104, 106, and 108 will stop driving LAD[3:0] and abort their LPC transaction. For one embodiment, host 102 asserts LFRAME# for a number of LCLK cycles (e.g., four clock cycles) to ensure that the abort indication is recognized by all peripheral devices. During one or more of these LCLK cycles, host 102 may also drive predetermined data on LAD[3:0] after peripheral devices 104-108 have stopped driving LAD[3:0]. For one embodiment, host 102 drives 1111 on LAD[3:0] while LFRAME# is asserted to indicate an abort sequence. At the end of the abort sequence, LFRAME# may be deasserted before a new LPC transaction commences.

For alternative embodiments, LFRAME# may be used between any two devices (e.g., a host and peripheral device) in any system as an independent control signal on an independent control line (apart from a bus) to communicate the start or abortion of a transaction such as a memory, I/O, bus master, or DMA transaction.

LPC bus 124 may also include a reset signal LRESET# input into each of LPC interfaces 112, 114, 116, and 118 and that resets the LPC interfaces to a known state (e.g., an idle state). For one embodiment,

LRESET# may be the same reset signal as a PCI reset signal on PCI bus 128.

I/O devices 104 and 106 may also have DMA request signals LDRQ0# and LDRQ1#, respectively, provided on separate control lines to LPC interface 118 of host 102. LDRQ0# and LDRQ1# may be used to transmit encoded DMA channel requests to host 102 prior to performing a DMA or bus master transaction. Each I/O device may have one unique LDRQ# signal such that peripheral device I/O controllers 120 and 122 share one DMA channel request on LDRQ0#.

LPC bus 124 may also include a low power or standby signal LPCPD# that may be output by host 102 to each of peripheral devices 104-108 on a separate control line. As illustrated in Figure 11, host 102 may assert LPCPD# (e.g., low) asynchronously at time t0 to cause peripheral devices 104-108 to enter a low power mode. Upon recognizing LPCPD# asserted by host 102, peripheral devices 104-108 will halt any LPC transactions and tri-state or drive the LDRQ# signals inactive. Peripheral devices 104-108 and/or host 102 may also tri-state or drive LAD[3:0] to a known state. Host 102 may also drive LFRAME# to a known state (high or low) after asserting LPCPD#. In the low power mode, LCLK may be shut off at time t1 and peripheral devices 104-108 may shut off their state machines and/or other logic. For one embodiment, the time difference between t0 and t1 is at least 30 microseconds. When exiting the low power mode, LCLK may be restarted asynchronously at time t2 and may run for a period of time before LPCPD# is deasserted at time t3. For one embodiment, the time difference between t2 and t3 is at least 100 microseconds. LRESET# may then be used to reset peripheral devices 104-108 from time t3 to t4. For one embodiment, the difference between times t3 and t4 is at least 60 microseconds.

Host 102 and peripheral devices 104-108 may also have additional sideband input and/or output signals utilized in ISA or PCI busses, including interrupt signals (e.g., SERIRQ and SMI#), additional

clock signals (e.g., CLKRUN#), and power management signals as will be described in more detail below.

By utilizing approximately 6-9 signal lines, in one embodiment, LPC bus 124 may provide substantial signal and pin-count savings over conventional ISA bus or X-bus parts or systems. For example, the following ISA bus or X-bus signals typically found on plug-n-play devices may no longer be required: D[7:0], SA[15:0], DREQ[3:0], DACK#[3:0], TC, IOR#, IOW#, IOCHRDY, IOCS16#, and MEMCS16#. This may result in a savings of approximately 30-40 signals on the bus, the host, and/or peripheral devices. For one embodiment, a fully functional Super I/O peripheral controller device that may have required as large as a 160 pin package under ISA or X-bus compatible situations, may require an 88 pin (or less) package using an LPC interface as illustrated in Figure 1.

The LPC bus may be used by the LPC interfaces 112-118 to perform memory, I/O, DMA, and bus master transactions or cycles including memory read, memory write, I/O read, I/O write, DMA read, DMA write, bus master memory read, bus master memory write, bus master I/O read, and bus master I/O write. The type of transaction may be indicated in a field of cycle type and direction control information driven on LAD[3:0] by host 102 or peripheral devices 104, 106, and 108. One embodiment of the cycle type and control information encoded on LAD[3:1] is shown in Table 2. Other encodings on the same or other signal lines may be used.

Table 2

LAD[3:1]	Transaction
000	I/O Read
001	I/O Write
010	Memory Read
011	Memory Write
100	DMA Read
101	DMA Write
110 - 111	Reserved

Figure 2 shows LPC interface 200 that is one embodiment of LPC interface 118 of host 102. LPC interface 200 generates and interprets the address, data, and control information on LPC bus 124 to implement the transactions of Table 2. LPC interface 200 includes memory and I/O LPC control logic 202 that may control memory read, memory write, I/O read, and I/O write LPC transactions between host 102 and one of peripheral devices 104-108. Memory and I/O LPC control logic 202 may include a state machine or sequencer for sequencing between the states necessary for the particular LPC transaction, and for generating address, data, or control information on LAD[3:0], LFRAME#, and/or LCLK. The information provided on LAD[3:0], LFRAME#, and/or LCLK may be provided to bus 214 for output by logic 208 to LPC bus 124.

For one embodiment, memory and I/O LPC control logic 202 determines that an LPC memory or I/O transaction may be required to be performed by communicating with LPC decode logic 210 over one or more signals 216. LPC decode logic 210 may receive memory or I/O read or write request and an associated address from host 102, ISA bus 126, or PCI or host bus 128. LPC decode logic may decode the address and signal memory and I/O LPC control logic 202 to begin a memory or I/O LPC transaction. LPC decode logic 210 may alternatively be located outside of LPC interface 200 but within host 102. The decoded address ranges for peripheral device 104-108 may be defined so as to be

compatible with previous ISA bus and X-bus legacy based peripheral devices to aide is making LPC transactions software transparent to operating system software and application software. For an alternative embodiment, control of the memory and I/O LPC transactions may be separated into different control circuits.

LPC interface 200 also includes DMA LPC control logic 204 that controls DMA read and write LPC transactions between host 102 and one of I/O devices 104 and 106. DMA LPC control logic 204 may include a state machine or sequencer for sequencing between the states necessary for the DMA LPC transaction, and for generating address, data, or control information on LAD[3:0], LFRAME#, and/or LCLK, and for receiving DMA request signals LDRQ0# and LDRQ1#. The information provided on LAD[3:0], LFRAME#, LDRQ0#, LDRQ1#, and/or LCLK may be provided to or from LPC bus 124 via logic 208 and bus 214.

DMA LPC control logic 204 may interface with host 102 and memory 110 via DMA controller 212 that may reside in host 102. DMA LPC control logic 204 may communicate with DMA controller 212 over one or more signals 218. For one embodiment, DMA controller 212 may include one or more 8237 or 8237-compatible DMA controllers. DMA controller 212 may perform arbitration within host 102 on behalf of a DMA channel such that the I/O device using that DMA channel may communicate with main memory 110 via host 102.

LPC interface 200 may also include bus master LPC control logic 206 that control bus master memory read and write LPC transactions, and bus master I/O read and write LPC transactions between host 102 and peripheral devices 104-108. Bus master LPC control logic 206 may include a state machine or sequencer for sequencing between the states necessary for the bus master LPC transactions, and for generating address, data, or control information on LAD[3:0], LFRAME#, and/or LCLK, and for receiving DMA request signals LDRQ0# and LDRQ1#. The information provided on LAD[3:0], LFRAME#, LDRQ0#, LDRQ1#,

and/or LCLK may be provided to or from LPC bus 124 via logic 208 and bus 214.

Logic 208 may comprise one or more buffers or transceivers for interfacing LPC bus 124 with LPC bus 214. Alternatively, LPC interface 200 may not include bus 214; rather, logic 208 may be connected directly and individually to the input and output signals of memory and I/O LPC control logic 202, DMA LPC control logic 204, and bus master LPC control logic 206. For this alternative embodiment, logic 208 may be a selector or multiplexer circuit that outputs or inputs the appropriate LPC signals to and from LPC bus 124 and control logic 202, 204, and 206 under the control of host 102.

Figure 3 is one embodiment of a state diagram that may be implemented one or more of LPC interfaces 112-118 to perform memory and/or I/O write or read LPC transactions between host 102 and peripherals 104-108. For example, the state diagram of Figure 3 may be used by a state machine or sequencer of memory and I/O LPC control logic 202 of Figure 2. This state diagram implements memory read, memory write, I/O read, and I/O write LPC transactions. For other embodiments, separate state machines may implement memory transactions and I/O transactions. The memory read and write LPC transactions will be described with the aide of exemplary timing diagrams of Figures 4 and 5.

An exemplary memory read LPC transaction by host 102 from a memory location in memory device 108 will first be described with reference to Figures 3 and 4. This memory read LPC transaction may be implemented, for example, by host 102 to read system start-up information from memory device 108. At state 302, LPC interface 118 is in an idle state not performing an LPC transaction when LFRAME# deasserted (e.g., high). The process transitions to state 304 when host 102 commences an LPC transaction (e.g., when control logic 202 receives decoded signals from LPC decode logic 210 and host 102 indicating that a memory read LPC transaction is to occur) by asserting

LFRAME# (e.g., low) on LPC bus 124. Host 102 may drive start control information indicating the start of an LPC transaction on LAD[3:0] during cycle T0. For example, the start control information may be 0000 on LAD[3:0] as indicated in Table 1.

The process then transitions to state 306 and host 102 drives cycle type and direction control information on LAD[3:0] during cycle T1. The cycle type and direction control information indicates to peripherals 104-108 that the LPC transaction is a memory read transaction. For example, the cycle type and direction control information may be 010 on LAD[3:1] as indicated in Table 2.

The process then transitions to state 308 and host 102 drives the address of the memory location in selected memory device 108 on LAD[3:0] during n clock cycles T2. The address may be any length of bits supported by a system such as system 100 of Figure 1, and sent over a number of n clock cycles T2. For one embodiment, the memory address space may be 4 GBytes requiring 32 bits of memory address. For this embodiment, eight clock cycles of address information are provided by control logic to LPC bus 124 to transfer 8 nibbles (32 bits) of address information from host 102 to memory device 108. The address information may be provided most significant nibble first, least significant nibble first, or in any other order expected by memory device 108.

For an alternative embodiment, host 102 may optionally drive size control information on LAD[3:0] during a subsequent cycle. For one embodiment, the size control information indicates the number of bytes of memory data to be read from memory device 108 starting at the address provided in state 308. Table 3 provides one embodiment of encoding the number of bytes on LAD[1:0] to be subsequently transferred over LAD[3:0]. Other encodings on the same or different LAD signal lines may also be used.

Table 3

LAD[1:0]	Size
00	8 bits (1 byte)
01	16 bits (2 bytes)
10	Reserved
11	32 bits (4 bytes)

For another embodiment, the size control information may indicate that 2^{size} number of bytes of memory data are to be read from memory device 108 starting at the address provided in state 306.

The process then transitions to turn around (TAR) state 312 to turn control of LPC bus 124 over from host 102 to memory device 108 during n clock cycle(s) T4. For one embodiment, state 312 is two clocks wide. In the first clock cycle host 102 may drive LAD[3:0] lines with predetermined control information (e.g., 1111). In the second clock cycle, host 102 may tri-state LAD[3:0] lines. These lines may have weak pull-ups attached to these lines so that they may remain at a logical high states. For other embodiments, n may be one clock cycle or greater than two clock cycles to turn around LAD[3:0].

The process may then transition to state 314 and in which host 102 waits n clock cycles T5 while memory device 108 drives inband synchronization control information on LAD[3:0]. The synchronization control information may be used to add wait states until the memory data is ready to be output by memory device 108. In this manner, memory devices of differing speeds or access times may be coupled to the same LPC bus. Conventionally, a dedicated control signal on a dedicated control line was used to indicate when data was ready (e.g., IOCHRDY in the ISA bus or X-bus). In contrast, memory device 108 may use inband signaling on LAD[3:0] to insert wait states without requiring the use of an additional dedicated pin or signal line.

For one embodiment, the synchronization control information field has two phases. The first phase drives first control information

on LAD[3:0] (e.g., 0101 or 0110) while accessing the requested data. The second phase drives second control information on LAD[3:0] indicating that the data is ready to be output to LPC bus 124 (e.g., 0000 on Lad [3:0]). Host 102 may monitor the synchronization control information and determine when the second information is sent. Host 102 may then know that the data output by memory device 108 in the subsequent clock cycle(s) T6 will be the data requested by host 102. For one embodiment, the first control information has an encoding (e.g. 0110 on LAD[3:0]) that indicates that memory device 108 may drive a relatively large number of wait states onto LAD[3:0] (e.g., approximately 20+ wait states). The first control information may also have an encoding (e.g., 0101 on LAD[3:0]) that indicates that memory device 108 may drive a relatively small number of wait states onto LAD[3:0] (e.g., approximately 1-20 wait states). For one example (see Table 4 below), memory device 108 may drive synchronization data of 0101 on LAD[3:0], respectively, for approximately 1-20 clock cycles and then drive one cycle of 0000 on LAD[3:0], respectively, to indicate that the requested data is ready and will be output on the following clock cycle.

If the number of wait states is large, host 120 may decide to abort the transaction by entering state 320. As previously described, host 102 may abort the transaction by asserting LFRAME# for one or more clock cycles and driving predetermined abort control information (e.g., 1111 as in Table 1) on LAD[3:0].

The synchronization field of control information may also be used by memory device 108 to send an inband an error message on LAD[3:0] to LPC interface 118 of host 102. The error message may be sent at any time during the synchronization sequence. For example, it may be sent as the second control information in the embodiment having first and second synchronization control information. The error message may indicate, for example, that the data requested is corrupted is some way, the peripheral device did not understand the

request, an invalid request was made, or that a request has been while the peripheral device is in a power down mode or is in another mode in which it is not ready or able to output data. For one embodiment, if an error message is sent during a memory read transaction, the data is read out and disregarded.

Table 4 provides one embodiment of encodings on LAD[3:0] for the synchronization control information described above. Other encodings may also be used.

Table 4

LAD[3:0]	Indication
0000	Ready
0001 - 0100	Reserved
0101	Short Wait
0110	Long Wait
0111 - 1000	Reserved
1001	DMA Ready More
1010	Error
1011	Reserved
1100	Ready No More Sync
1101 - 1111	Reserved

For alternative embodiments, the inband synchronization control information may be used between any two devices (e.g., a host and peripheral device) over a general purpose time-multiplexed bus in any system to communicate wait state, error, or any other information previously described or summarized in Table 4.

When memory device 108 is ready to output its data to LPC bus 124, after providing the appropriate synchronization control information on LAD[3:0] (e.g., 0000), the process transitions to state 316 for n clock cycle(s) T6. In state 316, host 102 reads data provided by memory device 108 onto LAD[3:0]. Memory device 108 may drive data onto LAD[3:0] in byte increments over two clock cycles. For one

embodiment of a two byte transfer, a first byte may be output in the first two clock cycles, and a second byte may be output in the following two clock cycles. For each byte, data may be output by memory device 108 low nibble first, high nibble first, or in any other format. Similarly, each group of bytes output by memory device 108 may be output low byte first, high byte first, or in any other format.

For one embodiment, multiple byte reads may be implemented by cycling between states 316 and 314 such that a field of synchronization control information is inserted between bytes. For an alternative embodiment, a burst read of multiple bytes may be read while the process is in state 316. In the alternative embodiment, memory device 108 may drive 1100 on LAD[3:0] in state 314 to indicate that no more synchronization control information will be transferred between data bytes. For yet another embodiment, no particular synchronization control information may be needed to indicate a multiple or burst data read.

After all of the data has been read by host 102, the process transitions to turn around (TAR) state 318 in which control of LAD[3:0] may be returned to host 102 during n clock cycle(s) T7. As in state 312, state 318 may be two clocks wide. In the first clock cycle memory device 108 may drive LAD[3:0] lines with predetermined control information (e.g., 1111). In the second clock cycle, memory device 108 may tri-state LAD[3:0] lines. For other embodiments, n may be one clock cycle or greater than two clock cycles to turn around LAD[3:0].

An exemplary memory write LPC transaction by host 102 to a memory location in memory device 108 will be described with reference to Figures 3 and 5. This memory write LPC transaction may be implemented, for example, by host 102 to write system start-up information to memory device 108. At state 302, host 102 is in an idle state not performing an LPC transaction. The process commences an LPC transaction (e.g., when control logic 202 receives decoded signals from LPC decode logic 210 and host 102 indicating that a memory read

LPC transaction is to occur) when host 102 asserts LFRAME# on LPC bus 124. The process then transitions to start state 304 and host 102 drives start control information on LAD[3:0] indicating the start of an LPC transaction. For example, the start control information may be 0000 as indicated in Table 1. The process then transitions to state 304 and drives cycle type and direction control information on LAD[3:0] during cycle T1 that indicates to peripherals 104-108 that the LPC transaction is a memory write transaction. For example, the cycle type and direction control information may be 011 on LAD[3:1] as indicated in Table 2.

The process then transitions to state 308 and host 102 drives the address of the memory location in selected memory device 108 on LAD[3:0] during n clock cycle(s) T2. As previously indicated, the address may be any length of bits supported by a system such as system 100 of Figure 1, and sent over of number of n clock cycles T2. The address information may be provided most significant nibble first, least significant nibble first, or in any other order expected by memory device 108.

For an alternative embodiment, the process host 102 may then drive size control information on LAD[3:0] during cycle T3. The size control information may indicate the number of bytes of memory data to be written to memory device 108 starting at the address provided in state 308.

The process then transitions to state 310 and host 102 writes the appropriate bytes of data in n clock cycle(s) T4 to memory device 108 over LAD[3:0]. Host 102 may drive data onto LAD[3:0] in byte increments over two clock cycles. For example, if the transfer is a 16 bit or two byte transfer, then a first byte may be output in the first two clock cycles, and a second byte may be output in the following two clock cycles. For each byte, data may be output by host 102 low nibble first, high nibble first, or in any other format. Similarly, each group of bytes output by host 102 may be output low byte first, high byte first, or

in any other format. Host 102 may write one or more (burst) bytes of data in state 310.

The process then transitions to turn around (TAR) state 312 to turn control of LPC bus 124 over to memory device 108 during n clock cycle(s) T5. The process then transitions to state 314 and host 102 waits n clock cycles while memory device 108 drives inband synchronization control information on LAD[3:0] during clock cycle(s) T6. For one embodiment, the synchronization control information may be one or more clock cycles and used to acknowledge the receipt of the data written into memory device 108 (e.g., 0000 on LAD[3:0]). The synchronization control information may also be used to add wait states until the data has been written into and stored by memory device 108. During these wait states, memory device 108 may drive short wait state synchronization control information or long wait state synchronization control information on LAD[3:0] as indicated in Table 4. When memory device 108 has written the data into the addressed memory location, memory device 108 may drive a ready state on LAD[3:0] (e.g., 0000).

As with a memory read LPC transaction, memory device 108 may also send an inband error message over LAD[3:0] (e.g., 1010) to host 102 using the synchronization field of control information. The error message may indicate, for example, that the write transaction did not occur correctly or that some other error has occurred.

Host 102 may then transition to turn around (TAR) state 318 in which control of LAD[3:0] may be returned to host 102 during n clock cycle(s) T7.

Figure 6 is another embodiment of a state diagram that may be implemented one or more of LPC interfaces 112-118 to perform memory and/or I/O write or read LPC transactions between host 102 and peripherals 104-108. For example, the state diagram of Figure 6 may be used by a state machine or sequencer of memory and I/O LPC control logic 202 of Figure 2. This state diagram implements memory

read and memory write LPC transactions. The memory read and write LPC transactions will be described with the aide of exemplary timing diagrams of Figures 7 and 8.

An exemplary memory read LPC transaction by host 102 from a memory location in memory device 108 will first be described with reference to Figures 6 and 7. This memory read LPC transaction may be implemented, for example, by host 102 to read system start-up information from memory device 108. At state 602, LPC interface 118 is in an idle state not performing an LPC transaction when LFRAME# deasserted (e.g., high). The process transitions to state 604 when host 102 commences an LPC transaction (e.g., when control logic 202 receives decoded signals from LPC decode logic 210 and host 102 indicating that a memory read LPC transaction is to occur) by asserting LFRAME# (e.g., low) on LPC bus 124. Host 102 may drive start control information indicating the start of an LPC memory read transaction on LAD[3:0] during cycle T0. For example, the start control information may be 1101 on LAD[3:0] as indicated in Table 1.

The process then transitions to device select (IDSEL) state 606 and host 102 drives device select control information on LAD[3:0] to select a particular memory device for the memory read transaction. For one embodiment, memory devices coupled to LPC bus 124 may have external hard-wired or internal register-based identifications. Figure 9 shows one embodiment of hard wiring memory device identifications. Memory device 900 has an identification of 0000 on inputs I0, I1, I2, and I3, respectively. Memory device 901 has an identification of 0001 on inputs I0, I1, I2, and I3, respectively. Memory device 915 has an identification of 1111 on inputs I0, I1, I2, and I3, respectively. Any number of hard-wired inputs may be used to identify or select any number of memory devices using a binary format or any other format. Each memory device coupled to LPC bus 124 compares the device select control information with its hard-wired identification to determine if the LPC memory read (or write)

transaction is for them. If the memory transaction is not for them, the memory device may enter a power down state (e.g., idle state 602).

The process then transitions to state 608 and host 102 drives the address of the memory location in selected memory device 108 on LAD[3:0] during n clock cycles T2. The address may be any length of bits supported by a system such as system 100 of Figure 1, and sent over a number of n clock cycles T2. For one embodiment, the memory address space may be 256 KBytes requiring 28 bits of memory address. For this embodiment, seven clock cycles of address information are provided by control logic to LPC bus 124 to transfer 7 nibbles (28 bits) of address information from host 102 to memory device 108. The address information may be provided most significant nibble first, least significant nibble first, or in any other order expected by memory device 108.

The process then transitions to state 610 and host 102 drives size control information on LAD[3:0] during clock cycle T3. For one embodiment, the size control information indicates the number of bytes of memory data to be read from memory device 108 starting at the address provided in state 606. Table 6 provides one embodiment of encoding the number of bytes. For another embodiment, the size control information may indicate that 2^{size} number of bytes of memory data are to be read from memory device 108 starting at the address provided in state 608.

For one embodiment, data transfers may be aligned to 2^{size} boundary by host 102. For example, if the size field is equal to 0100 on LAD[3:0], then the operation is aligned to a 16 byte boundary and memory device 108 should accept or send 16 bytes of data beginning from a 16 byte aligned address. For another embodiment, data transfers may be aligned to a size boundary as in Table 3. For example, if size is equal to 0010 on LAD[3:0], then the operation is aligned to a 16 bit or word boundary and memory device 108 should accept or send 16 bits of data beginning from a 16 bit or word aligned address.

The process then transitions to turn around (TAR) state 614 to turn control of LPC bus 124 over from host 102 to memory device 108 during n clock cycle(s) T4. The process then transitions to state 616 and in which host 102 waits n clock cycles T5 while memory device 108 drives inband synchronization control information on LAD[3:0]. As described above with respect to state 314 of Figure 3, the synchronization control information may be used to add wait states to the memory transaction. If the number of wait states is large, host 120 may decide to abort the transaction by entering state 622. As previously described, host 102 may abort the transaction by asserting LFRAME# for one or more clock cycles and driving predetermined abort control information (e.g., 1111 as in Table 1) on LAD[3:0].

As previously described with respect to state 314 of Figure 3, the synchronization field of control information may also be used by memory device 108 to send an inband an error message on LAD[3:0] to LPC interface 118 of host 102. Table 4 above provides one embodiment of encodings on LAD[3:0] for the synchronization control information described above. Other encodings may also be used.

For alternative embodiments, the inband synchronization control information may be used between any two devices (e.g., a host and peripheral device) over a general purpose time-multiplexed bus in any system to communicate wait state, error, or any other information previously described or summarized in Table 4.

When memory device 108 is ready to output its data to LPC bus 124, after providing the appropriate synchronization control information on LAD[3:0] (e.g., 0000), the process transitions to state 618 for n clock cycle(s) T6. In state 618, host 102 reads data provided by memory device 108 onto LAD[3:0]. Memory device 108 may drive data onto LAD[3:0] in byte increments over two clock cycles. For each byte, data may be output by memory device 108 low nibble first, high nibble first, or in any other format. Similarly, each group of bytes output by

memory device 108 may be output low byte first, high byte first, or in any other format expected by host 102.

For one embodiment, multiple byte reads may be implemented by cycling between states 618 and 616 such that a field of synchronization control information is inserted between bytes. For an alternative embodiment, a burst read of multiple bytes may be read while host 102 is in state 618. In the alternative embodiment, memory device 108 may drive 1100 on LAD[3:0] in state 616 to indicate that no more synchronization control information will be transferred between data bytes. For yet another embodiment, no particular synchronization control information may be needed to indicate a multiple or burst data read.

After all of the data has been read by host 102, the process transitions to turn around (TAR) state 620 in which control of LAD[3:0] may be returned to host 102 during n clock cycle(s) T7. As in state 614, state 620 may be two clocks wide. In the first clock cycle memory device 108 may drive LAD[3:0] lines with predetermined control information (e.g., 1111). In the second clock cycle, memory device 108 may tri-state LAD[3:0] lines. For other embodiments, n may be one clock cycle or greater than two clock cycles to turn around LAD[3:0].

An exemplary memory write LPC transaction by host 102 to a memory location in memory device 108 will be described with reference to Figures 6 and 8. This memory write LPC transaction may be implemented, for example, by host 102 to write system start-up information to memory device 108. At state 602, host 102 is in an idle state not performing an LPC transaction. The process commences an LPC transaction (e.g., when control logic 202 receives decoded signals from LPC decode logic 210 and host 102 indicating that a memory read LPC transaction is to occur) when host 102 asserts LFRAME# on LPC bus 124. The process then transitions to start state 604 and host 102 drives start control information on LAD[3:0] during clock cycle T0. The start control information indicates the start of a memory write LPC

transaction. For example, host 102 may drive start control information of 1110 on LAD[3:0] as indicated in Table 1. The process then transitions to device select (IDSEL) state 606 to identify or select a particular memory device for the memory write transaction.

The process then transitions to state 608 and host 102 drives the address of the memory location in selected memory device 108 on LAD[3:0] during n clock cycle(s) T2. As previously indicated, the address may be any length of bits supported by a system such as system 100 of Figure 1, and sent over of number of n clock cycles T2. The address information may be provided most significant nibble first, least significant nibble first, or in any other order expected by memory device 108.

The process then transitions to state 610 and host 102 drives size control information on LAD[3:0] during clock cycle T3. The process then transitions to state 612 and host 102 writes the appropriate bytes of data in n clock cycle(s) T4 to memory device 108 over LAD[3:0]. Host 102 may drive data onto LAD[3:0] in byte increments over two clock cycles. For example, if the transfer is a 16 bit or two byte transfer, then a first byte may be output in the first two clock cycles, and a second byte may be output in the following two clock cycles. For each byte, data may be output by host 102 low nibble first, high nibble first, or in any other format. Similarly, each group of bytes output by host 102 may be output low byte first, high byte first, or in any other format. Host 102 may write one or more (burst) bytes of data in state 612.

The process then transitions to turn around (TAR) state 614 to turn control of LPC bus 124 over to memory device 108 during n clock cycle(s) T5. The process then transitions to state 616 and host 102 waits n clock cycles while memory device 108 drives inband synchronization control information on LAD[3:0] during clock cycle(s) T6. For one embodiment, the synchronization control information may be one or more clock cycles and used to acknowledge the receipt of the data written into memory device 108 (e.g., 0000 on LAD[3:0]). The

synchronization control information may also be used to add wait states until the data has been written into and stored by memory device 108. As with a memory read LPC transaction, memory device 108 may also send an inband error message over LAD[3:0] (e.g., 1010) to host 102 using the synchronization field of control information. The error message may indicate, for example, that the write transaction did not occur correctly or that some other error has occurred. One embodiment of the synchronization control information encodings is shown in Table 4.

Host 102 may then transition to turn around (TAR) state 618 in which control of LAD[3:0] may be returned to host 102 during n clock cycle(s) T7.

Figure 10 shows memory device 1000 that is one embodiment of memory device 108. Memory device 1000 includes memory 1010 that may be any type of memory device including volatile and non-volatile memory devices such as SRAM, DRAM, EPROM, EEPROM, ROM, and the like. Memory 1000 includes the control circuitry and memory elements necessary for implementing any volatile or nonvolatile memory device. For one embodiment, memory 1000 is a nonvolatile (e.g., flash) memory device having a control engine coupled to and controlling the programming, reading, and erasing of an array of flash memory cells.

Memory device 1000 also includes one embodiment of LPC interface 116 having I/O latches 1002, control logic 1004, state machine 1006, and memory control logic 1008. Control logic 1004 receives LPC input signals LFRAME# LCLK, and/or LRESET# and provide an indication of these signals via control signal(s) on line(s) 1012 to state machine or sequencer 1006. For one embodiment, I/O latches 1002 receive and output address, control, and data information from and to LAD[3:0]. I/O latches 1002 pass the address, data, and control information received from LAD[3:0] to state machine 1006 and

memory control logic 1008 via bus 1014. I/O latches 1002 may concatenate the various nibbles of data received from LAD[3:0].

With the address, data, and control information received from bus 1014 and line(s) 1012, state machine 1006 controls the LPC interface of memory device 1000 such that memory device 1000 may communicate with host 102 over LPC bus 124, or may communicate with I/O devices 104 and 1010 in a bus master LPC transaction. One embodiment of state machine 1006 is shown in Figure 3. Another embodiment of state machine 1006 is shown in Figure 6.

State machine 1006 may work with memory control logic 1008 to provide the necessary control, address, and data information to memory 1010. Memory control logic 1008 generates the necessary control signals on line(s) 1018, address signals on line(s) 1020, and data signals on line(s) 1022 for memory 1010 to perform write or read transactions. The control signals on line(s) 1018 may include address strobes (e.g., RAS and/or CAS), chip enable or chip select signals (e.g., CE and/or CS), write or read enable signals, reset signals, power down signals, and the like.

When state machine 1006 determines that a memory write transaction is to occur, state machine 1006 may indicate via line(s) 1016 when address information is present on bus 1014 for memory control logic to provide over line(s) 1020 to memory 1010 along with appropriate control signals on line(s) 1018. Subsequently, state machine 1006 may indicate via line(s) 1016 when the data information is preset on bus 1014 for memory control logic to provide over line(s) 1022 to memory 1010 along with appropriate control signals on line(s) 1018. Similarly, when state machine 1006 determines that a memory read transaction is to occur, state machine 1006 may indicate via line(s) 1016 when address information is present on bus 1014 for memory control logic to provide over line(s) 1020 to memory 1010 along with appropriate control signals on line(s) 1018. Subsequently, state machine 1006 may indicate via line(s) 1016 when the requested data

should be output memory 1010 via line(s) 1026 to I/O latches 1002 and LAD[3:0]. Memory control logic 1008 may issue one or more I/O control signals on line(s) 1024 to control the bi-directional state of I/O latches 1002. For example, memory control logic 1008 may assert an output enable signal over line 1024 when data is read from memory 1010.

In the foregoing specification, the invention has been described with reference to specific exemplary embodiments thereof. It will, however, be evident that various modifications and changes may be made thereof without departing from the broader spirit and scope of the invention as set forth in the appended claims. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.



CLAIMS

What is claimed is:

1. A system comprising:

a host;

a bus coupled to the host and having a plurality of general purpose signal lines to carry time-multiplexed address, data, and control information; and

a memory device coupled to the bus and storing system start-up information, wherein the memory device communicates the system start-up information with the host over the bus.

2. A system comprising:

a bus comprising a plurality of general purpose signal lines to carry time-multiplexed address, data, and control information;

a memory device coupled to the bus and storing system start-up information; and

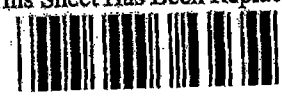
a host coupled to the bus and writing first control information to the memory device via the bus to indicate a memory transaction in which a part of the system start-up information is communicated between the host and the memory device.

3. The system of claim 2, wherein the bus further comprises an independent control signal line for carrying a control signal that indicates a beginning of the memory transaction.

4. The system of claim 3, wherein the control signal on the independent control signal line indicates aborting the memory transaction.

5. The system of claim 2, wherein the bus includes an independent clock signal line for carrying a clock signal, and wherein the memory

This Sheet Has Been Replaced





device synchronously communicates the system start-up information with the host with respect to the clock signal.

6. The system of claim 2, wherein the host writes second control information to the memory device via the bus to indicate a memory address in the memory device.

7. The system of claim 6, wherein the host writes third control information to the memory device via the bus to indicate a number of bytes to be transferred in the memory transaction.

8. The system of claim 7, wherein the host writes fourth control information to the memory device via the bus to turn control of the bus over to the memory device.

9. The system of claim 8, wherein the memory device provides fifth control information to the bus to synchronize operation of the memory device with the host.

10. The system of claim 9, wherein the memory device provides the fifth control information to the bus until data is ready to be output to the bus by the memory device.

11. The system of claim 9, wherein the fifth control information comprises an indication of an error in the memory transaction.

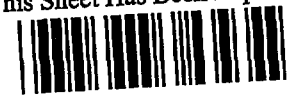
12. The system of claim 2, wherein the host comprises a processor.

13. The system of claim 2, wherein the host comprises a chip set.

14. The system of claim 2, wherein the plurality of general purpose signal lines comprises four general purpose signal lines.

15. The system of claim 2, wherein the system comprises a communication system.

This Sheet Has Been Replaced



16. The system of claim 2, wherein the bus further comprises an independent control signal line for carrying a control signal that indicates a low power mode for the memory device.

17. A method of performing a memory transaction between a host and a memory device across a bus comprising a plurality of general purpose signal lines and a separate control line, wherein the memory device stores system start-up information, the method comprising the steps of:

asserting a control signal on the control line to indicate the start of the memory transaction;

writing first control information from the host to the memory device on the plurality of general purpose signal lines to indicate that the memory transaction comprises the host writing a unit of data to the memory device;

writing second control information from the host to the memory device on the plurality of general purpose signal lines to indicate an address in the memory device to which the unit of data will be written;

writing third control information from the host to the memory device on the plurality of general purpose signal lines to indicate a number of bytes in the unit of data to be transferred; and

writing the unit of data to the memory device via the general purpose signal lines.

18. The method of claim 17, further comprising the steps of:

turning control of the bus over to the memory device;

providing fourth control information from the memory device to the bus to indicate a length of time before the memory device will turn control of the bus over to the host; and

turning control of the bus over to the host.

This Sheet Has Been Replaced



19. The method of claim 17, further comprising the step of writing third control information from the memory device to the host on the plurality of general purpose signal lines to indicate an error in the memory transaction.

20. A method of performing a memory transaction between a host and a memory device across a bus comprising a plurality of general purpose signal lines and a separate control line, wherein the memory device stores system start-up information, the method comprising the steps of:

asserting a control signal on the control line to indicate the start of the memory transaction;

writing first control information from the host to the memory device on the plurality of general purpose signal lines to indicate that the memory transaction comprises the host reading a unit of data from the memory device;

writing second control information from the host to the memory device on the plurality of general purpose signal lines to indicate an address in the memory device from which the unit of data will be read;

writing third control information from the host to the memory device on the plurality of general purpose signal lines to indicate a number of bytes in the unit of data to be read;

turning control of the bus over to the memory device;

writing fourth control information from the memory device to the host on the plurality of general purpose signal lines until the unit of data is ready to be read from the memory device; and

reading the unit of data from the memory device via the general purpose signal lines.

This Sheet Has Been Replaced



21. The method of claim 20, wherein the fourth control information comprises an indication of an error in the memory transaction.

22. A system comprising:

a bus;

a peripheral device coupled to the bus;

a host coupled to the bus; and

an independent control signal line for carrying a control signal that indicates a beginning of a transaction between the host and the peripheral device.

23. The system of claim 22, wherein the control signal line on the independent control signal line indicates aborting the transaction.

24. The system of claim 22, wherein the bus comprises a plurality of general purpose signal lines to carry time-multiplexed address, data, and control information between the host and the device.

25. The system of claim 22, wherein the peripheral device comprises a memory device.

26. A system comprising:

a bus having a plurality of general purpose signal lines to carry time-multiplexed address, data, and control information;

a peripheral device coupled to the bus; and

a host coupled to the bus, wherein the peripheral device communicates wait state information over the bus to the host device.

27. The system of claim 26, wherein the peripheral device communicates an error message over the bus to the host.

This Sheet Has Been Replaced