A BUS CONTROLLING METHOD AND EQUIPMENT USING THE METHOD

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates to a bus controlling method and equipment. More particularly, in a case where a plurality of devices are accessed through a common bus, the present invention relates to a method for controlling a bus cycle for each device and to equipment using the method.

Description of the Related Arts

A variety of data processing equipment such as PCs usually have microprocessors which centrally control the entire equipment. A microprocessor is connected to a variety of memories and I/O devices through buses such as a local bus and a system bus. In recent microprocessors, a cache memory and a circuit regarding a communication function are often installed. However, for memories other than a cache memory or I/O devices, microprocessors access them by performing an ordinary bus cycle.

Fig. 4 is a timing chart of a bus cycle described on page 125 of "Hitachi single chip RISC microcomputers SH7032, SH7034, HD6417032, HD6477034, and HD6437034 hardware manuals (third edition)" . In this figure, CK denotes an operation clock of

the microcomputer, and A21~0 are addresses for addressing external devices. *CSn represents a chip select asserted when an nth space (hereafter simply called address space n) of address spaces partitioned into a plurality of spaces is accessed, and *RD is a read command which is active on a low signal level. AD15~0 mean data. Hereafter, a signal prefixed with * means the signal which is active on a low signal level.

In Fig. 4, one period of the clock CK is equivalent to each state T1, or T2. If the configuration is as simple as possible, each device is assigned to each address space n on a one to one basis, and the address space n becomes an exclusive space for the device. Therefore, each device has a connection with one chip select *CS.

One of the characteristics of the microcomputer in Fig. 4 is that it can change an assertion timing of a read command *RD. As shown in the figure, if RDDTY = 0, the assertion timing is set to be slightly later, while it is set to be slightly earlier if RDDTY = 1. The signal RDDTY can respectively be set for each address space n. According to the manual description, an access time to an external device can be set long by setting RDDTY = 1.

For the microcomputer described above, the assertion timing of the read command can respectively be set for each device to be accessed. This is a technique which takes an access

time to a device into consideration.

On the other hand, when timings regarding a bus cycle are designed, an output disable time of a device should be considered in some cases. An output disable time is the time from negation timing of a read command to the device to a timing when a data buffer of the device actually becomes OFF so that an output from that device completely floats. If this output disable time is long and the next bus cycle is a write cycle, for example, a conflict occurs on the bus between the output from the device and a write data. This leads to an increased power consumption and a lower long term reliability of the device. Even when the next bus cycle is a read cycle, outputs from two devices may conflict. The possibility of data conflict will be lowered if a read command is negated earlier. However, for an ordinary IC, the lowermost output disable time is set to fall within 0~several ns in the standard. Therefore, it is possible for read data to float before the data is taken in. It is generally risky to design a read command to be negated early. Therefore, design with consideration of the output disable time can be very troublesome in some cases.

In the case of the microcomputer shown in Fig. 4, an inactive time of a read command can be extended by making RDDTY = 0. For this reason, if the read cycle is repeated, output timing of read data from a device shifts later, and the

possibility of data conflict between the data and read data from a device previously selected is lowered. However, if a write cycle comes after a read cycle, a data conflict can not be avoided.

SUMMARY OF THE INVENTION

The present invention is created from consideration of the problem described above. The object of the present invention is to provide a bus controlling method and equipment for a plurality of devices with different output disable times, which prevents data conflicts from occurring while avoiding performance degradation. Another object of the present invention is to automatically perform such control using pre-set parameters or the like. Still another object of the present invention is to provide a bus controlling method and equipment which enables easier design of external circuits.

1. Regarding a bus controlling method

A principle to solve the problem described above is, for controlling a bus cycle BC_i, to refer to the immediately preceding bus cycle BC_{i-1}. In other words, based on a characteristic of a device accessed in BC_{i-1}, for example, an output disable time thereof, activation of BC_i is delayed if necessary after the end of BC_{i-1}. Here, "devices" means general targets to be accessed.

The problem of the prior art can be solved with the present invention, since a bus cycle control is performed in response to the immediately preceding bus cycle. One method to delay a bus cycle activation is to insert an idle state.

In one aspect of the present invention, a characteristic of a device to be accessed in BC_{i-1} is specified by referring to:

a. an address being output when each device is allocated in an exclusive address space, and

b. the number of wait cycles in the current bus cycle or the like.

According to the present invention, activation of BC_i is delayed only when it is necessary. Therefore, a data conflict can be avoided without leading to unnecessary degradation of processing performance. When the output disable time is considered as a characteristic of a device, the device with a long output disable time is often a low speed device. Therefore, when the present invention is implemented, a low speed device may be specified. This method has high practicality, although it is simple.

In another aspect of the present invention, if the device having output data in BC_{i-1} is also the device to output data in BC_i, BC_i is activated without delay. This is because no conflict occurs when data are output from the same single device.

Device here is meant to include a microprocessor and the like. Therefore, according to this aspect, when a microprocessor carries out a writing operation continuously to a device, BCi will not be delayed unnecessarily.

2. Regarding bus controlling equipment

The present invention comprises setting means for setting a parameter related to an output disable time of a device to be accessed, and bus controlling means for inserting an idle state after a current bus cycle in response to an output disable time of a device being accessed in the current bus cycle, based on the parameter set by the setting means. The "parameter related to the output disable time" is meant to include all parameters to judge or estimate not only the output disable time itself but also whether the output disable time is long or short.

With this configuration, the effect same as 1 described above can be realized for equipment. A work load for designing a circuit outside the equipment is also reduced, since this equipment considers output disable times by itself.

One aspect of the present invention comprises judging means for judging whether or not a device to output data in a current bus cycle is also a device to output data in the following bus cycle, and prohibition means for prohibiting insertion of the idle state after the current bus cycle if the devices are judged to be the same. As described in 1 above, no data conflict

occurs when one and the same device continuously outputs data. Unnecessary delays will be avoided by taking such cases into consideration.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a circuit configuration diagram of a microcomputer using a bus controlling equipment related to an embodiment of the present invention.

Fig. 2 is a diagram showing an internal configuration of a bus controlling unit of the embodiment.

Fig. 3 shows a timing chart example when an external device is accessed by the microcomputer of the embodiment.

Fig. 4 is a timing chart of a bus cycle described on page 125 of "Hitachi single chip RISC microcomputers SH7032, SH7034, HD6417032, HD6477034, and HD6437034 hardware manual (third edition)" .

DESCRIPTION OF THE PREFERRED EMBODIMENT

Hereafter, a preferred embodiment of the present invention will be explained referring to accompanying drawings.

[1] Configuration

Fig. 1 is a circuit configuration diagram of a microcomputer using a bus controlling device related to an embodiment of the present invention. In this figure, a

processor 2 constitutes the core logic of a microcomputer 1 and controls the entire microcomputer. The processor 2 has an address Ai and data Di. Moreover, the processor 2 outputs a read command RDi, a write command WRi and STARTi which shows a start timing of a bus cycle, all of which are active when the signal level is high, and temporarily supplies these signals to a bus controlling unit 4. The bus controlling unit 4 refers to the address A by itself, and outputs the address as it is to an external device. As for the data Di, the bus controlling unit 4 serves as a data transceiver and connects to data D which is an external data bus. Signals with suffix "i" added to their names mean internal signals of the microcomputer 1.

Based on the address A, the bus controlling unit 4 generates chip select signals *CS1, *CS2, through *CSn for a first device 6, a second device 8, through an nth device 10, all of which are external devices. Here, an exclusive address space is meant to be allocated for each device such as

the first device 6 : 1000h ~ 1FFFh

the second device 8 : 2000h ~ 2FFFh

the nth device 10 : n000h ~ nFFFh.

Therefore, if the address A shows "1234h" for example, it can be specified as an access to the first device 6.

When the bus controlling unit 4 judges that an output disable time of the device being accessed in the current bus

cycle BC_i is long, it inserts an idle state between the bus cycle BC_i and the following bus cycle BC_{i+1}. In the present embodiment, the processor 2 is not involved in generation of such idle states. If a bus cycle is being executed within the processor 2, the processor 2 will activate the following bus cycle at the earliest timing. On the other hand, the bus controlling unit 4 delays activation of a bus cycle for the external devices when an idle state should be inserted. Therefore, a bus cycle may be being executed within the processor 2, although the processor 2 appears to be in an idle state when viewed from outside the processor 2.

The bus controlling unit 4 latches a read command RDi, a write command WRi, and a cycle start signal STARTi, all of which are supplied from the processor 2. The bus controlling unit 4 then respectively asserts the latched signals as signals for the external devices *RD, *WR, and START, immediately after the idle state is finished. During this process, a wait signal WAITi is asserted by the processor 2 in order to suspend the processor 2.

The first, second, and nth devices 6, 8, and 10 are connected to the address A, a bus for the data D, *RD, *WR, START, and corresponding chip selects *CS1, *CS2, and *CSn. For the first device 6 for example, if *CS1 and *RD are asserted together, data corresponding to the address A at this time is read out.

When *CS1 and *WR are asserted together, a write data from the processor 2 is written on a device specified by the address A. The START signal may not necessarily be referred to, depending on a device to be accessed.

Fig. 2 shows an internal configuration of the bus controlling unit 4. In Fig. 2, RDi, WRi, and STARTi are input into a timing control unit 20. An idle signal IDLEi which will be described later is also input into the timing control unit 20. When this signal is negated, *RD, *WR, and START are asserted by the devices outside the microcomputer 1.

First, second, and nth comparators 21, 22, and 2n take the address A in, and respectively compare this address to exclusive addresses 1xxxh, 2xxxh, and nxxxh for the first, second and the nth devices 6, 8, and 10. When they coincide with each other, coincidence signals CS1i, CS2i, and CSni are asserted, and input into a positive logic input terminal of NAND gates 51, 52, and 5n.

First, second, nth wait registers 31, 32, and 3n (respectively expressed as "first W", "second W", and "nth W" in Fig. 2) store wait numbers to be inserted upon access to the first, second, and nth devices 6, 8, and 10. The wait numbers are set by data writing to these registers. A wait selector 60 (expressed as "W selector" in Fig. 2) is supplied with the coincidence signals and outputs from the first, second, and nth

wait registers 31, 32, and 3n. For example, if CS1i is asserted, the wait number corresponding to the device, that is, an output from the first wait register 31 is selected and output to a wait timer 62 (expressed as "W timer" in Fig. 2). The wait timer 62 keeps asserting the wait signal WAITi over the number of clock periods corresponding to the input wait number, and suspends cycle advancement in the processor 2. To keep the processor 2 from advancing to the next bus cycle while an idle state is being inserted, the wait timer 62 keeps asserting WAITi while the idle signal IDLEi which will be described later is being asserted.

On the other hand, first, second, and nth output disable time registers 41, 42, and 4n (expressed as "first D", "second D" and "nth D" in Fig. 2) store parameters regarding the output disable time of the first, second, and nth devices 6, 8, and 10. These parameters can be set using software. When a maximum output disable time T_{dmax} is adopted as the parameter, T_{dmax}(ns) can be specified in 4 ways as follows, if the output disable time registers respectively have 2 bits:

when T_{dmax} can be regarded as $T_{dmax} = 0$ ..."00" (default)

when $10 < T_{dmax} < 29$ ns ..."01"

when $30 < T_{dmax} < 49$ ns ..."10"

when $50 < T_{dmax}$..."11".

An idle selector 64 (expressed as "I selector" in Fig.

2) is supplied with the coincidence signals and outputs from the first, second, and nth output disable time registers 41, 42, and 4n (hereafter, also called T_{dmax} codes). For example, when CS1i is being asserted, output from the first output disable time register 41 is selected and output to an idle timer 66 (expressed as "I timer" in Fig. 2). Based on the T_{dmax} code, the idle timer 66 first calculates the number of clock periods which can cover the output disable time. If the bus clock frequency is 20 MHz, 1 idle state is equivalent to 50 ns. Therefore, the number of clock periods is calculated as follows, based on 4 T_{dmax} codes described above:

If the T_{dmax} code is "00", then 0 clock period.

If the T_{dmax} code is "01" or "10", then 1 clock period.

If the T_{dmax} code is "11", then 2 clock periods.

If the maximum output disable time exceeds 50 ns, in other words, if the T_{dmax} code is "11", the exact number of clock periods is not known. However, 100 ns is generally sufficient, and the number of clock periods is fixed to 2 here.

The idle timer 66 then keeps asserting a signal IDLEi0 over this number of clock periods. To make assertion activation of IDLEi0 standby until the current bus cycle ends, the idle timer 66 refers to WAITi, and keeps itself from advancing while WAITi is being asserted. IDLEi0 is input to a positive logic input terminal of an AND gate 70.

An idle mask circuit 68 (expressed as "I mask circuit" in Fig. 2) takes in RDi, WRi, CS1i, CS2i, and CS2ni. The I mask circuit detects a state in which one and the same device consecutively drives the data bus for two bus cycles, and asserts an IMASK signal. This is because when one and the same device uses a data bus, data conflict can not exist in that case. Concrete assertion conditions of the IMASK signal are as follows:

when WRi is consecutively asserted in two bus cycles.

when RDi is consecutively asserted in two bus cycles, and the corresponding coincide signals CS1i or the like are the same in the two bus cycles.

The former corresponds to the case where the processor 2 consecutively drives the data bus , while the latter is the case where the same external device consecutively drives the data bus. The IMASK signal is input into a negative logic terminal of the AND gate 70, while an idle signal IDLEi which is an output from the AND gate 70 is input into negative logic terminals of NAND gates 51, 52, and 5n as well as into the timing control unit 20.

A transceiver 72 connects buses data Di and data D. The IDLEi signal is input into an enable terminal *EN of the transceiver 72. As long as IDLEi is being negated, the transceiver 72 is in an enable state. The RDi signal is input

into an data direction DIR, and as long as RDi is being asserted, data is directed from the outside to the inside, in other words, data D is output to data Di.

[2] Operation

Fig. 3 is a timing chart when an external device is accessed by the microcomputer 1. Numbers T1 through T15 are to show the clock period which serves as a reference of a bus operation. In Fig. 3, the following 3 bus cycles are depicted:

BC_1...data read (1000h) from the first device 6.

BC_2...data write (1004h) to the first device 6.

BC_3...data write (2000h) to the second device 8.

Regarding the first device 6, the number of the idle states is "2", while the number of wait states is "3". For the second device 8, the number of wait states is "2".

(1) BC_1

The processor 2 activates the first bus cycle BC_1 at the beginning of T1. STARTi is asserted at the beginning of T1, and START is asserted by the external device via the timing control unit 20. "1000h" appears on the address A, and *CS1 is asserted, since CS1i is asserted by the first comparator 21. *RD is also asserted in response to the assertion of RDi. The first device 6 starts driving the data D in the middle of the period T1 when *RD and *CS1 are both asserted. However, since an access time is long, an efficient read data RDATA appears

in T4 on the bus. Since IDLEi is not asserted in BC_1, the transceiver 72 is in the enable state, and the data D appears on the data Di while *RD is being asserted.

Following the assertion of CS1i, the number of waits "3" having been stored in the first wait register 31 is selected by the wait selector 60 and sent to the wait timer 62. WAITi is kept asserted by the wait timer 62 over 3 clock periods, and the count of wait is decreased by 1 at the end of each clock period. BC_1 ends at the end of the period T4 when the number of wait becomes 0. However, since the output disable time of the first device 6 is long, output of the read data on data D continues until near T6.

As for the idle state, following the assertion of CS1i, the number of idle states "2" having been stored in the first output disable register 41 is selected by the idle selector 64 and sent to the idle timer 66. The idle timer 66 keeps asserting IDLEi for 2 clock periods following the clock period in which WAITi is negated.

(2) BC_2

The processor 2 confirms that WAITi is negated at the end of T4, and starts the next bus cycle BC_2 by asserting STARTi at T5. However, since IDLEi is being asserted at this time, an idle state is inserted for the external bus. The idle state I corresponds to T5 and T6. During these periods, the

transceiver 72 is also disabled, and the data D and data Di are separated. As a result, a conflict is avoided between write data WDATA from the processor 2 appearing on the data Di and the read data RDATA from the first device 6 still remaining on the data D.

While IDLEi is being asserted, assertion of *WR and START is kept on standby in the timing control unit 20, and assertion of *CS1 is kept on standby at the NAND gate 51. These signals are asserted by the external devices in T7 when IDLEi is negated. While IDLEi is being asserted, count-down of the wait timer 62 is also suspended, and WAITi is kept asserted. The wait timer 62 starts the count-down after IDLEi is negated, and keeps asserting WAITi over 3 clock periods. The cycle ends in T10. Therefore, the processor 2 itself activates the bus cycle in T5, while the bus cycle is activated in T7 when viewed from the outside of the microcomputer. The number of waits is 3 for the bus cycle, while the number of waits for the processor is 5.

(3) BC_3

Since WRi is consecutively asserted in BC_2 and BC_3, the idle mask circuit 68 asserts IMASK signal. As a result, IDLEi which is an output from the AND gate 70 is negated, and BC3 becomes an ordinary bus cycle before which no idle state was inserted. BC_3 ends in T13 according to the number of waist "2" having been set in the second wait register 32. T14 is a natural

idle state which occurs when no following bus cycle is requested within the processor 2.

According to the embodiment described above, data conflicts can be avoided without unnecessary degradation of performance, since an idle state is only inserted when necessary. A designer of an external circuit will no longer be hampered by design regarding the output disable time.

[3] Modifications

Regarding the present embodiment, modifications such as those shown below are possible.

(1) The processor 2 and the bus controlling unit 4 are meant to be installed in the single microcomputer 1 here. However, they can of course be separated in different configurations. In that case, the present invention can be applied as an external circuit of a known microprocessor.

(2) T_{dmax}, which is the maximum output disable time, is considered as the parameter stored in the first output disable time register 41 or the like. However, a desired number of idle states itself may be used as the parameter. In that case, since the register has 2 bits, an idle state number of 0 to 3 can be specified, for example.

(3) The idle state number may be estimated using the wait number. Since a low speed device with a larger number of waits generally has longer output disable time, it is possible to

allow the idle state to be 1 when the wait number exceeds 4, and 2 when the wait number exceeds 10, for example. In that case, the circuit for the first output disable time register 41 or the like is not necessary, and it will be sufficient to have a table or the like to store a relationship between the output from the first wait register 31 or the like and the idle state number to be inserted.

(4) In the present embodiment, output disable time has been explained. It is of course possible, however, to delay a following bus cycle start depending on a device characteristic other than the output disable time. For example, if one device is read once and a recovery time is necessary before the next read, a recovery time may be set instead of the output disable time of the present embodiment.

While there has been described what is at present considered to be a preferred embodiment of the invention, it will be understood that various modifications can be made thereto, and it is intended that the appended claims cover all such modifications as fall within the true spirit and scope of the invention.

This Sheet Has Been Replaced

What is claimed is:

1. A bus controlling method which delays activation of a bus cycle following a current bus cycle by considering a characteristic of a device being accessed in the current bus cycle.

2. The bus cycle controlling method according to Claim 1 which delays the following bus cycle start by inserting an idle state after the current bus cycle.

3. The bus cycle controlling method according to Claim 1 or 2 which specifies the characteristic of the device being accessed in the current bus cycle by an address currently being output, when each device is allocated in an exclusive address space.

4. The bus cycle controlling method according to Claim 1 or 2 which specifies the characteristic of the device being accessed in the current bus cycle based on a number of waits in the current bus cycle.

5. The bus cycle controlling method according to any one of the Claims 1 to 4 which activates the following bus cycle without delay when a device to output data in the current bus cycle and

* C L M S *

THIS SHEET HAS BEEN REPLACED

a device to output data in the following bus cycle are one and the same.

6. The bus cycle controlling method according to any one of the Claims 1 to 5 wherein the characteristic is an output disable time.

7. Bus controlling equipment which comprises:

setting means for setting a parameter related to an output disable time of a device to be accessed, and

controlling means for inserting an idle state after a current bus cycle, in response to an output disable time of a device being accessed in the current bus cycle, based on the parameter set by the setting means.

8. The bus controlling equipment according to Claim 7 which further comprises:

judging means for judging whether or not a device to output data in the current bus cycle and a device to output data in the following bus cycle are one and the same, and

prohibition means for prohibiting insertion of the idle state after the current bus cycle if the devices are judged to be one and the same.

This Sheet Has Been Replaced

CLMS

S SHEET HAS BEEN REPLACED