

0462755296

JP9-2000-0383-JP1

DMA TRANSFER CONTROL METHOD AND CONTROL DEVICE

BACKGROUND OF THE INVENTION

Field of the Invention



159159

The present invention relates to an apparatus and method for controlling a direct memory access (DMA) transfer, and more particularly to an apparatus and method for controlling a memory-to-memory DMA transfer.

Description of Related Art

FIGs. 9(a) to 9(d) are schematic diagram of a memory-to-memory DMA transfer performed between memory areas in a main memory. In a data processing system shown in FIGs. 9(a) to 9(d), a central processing unit (CPU) 70, a cache unit 72, a main memory 90 and a DMA controller 82 are connected to a system bus 80.

The cache unit 72 is connected between the CPU 70 and the system bus 80 and contains a tag 74 and a cache 76. The cache 76 is a memory for temporarily storing a part of data stored in the main memory 90, and the tag 74 is a memory for storing addresses of the data stored in the cache 76. The CPU 70 accesses the cache 76 when the cache 76 stores a target data, while it accesses the main memory 90 when the cache 76 does not store the target data.

The main memory 90 is a semiconductor memory such as a dynamic random access memory (RAM). The DMA controller 82 performs a memory-to-memory data transfer between memory areas in the main



G00002

memory 90 without intervention by the CPU 70. A data transfer between the memory areas means a data transfer from a source memory location to a destination memory location in the main memory 90, as shown in FIGs. 9(a) to 9(d). A data transfer using the DMA controller 82 (hereinafter referred to as DMA data transfer) and processing in the CPU 70 can be performed in parallel.

For example, when a plurality of instructions or instruction groups use the same data, original data is not directly used, but a copy of original data is used in many cases. In the DMA transfer, data is transferred on a data block basis. An example of a data block is shown in FIG. 10(a). A data block DB comprises data DATA1, DATA2, DATA3, DATA4, DATA5, DATA6, DATA7, and DATA8 in source memory locations having addresses AD1', AD2', AD3', AD4', AD5', AD6', AD7' and AD8' in the main memory 90. The data DATA1, DATA2, DATA3, DATA4, DATA5, DATA6, DATA7, and DATA8 are copied to destination memory locations having addresses AD1, AD2, AD3, AD4, AD5, AD6, AD7 and AD8 in the main memory 90.

As shown in FIG. 9(a), the DMA controller 82 contains a register 84 (hereinafter referred to as transfer counter 84) for storing the number of DMA-untransferred data and a register 86 (hereinafter referred to as transfer address 86) for storing an address of data under DMA transfer. In the case of the data block DB shown in FIG. 10(a), the transfer counter 84 stores the number of data "8" as an initial value. The DMA controller 82 reduces a value of the transfer counter 84 one by one every time the DMA transfer is completed.

The transfer address 86 stores an address of a source memory location and an address of a destination memory location. In the case of a data block shown in FIG. 10(a), an address AD1' of a source location and an address AD1 of a destination location are stored as an initial value. The DMA controller 82 updates addresses of source and destination locations every time the DMA transfer is completed.

FIG. 10(b) is a schematic diagram of a DMA data transfer of the data block DB. Each time data is transferred, the transfer counter 84 and the transfer addresses 86 are updated and DATA1 to DATA8 are transferred one after another. When the DMA transfer of DATA8 is completed, the value of the transfer counter becomes "0" (not shown) and thus the DMA transfer of the data block DB is completed. In FIG. 10(b), "AD'1→AD1" indicates that an address of a source location is AD'1 and an address of a destination location is AD1.

The CPU cannot access data under DMA transfer until all the DMA transfers are completed. FIG. 11 shows an example of a procedure by which the CPU 70 accesses data stored in a destination memory location. As shown in FIG. 9(b), when the CPU 70 requests access to data stored at the destination memory location (S162) after the start-up of the DMA transfer of the data block DB (S160), the DMA transfer is preferentially continued (S163). When the DMA transfer of the data block DB is completed (S164), the DMA controller 82 sends an interrupt signal INT to notify the CPU 70 of the completion of the DMA transfer (S166), as shown in FIG. 9(c). Thus, after the completion of the DMA transfer of the data

block DB, the CPU 70 can access the transferred data at the destination location (S168), as shown in FIG. 9(d).

For example, as shown in FIG. 12, when the CPU 70 requests access to DATA7 (S162) after the completion of transfer of DATA1 (S180), the CPU 70 accesses the DATA7 at the destination address (S168) after the completion of DMA transfer of DATA2 to DATA8 (S182, S184, S186, S188, S190, S192 and S194). Since the CPU 70 cannot access the DATA7 during the DMA transfer of the DATA2 to DATA8, latency time occurs.

An object of the present invention is to allow the CPU to access data under DMA transfer to thereby reduce latency time of the CPU during the DMA transfer.

SUMMARY OF THE INVENTION

A method of controlling a memory-to-memory DMA transfer according to the present invention comprises the steps of:

detecting a CPU request for access to data which is included in a data block being DMA-transferred; and allowing the CPU to access the requested data in preference to the DMA transfer under execution.

An apparatus for controlling a DMA transfer according to the present invention comprises: means for detecting a CPU request for access to data which is included in a data block being DMA-transferred; and means for allowing the CPU to access the requested data in preference to the DMA transfer under execution.

In the present invention, CPU can access the requested data before DMA transfer is completed, regardless of whether or not the requested data

has been DMA-transferred. Therefore, time consumed in DMA transfer becomes transparent to the CPU.

BRIEF DESCRIPTION OF THE DRAWING

FIGs. 1(a) to 1(d) show an embodiment of a DMA transfer control between memory locations in a main memory according to the present invention. FIG. 1(a) shows an example of a system configuration, FIG. 1(b) shows a flow of signal for controlling the DMA transfer, FIG. 1(c) shows a flow of an access request signal from a CPU, and FIG. 1(d) shows a flow of signal issued by a DMA controller for DMA-transferring data requested by the CPU first.

FIG. 2(a) shows the requested data which is sent from the main memory to the CPU by a DMA controller after the completion of the DMA transfer shown in FIG. 1(d), and FIG. 2(b) shows a flow of data which is requested by the CPU for access and is sent from a cache or the main memory to the CPU.

FIG. 3 shows an example of a procedure for controlling a DMA transfer of data block DB by the DMA controller shown in FIG. 1(a).

FIG. 4 shows an example of a procedure for DMA-transferring a data block DB by the DMA controller shown in FIG. 1(a).

FIGs. 5(a) to 5(d) show another embodiment of a DMA transfer control between memory locations in a main memory according to the present invention. FIG. 5(a) shows an example of a system configuration, FIG. 5(b) shows a flow of an access request signal from a CPU during the DMA transfer, and FIG. 5(c) shows that an address signal of the data

requested by the CPU transfers from a DMA hit detector to the DMA controller when a DMA hit is detected, and FIG. 5(d) shows a flow of signal sent by the DMA controller for requesting the CPU to retry access to the requested data.

FIG. 6(a) shows a flow of signal issued by the DMA controller to instruct the main memory to DMA-transfer the requested data by temporarily suspending access of the CPU to a cache or main memory when a retry of access to the requested data is requested as shown in FIG. 5(d), FIG. 6(b) shows a flow of DMA-transferred requested data sent to the CPU by the DMA controller after the completion of the DMA-transfer of the requested data, and FIG. 6(c) shows a flow of the requested data from the cache or main memory to the CPU when a DMA hit is not detected.

FIG. 7 shows an example of a procedure for controlling a DMA transfer by the DMA controller shown in FIG. 5(a).

FIGs. 8(a) to 8(c) show still another embodiment of a DMA transfer control between memory locations in a main memory according to the present invention. FIG. 8(a) shows an example of a system configuration, FIG. 8(b) shows a flow of the DMA-transferred data sent to a buffer by the DMA controller, and FIG. 8(c) shows a flow of data where the DMA hit detector transfers data stored in the buffer shown in FIG. 8(b) to the CPU.

FIGs. 9(a) to 9(d) show an embodiment of a conventional DMA transfer control. FIG. 9(a) shows an example of a system configuration, FIG. 9(b) shows a flow of signal for controlling a DMA transfer, and FIG. 9(c) shows a flow of signal from a DMA controller to a CPU to announce the

completion of the DMA transfer, and FIG. 9(d) shows access between the CPU and a cache or main memory.

FIG. 10(a) shows an example of a data block to be DMA-transferred, and FIG. 10(b) shows a value of a transfer counter and positional information of a transfer address in the DMA controller and a state of each data.

FIG. 11 shows an example of a procedure for controlling a DMA transfer by the DMA controller shown in FIG. 9(a).

FIG. 12 shows an example of a procedure for performing a DMA transfer of a data block by the DMA controller shown in FIG. 9(a).

DESCRIPTION OF THE PREFERRED EMBODIMENTS

Next, an embodiment of a method and apparatus for controlling a memory-to-memory DMA transfer according to the present invention will be described with reference to the accompanying drawings. In the following embodiments, a memory-to-memory DMA transfer is performed between memory areas in a main memory. In a data processing system shown in the drawings, it is assumed that a CPU 70, cache unit 72, system bus 80 and main memory 90 are similar to conventional ones shown in FIG. 9(a). In addition, it is assumed that a data block DB to be transferred is also similar to the conventional data block shown in FIG. 10(a).

An apparatus 10 (DMA controller) for controlling a memory-to-memory DMA transfer shown in FIGs. 1(a) to 1(d) according to the present invention comprises a DMA hit detector for detecting a CPU request for access to data at a destination memory location which may be

updated by data included in a data block under DMA transfer. The DMA controller 10 functions as means for allowing the CPU to access the requested data to be written at the destination location in preference to the DMA transfer under execution. The DMA controller 10 comprising the DMA hit detector 12 is connected between the CPU 70 and the system bus 80.

The DMA hit detector 12 detects whether or not the CPU requests access to DMA-untransferred data, that is, data which is to be written at a destination location accessed by the CPU but has not yet been DMA-transferred. The DMA controller 10 suspends the DMA transfer when a DMA hit is detected, and then forwards the requested data to the CPU 70.

Like a conventional DMA controller, the DMA controller 10 comprises a transfer counter 84 and a transfer address register 86. The DMA hit detector 12 can know destination addresses of DMA-untransferred data from the transfer counter 84 and the transfer address 86. The address of data to be accessed by the CPU 70 is transferred to the cache unit 72 and the DMA controller 10. The DMA hit detector 12 compares the address of data requested by the CPU 70 (hereinafter referred to as requested data) with the destination addresses of the DMA-untransferred data to detect if there is a match. A case where the address of the requested data coincides with one of the destination addresses is hereinafter referred to as "DMA hit". Letting ADi and Cj denote the destination address in the transfer address register 86 and the count in the transfer counter 84 at the time of occurrence of a CPU access

request, respectively, and AD_{cpu} denote the address of data requested by the CPU, a DMA hit occurs when $(AD_i) < (AD_{cup}) < (AD_i + C_j)$. It is to be noted that the block data addresses in the transfer address register 86 are sequential.

When a DMA hit is detected, it means that DMA transfer of the requested data is not completed. The DMA controller 10 suspends the DMA transfer of a data block under execution and performs a DMA transfer of the requested data. After the completion of the DMA transfer of the requested data, the DMA controller 10 forwards the requested data from the source location to the CPU 70. Then, the suspended DMA transfer is resumed with reference to the transfer counter 84 and the transfer address 86.

When a DMA hit is not detected, a DMA transfer of the requested data has already been completed. In that case, the CPU 70 can access the requested data even when a DMA transfer of a data block has not been completed yet. The CPU 70 accesses the cache 76 or the main memory 90 through the cache unit 72 to obtain the requested data. As a result, CPU can access the requested data under DMA without waiting for completion of DMA transfer regardless of whether DMA hit is detected or not.

Next, a DMA transfer control using such DMA controller 10 will be described.

FIG. 3 shows an example of a procedure for controlling a DMA transfer. It is assumed that after the DMA controller 10 starts a DMA transfer of a data blocks shown in FIG. 1(b) (S160), the CPU 70 requests access to data (S162). Then an address AD_n of the requested data $DATAn$

is sent from the CPU 70 to the DMA controller 10 and the cache unit 72, as shown in FIG. 1(c). In the DMA controller 10, the DMA hit detector 12 checks if the DMA hit is detected (S110).

When a DMA hit is detected, the DMA controller 10 suspends the DMA transfer of the data block under execution (S112), and performs a DMA transfer of the requested data (S114), as shown in FIG. 1(d). When a DMA hit is detected, the DMA controller 10 uses the system bus preferentially. After the completion of the DMA transfer of the requested data, the controller 10 forwards the requested data (DATA_n) from the source location of the main memory 90 to the CPU 70 (S116) as shown in FIG. 2(a), and then resumes the suspended DMA transfer (S124).

When a DMA hit is not detected, the DMA controller 10 suspends a DMA transfer of data under execution (S113). As shown in FIG. 2(b), the CPU 70 is allowed to access the cache memory 76 or the main memory 90 through the cache unit 72 (S118). After the completion of access to the data requested by the CPU 70, the controller 10 resumes the suspended DMA transfer of the data block (S124).

When all the data included in the data block is transferred (S164), the DMA controller 10 sends a signal to notify the CPU 70 of the completion of the DMA transfer (S166).

For example, as shown in FIG. 4, it is assumed that after a DMA transfer is started (S160) and a DMA transfer of DATA1 is completed (S180), the CPU 70 requests access to DATA7 (S162). Then the DMA controller 10 suspends the DMA transfer of the data block under execution (S112) and performs a DMA transfer of the DATA7 preferentially (S120).

In that case, the value of the transfer counter 84 is "7" and the transfer address 86 stores positional information "AD2'→AD2" at the time of suspension of the DMA transfer.

After the completion of the DMA transfer of the DATA7, the DMA controller 10 transfers the DATA7 to the CPU (S122) and then resumes the suspended DMA transfer of the data block DB (S124). When a series of DMA transfers of DATAs 2 to 8 are completed (S182, S184, S186, S188, S190, S192 and S194), the DMA controller 10 sends a signal to notify the CPU 70 of the completion of the DMA transfer of the data block (S166).

While the DATA2 to DATA 8 are being DMA-transferred, the CPU 70 can perform an operation using the DATA7. In an example shown in FIG. 4, the DMA transfer of the DATA7 is performed again in a step S192, however, it has no problem since the CPU 70 only refers to the DATA7. The system shown in FIG. 1 may be implemented with a single processor comprising a CPU, cache unit and DMA controller.

While one embodiment of the present invention has thus been described, the present invention can be materialized in other embodiments. For example, as shown in FIG. 5(a), the DMA hit detector 22 can be connected between the CPU 70 and the system bus 80, and the DMA controller 20 can be connected to the CPU 70 and the main memory 90 through the system bus 80. The DMA hit detector 22 detects a DMA hit with reference to the transfer counter 84 and the transfer address 86 in the DMA controller 20.

When a DMA hit is detected, the DMA controller 20 causes the CPU 70 to retry access to the requested data at the destination location. The

DMA controller 20 suspends the DMA transfer of a data block under execution and DMA-transfers the requested data until the CPU 70 re-accesses the requested data after it suspended access to the data. After the CPU 70 accesses the DMA-transferred requested data, the DMA controller 20 resumes the suspended DMA transfer of the data block.

FIG. 7 shows an example of a procedure for controlling a DMA transfer. As shown in FIG. 5(b), when the CPU 70 requests access to data at a destination location (S162) after the DMA controller 20 starts a DMA transfer of a data block (S160), an address AD_n of the requested data DATAn is transferred from the CPU 70 to the DMA hit detector 22 and the cache unit 72. The DMA hit detector 22 detects a DMA hit with reference to the transfer counter 84 and the transfer address 86 in the DMA controller 20.

When the DMA hit is detected, the detector 22 transfers the address AD_n of the requested data to the DMA controller 20 through the system bus 80 (S130), as shown in FIG. 5(c). In this case, the requested data is DMA-untransferred data. Receiving the address AD_n from the detector 22, the DMA controller 20 requests the CPU 70 to retry access to the requested data (S132), as shown in FIG. 5(d). The CPU 70 temporarily suspends access to the requested data, and after a lapse of predetermined time, the CPU 70 re-accesses the requested data. While the CPU 70 temporarily suspends the access to the data, the controller 20 suspends a DMA transfer of a data block under execution (S112) as shown in FIG. 6(a) and performs a DMA transfer of the requested data DATAn (S134).

At the time when the CPU 70 retries access to the data at the destination address, the data has been updated by the DMA-transferred data. As shown in FIG. 6(b), after the CPU 70 accesses the DATAn (S136), the DMA controller 20 resumes the suspended DMA transfer of the data block (S124). When all the data included in the data block is DMA-transferred (S164), the DMA controller 20 sends a signal to notify the CPU 70 of the completion of the DMA transfer (S166).

In FIG. 7, when a DMA hit is not detected, the CPU 70 accesses the cache 76 or the main memory 90 through the cache unit 72, as shown in FIG. 6(c). In this case, the DMA controller 20 suspends the DMA transfer of the data block under execution (S113). Then when the CPU 70 completes access to the requested data (S118), the suspended DMA transfer of the data block is resumed (S124).

As in the procedure shown in FIG. 4, even when the CPU 70 requests access to DATA7 (S162) after a DMA transfer is started (S160) and the DMA transfer of DATA1 is completed (S180), the DMA transfer of the DATA7 can be performed first. While the DATA2 to DATA 8 are being DMA-transferred, the CPU 70 can perform an operation using the DATA7. The system shown in FIG. 5(a) may be implemented with a single processor comprising a CPU, cache unit and DMA hit detector.

As shown in FIG. 8(a), a buffer 34 can be connected to the DMA hit detector 32. Data or a data block which is detected as a DMA hit is temporarily stored in the buffer 34, after the CPU accesses the requested data. Where the CPU 70 re-accesses the data stored in the buffer 34, the

DMA hit detector 32 transfers the data stored in the buffer 34 to the CPU 70. Thus the buffer 34 reduces the number of retry requests.

While the embodiments of the present invention have thus been described, it should be understood that the present invention be not limited to the above embodiments. For example, although a memory-to-memory DMA transfer between memory areas in the main memory has been described in this specification, the present invention can also be applied to a memory-to-memory transfer between any high-speed memories connected to a system bus. Also, in the system as described with reference to FIG. 1 to FIG. 4 in which the requested data is forwarded to the CPU from a source location, the step S114 in FIG. 3 may be omitted. Many changes, modifications, variations and other uses and applications can be made to the embodiments on the basis of knowledge of those skilled in the art without departing from the scope of the present invention.

WHAT IS CLAIMED IS:

(1) A method of controlling a DMA (Direct Access Memory) controller to perform a memory-to-memory DMA transfer in a data processing system including a main memory connected to a CPU (Central Processing Unit) through a system bus, comprising the steps of:

detecting a CPU request for access to data included in a data block being DMA-transferred; and

allowing the CPU to access the requested data in preference to the DMA transfer under execution.

(2) The method according to claim 1, wherein

said detecting step comprises detecting whether the requested data has already been DMA-transferred or not; and

said allowing step comprises:

a first step of allowing the CPU to access the requested data, if the requested data has already been DMA-transferred; and

a second step of suspending the DMA transfer to allow the CPU to access the requested data preferentially, if the requested data has not been DMA-transferred yet.

(3) The method according to claim 2, wherein said DMA controller comprises:

a first register for storing the number of DMA-untransferred data; and

a second register for storing the address of data under DMA transfer;
and said detecting step comprises:

comparing the address of the requested data with an address of
DMA-untransferred data obtained from the first and second registers to
detect if there is a match.

(4) The method according to claim 2 or 3, wherein said DMA
controller is connected between the CPU and the system bus and said
second step comprises:

causing the DMA controller to forward the requested data to the
CPU; and

resuming the suspended DMA transfer.

(5) The method according to claim 2 or 3, wherein said DMA
controller is connected to the CPU and the main memory through the
system bus and said second step comprises:

requesting the CPU to retry access to the DMA-untransferred data;

suspending the DMA transfer under execution to DMA-transfer the
requested data until the CPU retries access to the data; and

resuming the suspended DMA transfer after the CPU accesses the
DMA-transferred data.

(6) An apparatus for controlling a memory-to-memory DMA
transfer in a data processing system including a main memory connected to
a CPU through a system bus, comprising:

means for detecting a CPU request for access to data included in a data block being DMA-transferred; and

means for allowing the CPU to access the requested data in preference to the DMA transfer under execution.

(7) The apparatus according to claim 6, wherein said detecting means includes means for detecting that the requested data is DMA-untransferred and said allowing means includes:

means for DMA-transferring the requested data preferentially; and

means for allowing the CPU to access the DMA-transferred data.

(8) The apparatus according to claim 7 further comprising:

a first register for storing the number of DMA-untransferred data;

and

a second register for storing an address of data under DMA transfer, wherein said detecting means includes means for comparing the address of the requested data with an address of DMA-untransferred data obtained from the first and second registers to detect if there is a match.

(9) The apparatus according to any one of claims 6 to 8, wherein said detecting means and allowing means are connected between the CPU and the system bus.

(10) The apparatus according to any one of claims 6 to 8, wherein said detecting means is connected between the CPU and the system bus

and said allowing means is connected to the CPU and the main memory through the system bus.

(11) The apparatus according to claim 10, wherein said detecting means comprises a buffer memory for temporarily storing the data detected as DMA-untransferred data.

(12) The apparatus according to any one of claims 6 to 11, wherein a cache memory is connected between the CPU and the system bus.