Specification

1. Title of the Invention:

Disk drive device and method for controlling the same

Detailed Description of the Invention:

[0001]

[Field of the invention]

The present invention relates to a disk drive device for use with a hard disk drive (HDD) or the like, and a method for controlling the same, and specifically, to a disk drive device including a cache memory and a method for controlling the same.

[0002]

[Background art]

In the hard disk drive (HDD) which is used as an auxiliary storage for an information processing unit, there are generally provided a cache memory for temporarily retaining the data supplied from the information processing unit (hereinafter referred to as host) or temporarily retaining the data read out from a magnetic disk until this data is transferred to the host, and a controller for performing the cache control.

[0003]

When supplied with data from the host, the controller retains the supplied write data in the cache memory, and then releases the host from the write processing. Concurrently, the controller controls a writing system so that the write data retained in the cache memory are sequentially written into the magnetic disk (hereinafter referred to as medium). Further, the controller instructs a reading system to read from the medium the data for which a request to read is expected from the host, and retains the read data in the cache memory. And, if a request to read is issued from the host and the data to be read is retained in the cache memory, it is supplied to the host. Such cache control increases the apparent access speed of the HDD.

[0004]

In the recent low end HDD, as the means for increasing the performance by decreasing the command overhead, an approach tends to be taken for reducing the intervention of the microprocessor of the HDD (hereinafter referred to as local MPU) in the processings around the host interface, and causing a host

interface controller (HIC) by hardware to perform the processings as much as possible.

[0005]

As one of them, it is possible to use a system in which data transfer can be performed between the host and the cache memory under the control of the HIC without the intervention of the local MPU, while the data transfer between the medium and the cache memory requires the intervention of the local MPU. In such system, if all the data required by a read command from the host are already retained in the cache memory (all-hit), and all the data transfer can be performed without the intervention of the local MPU, the following techniques are conventionally employed.

[0006]

(1) While the intervention of the local MPU is not required, all the processings are performed by hardware.

[0007]

(2) No data transfer is performed by hardware, and there is always the intervention of the local MPU.

[0008]

[Problems to be solved by the invention]

However, in the HDD taking such conventional approach in which the intervention of the local MPU is reduced as much as possible and the HIC is caused to perform the processings as much as possible, there is a limit to the increase in performance for the following reason.

[0009]

That is, in the technique of (1), there is an advantage that the corresponding command itself can be processed at a very high speed since the processing is performed by hardware, but the performance decreases in the following case.

[0010]

It is now supposed that the above described all-hit continues at very short command intervals. This case corresponds to a sequential read, and frequently occurs in general. In this case, data can be transferred at high speed as long as the required data exists in the cache memory, but, since no intervention of the local MPU is allowed, an additional read into the cache memory from the medium cannot be performed and no additional read is performed. Accordingly, the data in the cache memory is exhausted after a while, and the required data must be read from the medium. This requires an order-of-magnitude long time as compared with the command overhead time needed for data transfer, such as the turning of the medium, and thus the data transfer speed largely decreases.

[0011]

On the other hand, if the above technique of (2) is used, the above disadvantage can be avoided since every time the intervention of local MPU is possible, but the transfer function by hardware cannot be exploited at all.

[0012]

It is the object of the present invention to provide a disk drive device and a method for controlling the same in which an additional read from a medium can be performed as needed while

suppressing the command overhead to the minimum, thereby to enable the total performance to be increased.

[0013]

[Summary of the invention]

The disk drive device of the present invention comprises a disk storage medium, a disk controller for controlling a read/write operation to the disk storage medium, a cache memory for retaining data additionally read from the disk storage medium, a host interface controller for transferring data between a host and the cache memory by hardware, and a microprocessor for controlling the operation of the whole device, including the data transfer between the disk storage medium and the cache memory, and it is characterized in that even if a read command is issued, and the required data hit the data retained in the cache memory so that all the data transfer can be performed without the intervention of the microprocessor, the host interface controller causes an additional read into the cache memory through the microprocessor at least once before completing the read command.

[0014]

Further, the method for controlling a disk drive device is characterized in that, if a read command is issued, and the whole required data are retained in the cache memory so that all the data transfer can be performed without the intervention of the microprocessor, the host interface controller is caused to perform the data transfer of the blocks except the last predetermined number of blocks and to notify the microprocessor of the start of the data transfer, instructions to additionally read data into the cache memory are given from the microprocessor to the disk controller in response to the notification of the start of the data transfer, and the host interface controller is

caused to transfer the last predetermined number of blocks after the additional read control is terminated by the disk controller.

[0015]

[Preferred embodiment]

The disk drive device related to the present invention can be applied to a low end HDD including an IDE (Integrated Device Electronics) interface.

[0016]

Figure 1 is a block diagram showing the construction of the disk drive device related to the embodiment of the present invention. In the figure, thick solid arrows represent data flows, and thin solid arrows represent control signal flows.

[0017]

In Figure 1, a magnetic disk unit (HDD) 10 comprises a magnetic head 11 for performing a data read/write to a magnetic disk, a channel module 12 including a circuit for amplifying a detected signal, a waveform shaping circuit, an analog-to-digital converter (ADC), and a digital-to-analog converter (DAC) in modules, a hard disk controller (HDC) 13 for controlling the operation of data read/write to the magnetic disk, a cache memory 14 for retaining supplied write data, a host interface controller (HIC) 15 for performing the processings around the host interface by hardware, and a local MPU for controlling the operation of the whole HDD, including the controls on the HDC 13 and HIC 15.

[0018]

Further, the HDC 13, cache memory 14, and HIC 15 are connected to each other by a data bus 17, the HDC 13 and HIC 15 are connected to the local MPU by a control bus 18, and the HIC 15 is connected to a host outside the HDD by an IDE bus 19.

[0019]

The cache memory 14 has a storage capacity to the extent that it can retain more than several hundreds of recording units (sectors) on the magnetic disk, for instance, 128K bytes (256 blocks). In this example, 1 block=1 sector=512 bytes. However, in IDE, the host can specify a plurality of blocks on the cache memory as one block, that is, it can specify 1, 2, 4, 8, 16, 32, ... sectors as one block.

[0020]

The HIC 15 has an internal memory capable of storing all the information necessary for data transfer, and transfers data between the host and the cache memory 14 by hardware without the invention of the local MPU 16.

[0021]

In the internal memory of the HIC 15, the correspondence between addresses on the cache memory 14 and the magnetic disk, and the number of continuously stored blocks are preserved in the form of a cache table.

[0022]

When a command is issued from the host, the HIC 15 compares the address of the issued command with the cache table, and notifies

the local MPU 16 of the hit status of the one having the highest hit rate (this is not the one which simply hits most, but the one which satisfies such a condition that the beginning of a block is hitting). The HIC 15 can determine whether or not the required data exist in the cache memory 14 when a read command is issued from the host, and also can determine how many blocks to transfer without the intervention of the local MPU 16.

[0023]

Specifically, the HIC 15 has a function of calculating how many blocks to transfer for leaving one block, and transferring the blocks based on the calculation if it determines that all the required data exist in the cache memory 14 when a read command is issued. Further, when the read command is issued, the HIC 15 sends to the local MPU 16 the contents of the command, the fact that the all the required data are contained in the cache memory, and a request for being notified of the completion of the preparation for data transfer.

[0024]

The local MPU 16 is a microprocessor for executing a control program. It has a memory or the like for storing the control program and data, and executes a processing according to the control program to control the operation of the whole HDD and control the operations of the HDC 13 and the HIC 15 based on the command and control data supplied from the HIC 15 and the host. That is, when the local MPU 16 is informed of the command information from the host and the hit information for the command by the HIC 15, it controls the HIC 15 based on such information.

[0025]

Further, the local MPU performs a processing such as a calculation of the additional read from the medium as needed, and provides instructions on the additional read to the HDC 13. The HDC 13 performs the additional read from the medium according to the instructions, and write the additionally read data into the cache memory 14. To newly write data into the cache memory 14 from the medium as described above, it is needed to update the above described cache table. The local MPU 16 calculates whether or not to perform an additional read into the cache by using an algorithm taking into consideration the past access pattern of the host and the amount of data remaining in the cache memory 14.

[0026]

As described above, the HDD 10 has an IDE interface, and includes the cache memory 14 between the data bus 17 and the medium. The HIC 15 can transfer data between the HIC 15 and the cache memory 14 without the intervention of the local MPU 16, but the data transfer between the medium and the cache memory 14 requires the intervention of the local MPU 16.

[0027]

Now, the operation of the HDD 10 constructed as described above is explained.

[0028]

First, the basic idea of the present invention is described.

[0029]

The IDE interface is characterized in that, for a write command, the HDD can control the timing for terminating the command after the termination of the all the data transfer. That is, it is possible to cause the HIC to perform the data transfer itself to the cache memory, and cause the command to be terminated at a time convenient for the local MPU after a space of, for instance, one or more blocks is made in the cache memory by a write into the disk.

[0030]

On the other hand, for a read command, the data transfer termination itself means the termination of the command. In other words, all data transfers are performed by the HIC. As a result, there may be left no room for the local MPU to perform a processing such as an additional read for the cache. This corresponds to the case in which continuous read commands successively hit the cache, and it may frequently occur in a sequential reading in general.

[0031]

Accordingly, the present invention is to improve the above described disadvantage while suppressing the command overhead to the minimum by giving room for the intervention of the local MPU at least once before completing a read command, while performing the data transfer to the maximum without the intervention of the local MPU. Specifically, even if a read command is issued and the required data hit the cache so that all the data transfer can be performed without the intervention of the local MPU, the HIC stops the transfer leaving the last one block and waits for the intervention of the local MPU. The local MPU permits the

transfer of the last one block when the preparation for the command termination is made.

[0032]

That is, paying attention to the point that the command does not terminate unless all the data transfer terminates, data are transferred by the HIC, leaving only the last one block (the command has not terminated even if only one block is left).

[0033]

Upon receipt of a command, the transfer is started, and simultaneously the local MPU is informed of this. In the local MPU, a predetermined time is taken for the calculations of the additional read and the like (calculation related to the additional read and calculation related to the cache table update), and the HIC is performing the data transfer during that time. If the time taken for the calculations of the additional read and the like in the local MPU is longer than the time taken for the data transfer, the transfer is paused by the HIC, leaving the last one block. On the other hand, if the time taken for the calculations of the additional read and the like is shorter than the time taken for the data transfer, the transfer is not paused.

[0034]

Now, the operation of the HDD 10 is described in detail based on the above basic idea.

[0035]

As shown by the main data flows represented by thick solid arrows in Figure 1, data read in from the medium enter the cache memory 14, and are transferred from the cache memory 14 to the host

through the HIC 15 and the IDE bus 19 as long as there is no intervention of the local MPU 16.

[0036]

If all-hit of the cache memory 14 takes place in a read command (that is, a cache-hit case in which all the required data exist in the cache memory 14), the HIC 15 immediately starts the data transfer without the intervention of the local MPU 16. In addition, the HIC 15 notifies the local MPU 16 of the issuance of the command and the all-hit, and continues the transfer until the last block is left.

[0037]

On the other hand, upon receipt of the notification, the local MPU 16 performs the processing for starting an additional read from the medium as needed (for instance, the processing in which, when it is now already issuing a command for reading data to the HDC 13, it recalculates how long the read to the HDC 13 should be extended by an additionally issued command, and reflecting the result on the cache table, issues instructions to the HDC 13), and when the processing for starting the additional read from the medium by the local MPU 16 is terminated, it notifies the HIC 15 of the termination of the control of additional read. This notification means that the preparation for the command termination has been made, and allows the HIC 15 to transfer the last block.

[0038]

As to the control of additional read, it is only needed that the control of additional read has been issued, and an actual additional read need not always be performed.

[0039]

Depending on the data transfer status of the HIC 15 when the HIC 15 is allowed to transfer the last block by the local MPU 16, there are the following two cases.

[0040]

Case 1
At the above point of time, the HIC 15 has already terminated the data transfer except the last block, and the HIC 15 has stopped the transfer, leaving the last block. In this case, the HIC 15 restarts the transfer to transfer the last block, terminating the command.

[0041]

Case 2
Data of one or more blocks is remaining in the HIC 15 when the HIC 15 is allowed to transfer the last block, and the HIC 15 is still transferring data. In this case, the HIC 15 continuously executes the transfer to the end without leaving the last block, completing the command.

[0042]

The above operation is described in detail with reference to the flowcharts shown in Figures 2 and 3.

[0043]

Figures 2 and 3 are timing flows showing the operation of the HDD. Figure 2 shows the case in which the transfer of the last block is allowed by the local MPU 16 after the transfer is stopped leaving the last block (corresponding to the above case

1), and Figure 3 shows the case in which the transfer of the last block is allowed by the local MPU 16 before the transfer is stopped leaving the last block (corresponding to the above case 2).

[0044]

In Figure 2, first, if a read command is issued from the host (step ST1), the HIC 15 checks for a cache hit in step ST2 to determine whether all-hit of the cache memory 14 has occurred in the read command, that is, whether the all the required data exist in the cache memory 14.

[0045]

For all-hit, the process goes to step ST3 where the HIC 15 starts the data transfer is step ST3.

[0046]

The HIC 15 immediately starts the data transfer without the intervention of the local MPU 16, as shown by transfer data d1 in Figures 1 and 2. It continues the data transfer, and terminates the transfer in step ST4 to leave the last block, waiting for the allowance of the transfer of the last block.

[0047]

On the other hand, simultaneously with the start of the data transfer, in step ST5, the HIC 15 notifies the local MPU 16 of the command issuance and the all-hit. This notification is shown by control c1 in Figures 1 and 2.

[0048]

Upon receipt of the-notification, the local MPU 16 performs the processing such as the calculation of an additional read from the medium in step ST6, as needed, and instructs the HDC 13 to perform an additional read in step ST7. This control is shown by control c2 in Figures 1 and 2. The HDC 13 performs an additional read from the medium according to the control c2, and retains the additionally read data in the cache memory 14, as shown by additionally read data d2 in Figures 1 and 2.

[0049]

In step ST8, the local MPU 16 determines the termination of the control of the additional read, and if the control of the additional read is terminated, it determines that the preparation of the command termination is made, and issues a command permitting the transfer of the last block to the HIC 15. This control is shown by control c3 in Figures 1 and 2.

[0050]

If the transfer is permitted, the HIC 15 resumes the transfer in step ST9, and transfers the last block as shown by transfer data d3 in Figures 1 and 2, terminating the command (step ST10).

[0051]

On the other hand, the timing flow of Figure 3 shows the case in which the transfer of the last block is allowed by the local MPU 16 before the transfer is stopped leaving the last block.

[0052]

In Figure 3, first, when a read command is issued from the host (step ST1), the HIC 15 performs a cache hit check in step ST2 to determine whether all-hit of the cache memory 14 has occurred in the read command.

[0053]

For all-hit, the process goes to step ST3 where the HIC 15 starts the data transfer.

[0054]

In this case, the HIC 15 immediately starts and continues the data transfer without the intervention of the local MPU 16, as shown by transfer data d1 in Figures 1 and 3, and executes the transfer to the end without stopping the transfer to leave the last block in step ST11, terminating the command (Step ST10).

[0055]

Further, simultaneously with the starting of the data transfer in the above step ST3, the HIC 15 notifies the local MPU 16 of the command issuance and the all-hit in step ST5. This control is shown by control c1 in Figures 1 and 2.

[0056]

Upon receipt of the notification, the local MPU 16 performs the processing such as the calculation of an additional read from the medium in step ST6, as needed, and instructs the HDC 13 to perform an additional read in step ST7. This control is shown by control c2 in Figs 1 and 2. The HDC 13 performs an additional read from the medium according to the control c2, and retains the

additionally read data in the cache memory 14, as shown by additionally read data d2 in Figures 1 and 2.

[0057]

In step ST8, the local MPU 16 determines the termination of the control of the additional read, and if the control of the additional read is terminated, it determines that the preparation of the command termination is already made, and issues a command permitting the transfer of the last block to the HIC 15. This control is shown by control c3 in Figures 1 and 2. Since the additional read processing has been completed if the HIC 15 receives the allowance of transferring the last block before stopping the transfer to leave the last block, as described above, it executes the transfer to the end without stopping the transfer to leave the last block in step ST11, terminating the command (step ST10).

[0058]

As described above, when all-hit of the cache memory 14 occurs in the read command, the data transfer is started by the HIC 15 so as to leave the last one block and the local MPU 16 is informed of this, and the local MPU 16 performs the control of an additional read while the HIC 15 is transferring data, and the transfer is paused if the notification of the termination of the control of the additional read is not received before the last one block (corresponding to the timing flow of Figure 2), otherwise the last one block is also transferred without pausing the transfer (corresponding to the timing flow of Figure 3).

[0059]

Accordingly, by giving room for the local MPU to intervene at least once before completing a read command while performing the

maximum data transfer in the read command without the intervention of the local MPU, the command overhead can be suppressed to the minimum, and the additional read from the medium can be performed as needed.

[0060]

As described above, the HDD 10 related to this embodiment comprises a magnetic head 11, a channel module 12, a HDC 13, a cache memory 14, a HIC 15 for performing the processings around the host interface by hardware, and a local MPU 16 for performing the control of the whole HDD, including the control on the HDC 13 and the HIC 15, and even if a read command is issued, and the required data hit the data retained in the cache memory so that all the data transfer can be performed without the intervention of the local MPU 16, the HIC 15 transfers data except the last one block and waits for the intervention of the local MPU 16, permits the transfer of the last one block when the preparation of the command termination is made. And, if a plurality of blocks of data to be transferred is remaining, when the local MPU 16 allows the HIC 15 to transfer the last one block, the HIC 15 executes all the data transfer without stopping the data transfer before the last one block and completes the command, and thus, while suppressing the command overhead to the minimum, an additional read from the medium can be performed as needed, whereby the total performance can be increased.

[0061]

A calculation result obtained when the present invention is applied to the actual HDD is described.

[0062]

It is a simulation result for the case in which a sequential read of a total of 4 MB with 8 blocks/command is issued. The simulation was made with the following hardware construction and condition.

[0063]

Cache memory size: 128 KB (256 blocks)
Average read time from medium: 5.5 msec
Overhead of microcode for all-hit: 100 μsec
Overhead before data transfer for hardware: 0 μsec
Command interval: 0 μsec
Data transfer speed: PIO mode 4 of IDE (32 μsec/1 block)
Assuming the above condition, the transfer time per eight blocks is 32 μsec × 8 blocks = 256 μsec.

[0064]

If a sequential read is issued from a cache memory full condition, assuming that data for 32 times can be retained under the cache memory full condition, first, in a background art example,

(1) 438 msec is taken if all the processing is performed by hardware, and
(2) 365 msec is taken if all the processing is intervened by the local MPU.

On the other hand, 262 msec is taken for the HDD employing this embodiment, and accordingly the speed can be largely increased as compared with the conventional method. That is, for the above (1), the average read time for additional reading from the medium (5.5 msec × 32) is eliminated, and for the above (2), the execution time of the local MPU for each all-hit (100 μsec × 32)

is eliminated, whereby the total performance can be increased by 30 to 40%.

[0065]

Although, in this embodiment, an example has been described in which the present invention is applied to the HDD, the present invention is not limited to this, but it can be applied to any apparatus as long as it is a disk drive device including a cache memory. For instance, it may be used with an external storage device other than the HDD, such as a magneto-optic disk, and an effect similar to the above described embodiment can be obtained.

[0066]

Further, in this embodiment, the HIC transfers data so as to leave the last one block, but any type of intervention may be used, provided that the transfer is intervened by the control by the microprocessor at least once before the read command is completed. Furthermore, data is transferred so as to leave the last one block, but, not being limited to one block, the data transfer may be made so as to leave a predetermined number of blocks.

[0067]

Moreover, it is needles to say that the types and number of the HDC, cache memory, HIC, etc. which are forming the above disk drive device are not limited to the above described embodiment.

[0068]

[Advantages of the invention]

In the disk drive device and the method for controlling the same which are related to the present invention, even if a read command is issued, and the required data hit the data retained in the cache memory so that all the data transfer can be performed without the intervention of the microprocessor, the control by the microprocessor is caused to intervene at least once before the read command is completed, and thus an additional read from the medium can be performed as needed while suppressing the command overhead to the minimum, whereby the total performance can be increased.

4. Brief Description of the Drawings:

Figure 1 is a block diagram showing the construction of the disk drive device related to the embodiment to which the present invention is applied;

Figure 2 is a timing flowchart for explaining the operation of the disk drive device;

Figure 3 is a timing flowchart for explaining the operation of the disk drive device.

[Description of symbols]

10 ... Disk drive device
11 ... Magnetic head
12 ... Channel module
13 ... Hard disk controller (HDC)
14 ... Cache memory

15 ... Host interface controller (HIC)

16 ... Local MPU

Claims:

(1) A disk drive device comprising:

a disk storage medium;

a disk controller for controlling a data read/write operation to said disk storage medium;

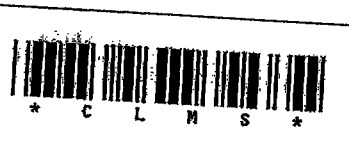
a cache memory for retaining data additionally read from said disk storage medium;

a host interface controller for transferring data between a host and said cache memory by hardware; and

a microprocessor for controlling the operation of the whole device, including the data transfer control between said disk storage medium and said cache memory,

wherein said host interface controller causes an additional read into said cache memory through the control by said microprocessor at least once before completing a read command, even if the read command is issued and the whole required data are retained in said cache memory.

(2) A disk drive device as set forth in Claim 1,



wherein said host interface controller is an interface system in which the data transfer termination is the read command termination, and even if said read command is issued and the whole required data are retained in said cache memory so that all the data transfer can be performed without the intervention of said microprocessor, it starts the data transfer for the blocks other than the last predetermined number of blocks and notifies said microprocessor of the execution of said data transfer, and

said microprocessor allows said host interface controller to transfer said last predetermined number of blocks after terminating said additional read control.

(3) A disk drive device as set forth in Claim 2,

wherein if transfer data more than said last predetermined number of blocks are remaining when said microprocessor allows said host interface controller to transfer said last predetermined number of blocks, said host interface controller executes all the data transfer without stopping the data transfer before starting the transfer of said last predetermined number of blocks, and terminates the command.

(4) A disk drive device as set forth in Claim 1, 2 or 3,

wherein said host interface controller has a unit for storing the correspondence between addresses on said cache memory and said disk, and the number of continuously stored blocks.

(5) A disk drive device as set forth in Claim 1, 2 or 3,

wherein said host interface controller has a logic for calculating the number of blocks to be transferred except

* C L M S *

the last predetermined number of blocks if all the data required when a read command is issued are retained in said cache memory.

(6) A disk drive device as set forth in Claim 1, 2 or 3,

wherein when a read command is issued, said host interface controller sends the contents of said command, information showing that all the data required by said host are retained in said cache memory, to said microprocessor and a request for the notification of the preparation for the command termination by said microprocessor.

(7) A disk drive device as set forth in Claim 1, 2 or 3,

wherein said microprocessor instructs said disk controller to additionally read data into said cache memory, and determining that said preparation for the command termination has been made when the additional read by said disk controller is terminated, allows said host interface controller to transfer the last predetermined number of blocks.

(8) A method for controlling a disk drive device comprising:

a disk controller for controlling a data read/write operation to a disk storage medium;

a cache memory for retaining data additionally read from said disk storage medium;

a host interface controller for transferring data between a host and said cache memory by hardware; and

* C L M S *

a microprocessor for controlling the operation of the whole device, including the data transfer control between said disk storage medium and said cache memory,

wherein if a read command is issued, and the whole required data are retained in said cache memory so that all the data transfer can be performed without the intervention of said microprocessor, said host interface controller is caused to perform the data transfer of the blocks except the last predetermined number of blocks,

notify said microprocessor of the start of said data transfer, so that instructions to additionally read data into said cache memory are given from said microprocessor to said disk controller in response to the notification of the start of said data transfer and

transfer the last predetermined number of blocks after the additional read control is terminated by said disk controller.

* C L M S *