US 20060112252A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2006/0112252 A1

Dixon (43) Pub. Date: May 25, 2006

(54) DEVICE-MANAGED HOST BUFFER

(75) Inventor: **Robert W. Dixon**, Longmont, CO (US)

Correspondence Address:
SHUMAKER & SIEFFERT, P. A.
8425 SEASONS PARKWAY
SUITE 105
ST. PAUL, MN 55125 (US)

(73) Assignee: **Seagate Technology LLC**

(21) Appl. No.: **11/265,617**

(22) Filed: **Nov. 2, 2005**

Related U.S. Application Data

(63) Continuation of application No. 10/443,947, filed on May 22, 2003, now Pat. No. 6,981,123.

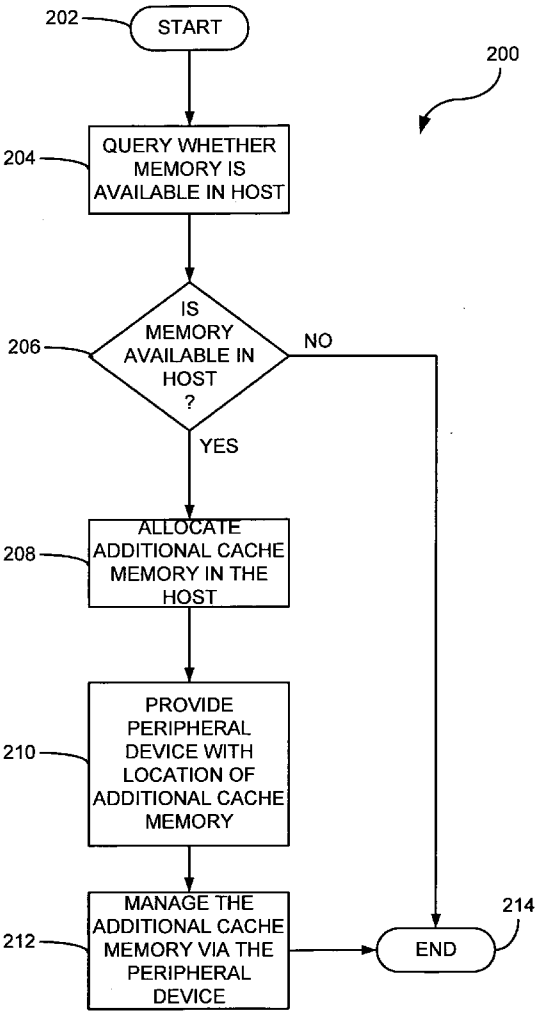
Publication Classification

(51) **Int. Cl.**
G06F 12/00 (2006.01)

(52) **U.S. Cl.** **711/170**; 711/114; 710/22

(57) **ABSTRACT**

A method and apparatus is provided to virtually increase the size of the memory cache of a peripheral device without additional cost. A portion of the memory space of a host computer is used as additional cache memory for the peripheral device. The peripheral device and the host computer may be interfaced with an interface that has a first-party direct memory access (FPDMA) mechanism, for example, IEEE **1394** or Serial ATA. FPDMA allows the peripheral device to access the memory space of the host computer under the control of the peripheral device. The host computer provides the peripheral device with the location of the additional cache memory. The peripheral device can transfer data to and from the additional cache memory via FPDMA. The peripheral device effectively manages the additional cache memory as part of the peripheral device's own cache.



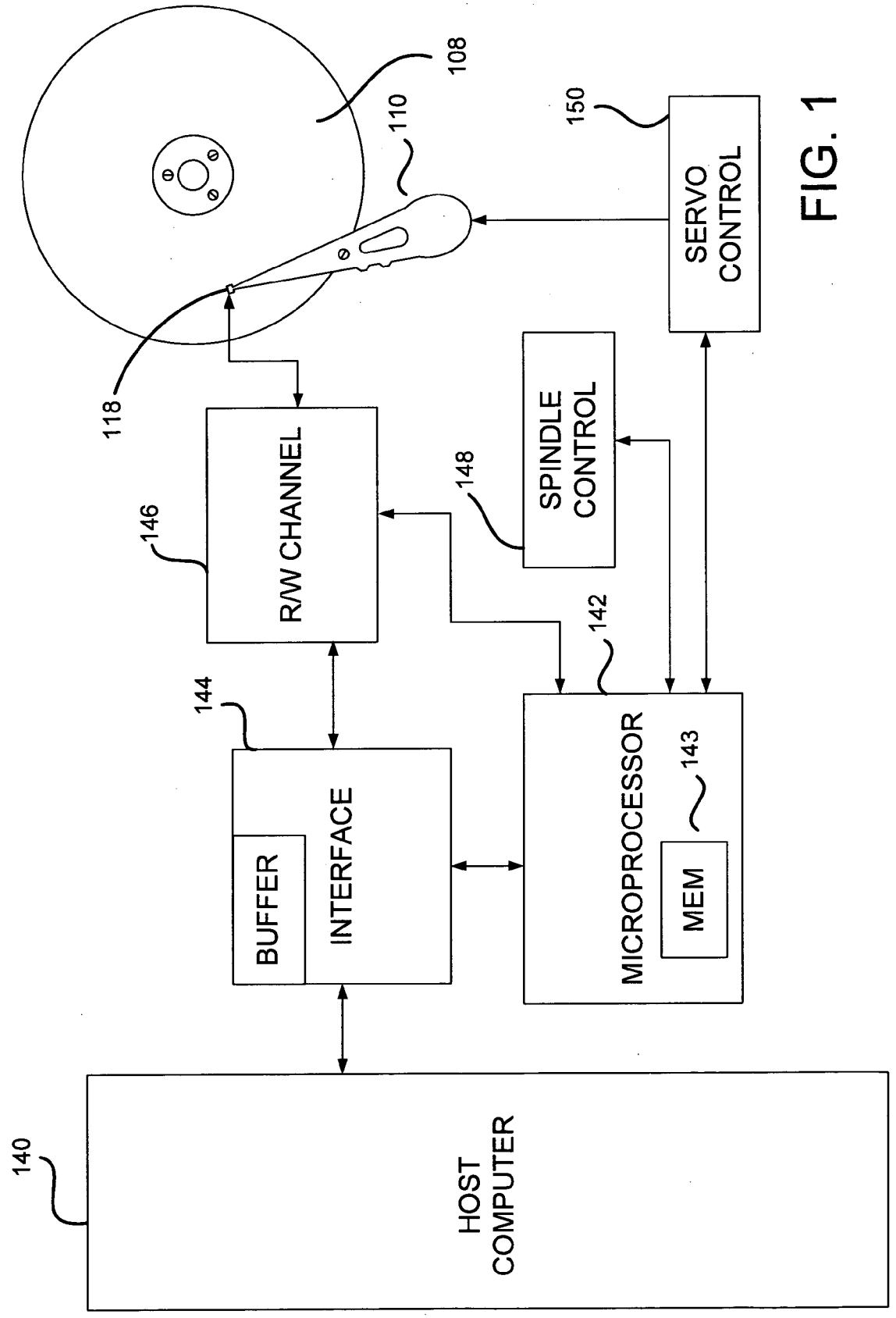

108
110
150
SERVO CONTROL
118
R/W CHANNEL
146
SPINDLE CONTROL
148
142
144
BUFFER
INTERFACE
MICROPROCESSOR
143
MEM
140
HOST COMPUTER

FIG. 1

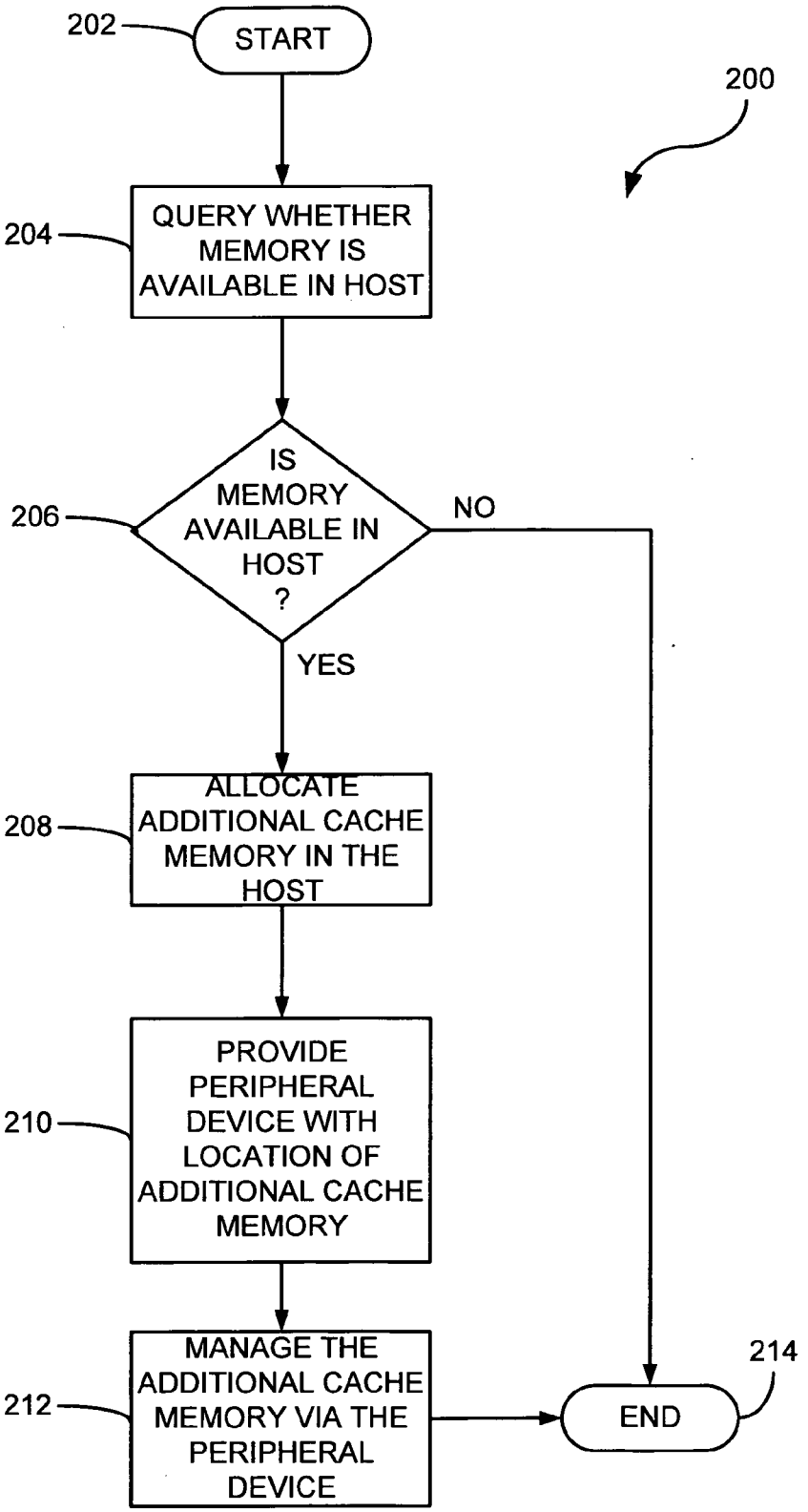

202
START
200
204
QUERY WHETHER MEMORY IS AVAILABLE IN HOST
206
IS MEMORY AVAILABLE IN HOST ?
NO
YES
208
ALLOCATE ADDITIONAL CACHE MEMORY IN THE HOST
210
PROVIDE PERIPHERAL DEVICE WITH LOCATION OF ADDITIONAL CACHE MEMORY
212
MANAGE THE ADDITIONAL CACHE MEMORY VIA THE PERIPHERAL DEVICE
214
END

FIG.2

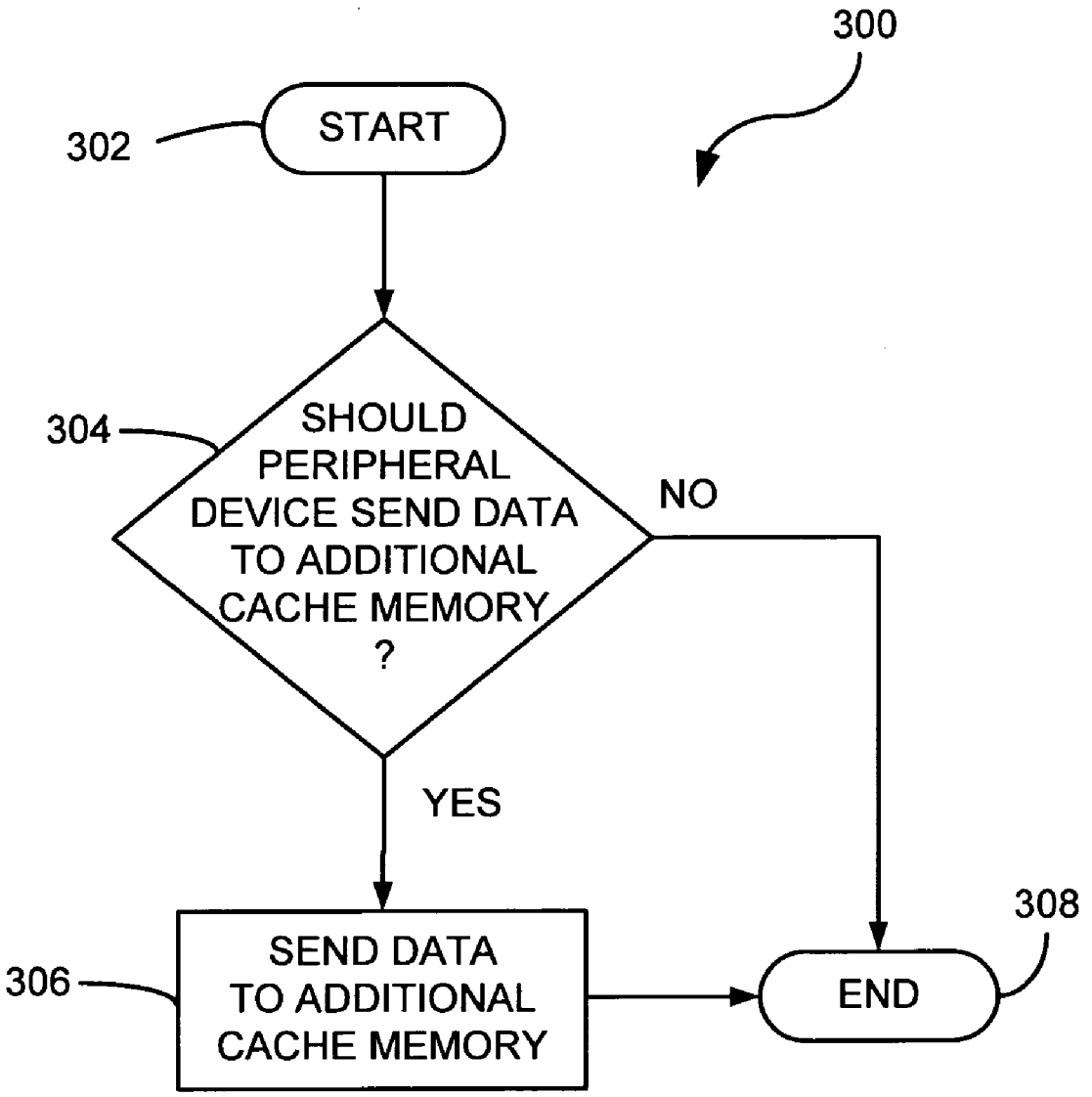

300
302
START
304
SHOULD PERIPHERAL DEVICE SEND DATA TO ADDITIONAL CACHE MEMORY ?
NO
YES
306
SEND DATA TO ADDITIONAL CACHE MEMORY
308
END

FIG.3

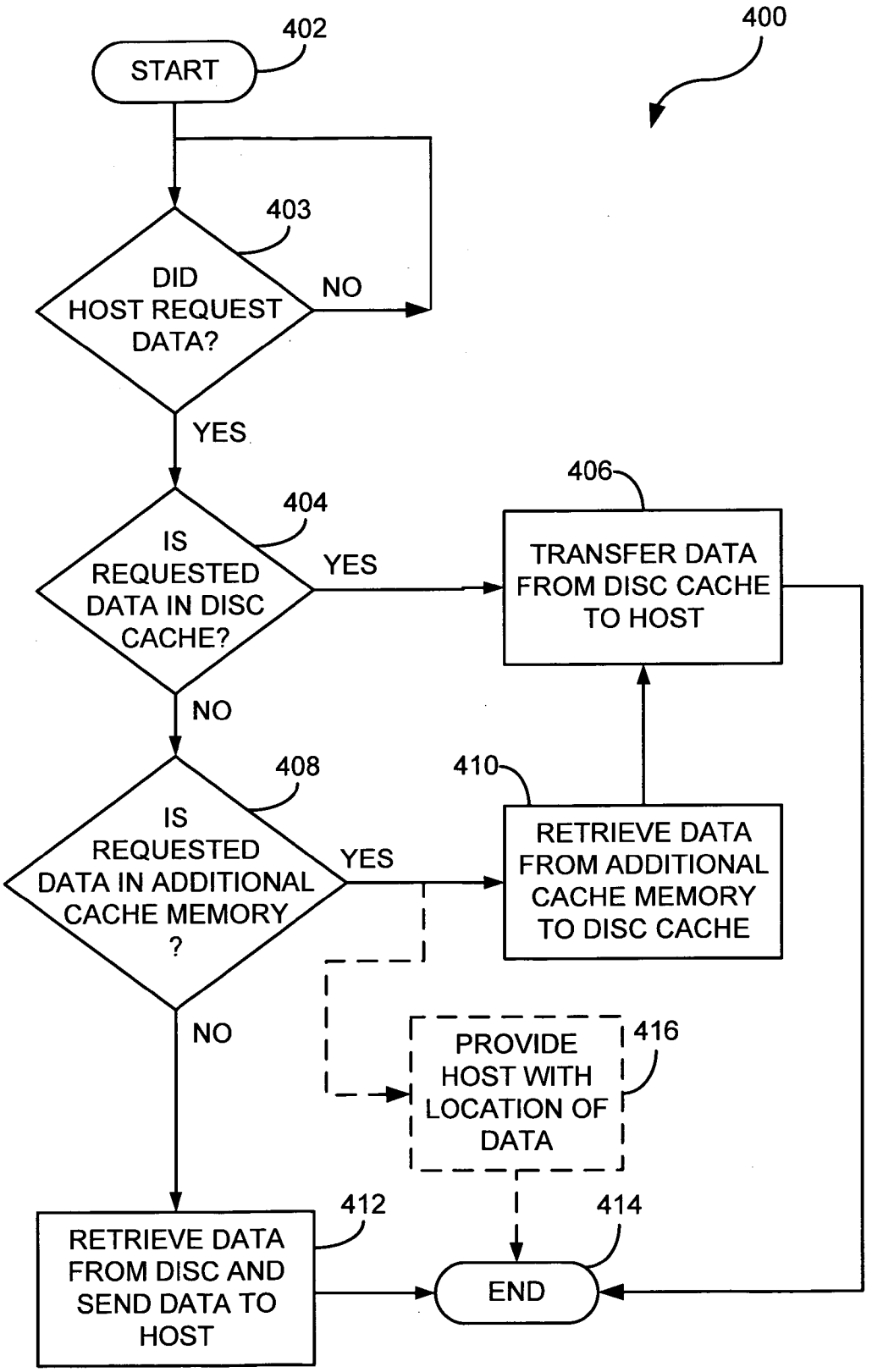

400
402
START
403
DID HOST REQUEST DATA?
NO
YES
404
IS REQUESTED DATA IN DISC CACHE?
YES
406
TRANSFER DATA FROM DISC CACHE TO HOST
NO
408
IS REQUESTED DATA IN ADDITIONAL CACHE MEMORY ?
YES
410
RETRIEVE DATA FROM ADDITIONAL CACHE MEMORY TO DISC CACHE
NO
PROVIDE HOST WITH LOCATION OF DATA
416
412
RETRIEVE DATA FROM DISC AND SEND DATA TO HOST
414
END

FIG.4

DEVICE-MANAGED HOST BUFFER

FIELD OF THE INVENTION

[0001] This application relates generally to memory caching in a peripheral device and more particularly to a method and apparatus for using a portion of a memory space of a host computer as additional cache memory for a peripheral device.

BACKGROUND OF THE INVENTION

[0002] The performance of a peripheral device that has a memory cache can be substantially increased when the size of the memory cache is increased. However, increasing the size of the memory cache of a peripheral device can be prohibitively expensive. Consequently, peripheral devices have limited cache capability built into them. This limited cache memory potentially provides a substantial burden on the throughput that the host and the peripheral device are able to handle, and thus achievable performance of the peripheral device in operation is a compromise of performance and cost.

[0003] Accordingly there is a need for effectively increasing the buffer memory of a peripheral device without additional cost. The present invention provides a solution to this and other problems, and offers other advantages over the prior art.

SUMMARY OF THE INVENTION

[0004] Against this backdrop the present invention has been developed. In embodiments of the present invention, the effective size of the memory cache of a peripheral device is virtually increased by allocating unused, available memory space in a host computer to use by the peripheral device cache. According to one example, a method and apparatus is provided for virtually increasing the size of the memory cache of a peripheral device by ascertaining if there is memory space available in a connected host. In this case, the host allocates a portion of a memory space of the host for use as additional cache memory for the peripheral device. The host may provide the peripheral device with the location of the additional memory as well. The peripheral device itself may manage the additional cache memory, and preferably transfers data to and from the additional cache memory via first-party direct memory access.

[0005] These and various other features as well as advantages which characterize the present invention will be apparent from a reading of the following detailed description and a review of the associated drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] **FIG. 1** illustrates an exemplary disc drive.

[0007] **FIG. 2** illustrates an exemplary process for virtually increasing the size of a memory cache of a peripheral device in accordance with an embodiment of the present invention.

[0008] **FIG. 3** illustrates an exemplary process for saving data in the additional cache memory in accordance with an embodiment of the invention.

[0009] **FIG. 4** illustrates an exemplary process for retrieving data from a data storage device such as a disc drive in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION

[0010] In an embodiment of the present invention, a portion of a memory space of a host may be used as additional cache memory for a peripheral device. One such peripheral device may be a data storage device such as a disc drive.

[0011] Referring now to **FIG. 1**, shown therein is a functional block diagram of a disc drive **100**, generally showing the main functional circuits which are resident on the disc drive printed circuit board and used to control the operation of the disc drive **100**. **FIG. 1** illustrates a disc drive for exemplary purposes only; as the embodiments of the present invention can be applied to any peripheral device that has a memory cache, including a disc drive. The disc drive **100** is operably connected to a host computer or other device **140** in a conventional manner. Control communication paths are provided between the host computer **140** and a disc drive microprocessor **142**, the microprocessor **142** generally providing top level communication and control for the disc drive **100** in conjunction with programming for the microprocessor **142** stored in a microprocessor memory (MEM) **143**. The MEM **143** can include random access memory (RAM), read only memory (ROM) and other sources of resident memory for the microprocessor **142**.

[0012] The discs **108** are rotated at a constant high speed by a spindle motor control circuit **148**. During a seek operation; the actuator **110** moves the heads **118** between tracks on the discs **108**. A servo control circuit **150** controls the position of the heads **118**. During a seek operation the microprocessor **142** receives information regarding the velocity of the head **118**, and uses that information in conjunction with a velocity profile stored in memory **143** to communicate with the servo control circuit **150**, thereby causing the actuator assembly **110** to be pivoted.

[0013] Data is transferred between the host computer **140** or other device and the disc drive **100** by way of an interface **144**, which typically includes a buffer to facilitate high-speed data transfer between the host computer **140** or other device and the disc drive **100**. Data to be written to the disc drive **100** is thus passed from the host computer **140** to the interface **144** and then to a read/write channel **146**, which encodes and serializes the data and provides the requisite write current signals to the heads **118**. To retrieve data that has been previously stored in the disc drive **100**, read signals are generated by the heads **118** and provided to the read/write channel **146**, which performs decoding and error detection and correction operations and outputs the retrieved data to the interface **144** for subsequent transfer to the host computer **140** or other device.

[0014] The interface **144** in embodiments of the present invention preferably includes a first-party direct memory access (FPDMA) mechanism. Serial Advanced Technology Attachment (SATA) and Institute of Electrical and Electronics Engineers (IEEE) 1394 are two examples of an interface **144** that includes an FPDMA mechanism.

[0015] Direct memory access (DMA) is a method of direct communication between a peripheral and the buffer memory of a host computer. Typically, the communication between the peripheral and the host computer is controlled by a DMA controller, which is a specialized processor that transfers data between buffer memory and peripheral while allowing

the central processing unit (CPU) to perform other tasks. Typically, the CPU first programs the registers associated with each channel of the DMA controller. The registers in the DMA controller are given a start address of a first buffer in buffer memory where data can be read from or written to, the length of this buffer, and the direction of the data flow. A peripheral requesting a DMA transfer first signals the DMA controller via a DMA request signal. The DMA controller, in turn, responds by returning a corresponding DMA acknowledge signal.

[0016] The DMA controller then directs the transfers, asserting address and strobing lines, with the peripheral asserting or receiving data to or from buffer memory. When the length field of the buffer in the DMA controller goes to zero and there is still data to be transferred, the DMA controller sends the peripheral a signal that the buffer in buffer memory is full or empty, stopping the peripheral's activity. The DMA controller or peripheral also asserts a CPU interrupt signal. In response to the interrupt, the CPU reprograms the DMA controller, giving the DMA controller a start address of a subsequent buffer where data is to be read from or written to, the length of this buffer, and the direction of the data flow. After the DMA controller has been reprogrammed, data transfer resumes.

[0017] FPDMA is an alternative method for DMA in which the peripheral device is a bus master. The peripheral device may have address and control lines that connect the peripheral to the buffer memory or there may be other methods to allow the peripheral device to program the host DMA controller. The address and control lines or other method allow the peripheral device to access information regarding the location of buffers that need to be read or written to without interrupting the CPU. FPDMA allows the peripheral device to access the buffer memory of the host computer under the control of the peripheral device itself. Hardware in the host computer **140** may be configured to allow data to be sent into the memory space of the host computer **140** via FPDMA.

[0018] The buffer in interface **144** is a memory cache. Whenever data is accessed from the disc, the data requested, and additional adjacent data, is stored in the memory cache. ROM in MEM **143** may include code in a module for performing certain acts of the peripheral device in accordance with the present invention, and the host computer **140** preferably includes a driver that includes code for performing certain acts of the host computer **140** in accordance with the present invention.

[0019] **FIG. 2** illustrates an exemplary process **200** that may be provided, for example, in such a code module, for virtually increasing the size of a memory cache of a peripheral device such as disc drive **100**. Process **200** is preferably incorporated into a software module in the ROM **143** of the peripheral device and includes start block **202**, block **204**, decision block **206**, block **208**, block **210**, block **212**, and end block **214**.

[0020] After start block **202**, the process proceeds to block **204**. At block **204**, the peripheral device **100** queries the host computer **140** whether memory is available in the memory space of the host **140**. After block **204**, the process proceeds to decision block **206**. At decision block **206**, the host computer **140** evaluates whether memory is available in the memory space of the host **140**. The process proceeds from decision block **206** to end block **214** when memory is not available in the memory space of the host **140**. The process proceeds from decision block **206** to block **208** when memory is available in the memory space of host **140**.

[0021] At block **208**, the host **140** allocates additional cache memory from the memory space of the host computer **140**. The process proceeds from block **208** to block **210**. At block **210**, the host computer **140** provides the peripheral device with the location of the additional cache memory. According to one example, the host computer **140** responds to the query, the peripheral device **100** receives the response, and the response includes the location of the additional cache memory. The process proceeds from block **210** to block **212**. At block **212**, the peripheral device **100** manages the additional cache memory. The process proceeds from block **212** to end block **214**.

[0022] The additional cache memory may be continuous address space. Alternatively, a table of addresses may be used.

[0023] Process **200** involves a two-part handshake. This handshake may be implemented in several ways. According to one example, the host computer **140** includes a driver that recognizes when the peripheral device **100** is connected to the host computer **140**, and the driver automatically allocates a pre-determined amount of the additional cache memory and provides the peripheral device **100** with the location of the additional cache memory when the peripheral device **100** is connected to the host **140**.

[0024] According to another example, when the peripheral device **100** is connected to the host **140**, the driver queries the peripheral device how much memory the peripheral device **100** needs. The peripheral device **100** then responds to this request. Next, the host **140** allocates the requested amount of memory space as the additional cache memory, and provides the peripheral device **100** with the location of the additional cache memory.

[0025] According to another example, when the peripheral device **100** is connected to the host **140**, the peripheral device **100** makes a query whether a specific amount of memory is available, and the host **140** responds to the query. The host **140** then allocates that amount of memory as additional cache memory and provides the peripheral device **100** with the location of the additional cache memory, if the specific amount of memory is available.

[0026] According to another example, when the peripheral device **100** is connected to the host **140**, the module in the peripheral device **100** makes a query whether any memory is available, and the host **140** responds with the amount of memory available if any. The peripheral device then responds with the amount of memory that it requires, and the host responds in turn with the location of the additional cache memory.

[0027] Process **200** thus effectively allows the peripheral memory cache module to use the additional host cache memory as if it were part of the peripheral memory cache, therefore virtually increasing the size of the peripheral memory cache.

[0028] **FIG. 3** illustrates an exemplary process (**300**) for saving data in the additional cache memory. Process **300** includes start block **302**, decision block **304**, block **306**, and end block **308**.

[0029] After start block **302**, the process proceeds to decision block **304**. At decision block **304**, the module in the peripheral device **100** evaluates whether the peripheral device **100** should send any data from the memory cache of the peripheral device **100** to the additional cache memory. According to one example, the peripheral device **100** evaluates that it should send data from the memory cache to the additional cache memory when the memory is full, and new data is about to be added to the memory cache. However, in various embodiments of process **300**, any criteria may be used for evaluating whether data should be sent from the memory cache to the additional cache memory. The process proceeds from decision block **304** to end block **308** when the peripheral device **100** evaluates that the peripheral device **100** should not send data from the memory cache to the additional cache memory. The process proceeds from decision block **304** to block **306** when the peripheral device **100** evaluates that the peripheral device **100** should send data from the memory cache to the additional cache memory.

[0030] At block **306**, the data is transferred from the memory cache of the peripheral device **100** to the additional cache memory. The transfer is accomplished via FPDMA. The transfer may be accomplished as follows. The interface **144** indicates to the host **140** an address of the host that will be written to. Next, the interface **144** waits to receive a signal from the host **140** indicating that the host **140** is ready to receive data at the indicated address. Then, data is sent to the indicated address. The interface **144** may be a SATA interface that uses the first-party DMA protocol described in the SATA specification to transfer data from the memory cache to the additional cache memory. After block **306**, the process proceeds to end block **308**.

[0031] According to one example, the peripheral device **100** keeps a table of all entries for the additional cache memory. The table may include information such as the starting address of the entry, the size of the entry, and other information describing the data. According to another example, the host computer **140** keeps a table of all entries for the additional cache memory.

[0032] **FIG. 4** illustrates an exemplary process **400** in the module in the peripheral device **100** for retrieving data from the disc drive **100**. Process **400** includes start block **402**, decision block **403**, decision block **404**, block **406**, decision block **408**, block **412**, and end block **414**. According to a first embodiment, process **400** further includes block **410**. According to a second embodiment, process **400** further includes block **416**. Although **FIG. 4** illustrates an example of process **400** in which the peripheral device **100** is a disc drive, any peripheral device with a memory cache can be used.

[0033] The process begins at block **402** in which routine **400** is called. Control then passes to query operational block **403**. In query block **403**, the disc drive **100** module evaluates whether data was requested from the host **140**. The process continually loops back to decision block **403** if data is not being requested from the host **140**. However, control transfers from decision block **403** to decision block **404** when data is requested from the host **140**.

[0034] In decision block **404**, the disc drive **100** evaluates whether the requested data is in the memory cache of the disc drive **100**. If the requested data is not in the memory cache control transfers from decision block **404** to decision block **408**. Control transfers from decision block **404** to block **406** if the requested data is in the memory cache. At block **406**, the data is transferred from the memory cache to the host. The transfer is accomplished via whatever protocol host **140** is expecting. Control then transfers from block **406** to end block **414** where this routine ends.

[0035] If on the other hand, control transferred to decision block **408** because the data was not in the memory cache, it is evaluated whether the requested data is in the additional cache memory. The evaluation may be made by searching a table of entries for the additional cache memory that is kept by the disc drive **100**. Alternatively, disc drive **100** may retrieve a table of entries for the additional cache memory that is kept by the host **140**, or in another location, and then the disc drive **100** searches the table of entries in order to evaluate whether the requested data is stored in the additional cache memory.

[0036] Control transfers from decision block **408** to block **412** if the requested data is not in the additional cache memory. If the requested data is in additional memory, control transfers to retrieve the data. According to the first embodiment, the process proceeds from decision block **408** to block **410** when the requested data is in the additional cache memory. According to the second embodiment, the process proceeds from decision block **408** to block **416** when the requested data is in the additional cache memory.

[0037] According to the first embodiment, in operational block **410**, data is transferred from the additional cache memory to the memory cache of the disc drive **100**. The transfer is accomplished preferably via FPDMA. The transfer may be accomplished as follows. The interface **144** indicates to the host an address of the host that will be read from. The interface **144** of the disc drive **100** then waits to receive a signal from the host indicating that the data at the indicated address is ready to be read. When the signal is received, the requested data is read from the indicated address. The interface **144** may be a SATA interface that uses the first-party DMA protocol described in the SATA specification to transfer data from the memory cache to the additional cache memory. The process proceeds from block **410** to block **406**.

[0038] According to the second embodiment, at block **416**, the host **140** is provided with the location of the requested data in the memory space of the host. The process proceeds from block **416** to end block **414** where control returns to the calling routine.

[0039] At block **412**, data is retrieved from the disc **108** and sent to the host **140**. The process proceeds from block **412** to end block **414** where control again returns to the calling routine.

[0040] In summary, an embodiment of the present invention may be viewed as a method (such as **200**) for virtually increasing a size of a cache (such as **144**) of a peripheral device (such as **100**) that is connectable to a host (such as **140**). The method includes querying the host (such as in acts **204** and **206**) whether memory is available in a memory space of the host (such as **140**). The method also includes allocating (such as in act **208**) additional cache memory from the memory space of the host for use by the peripheral device if memory is available in the memory space of the host.

[0041] Alternatively, an embodiment of the present invention may be viewed as a peripheral device (such as **100**) that is connectable to a host computer (such as **140**). The peripheral device includes a memory cache (such as **144**). The peripheral device also includes means for using a portion of a memory space of the host computer as additional cache memory for the peripheral device (such as in acts **202** through **214**). The using means may be implemented in the ROM (such as **143**) of the peripheral device. The using means includes means for making a query to the host (such as in act **206**) whether memory is available in the memory space of the host. The using mean also includes means for receiving a response (such as in act **210**) to the query if memory is available in the memory space of the host, wherein the response includes a location of the additional cache memory.

[0042] The using means also includes means for evaluating whether data should be transferred from the memory cache of the peripheral device to the additional cache memory in the memory space of the host (such as in acts **302** through **308**). The using means also includes means for transferring data (such as in acts **402** through **414**) from the memory cache of the peripheral device to the additional cache memory via first-party direct memory access (DMA) when data should be transferred from the memory cache of the peripheral device to the additional cache memory. The using means also includes means for evaluating whether requested data is in the additional cache memory (such as in acts **408** and **410**) if the host requests the requested data.

[0043] It will be clear that the present invention is well adapted to attain the ends and advantages mentioned as well as those inherent therein. While a presently preferred embodiment has been described for purposes of this disclosure, various changes and modifications may be made which are well within the scope of the present invention. For example, some of the examples described above used a disc drive for purposes of illustration. However, the invention can be applied to any peripheral device that has a memory cache, such as a printer or scanner, not just a disc drive. Numerous other changes may be made which will readily suggest themselves to those skilled in the art and which are encompassed in the spirit of the invention disclosed and as defined in the appended claims.

1-30. (canceled)

31. A method for virtually increasing a size of a cache of a peripheral device that is connectable to a second device, comprising:

querying the second device whether memory is available in a memory space of the second device; and

allocating additional cache memory from the memory space of the second device for use by the peripheral device if memory is available in the memory space of the second device.

32. The method of claim 31, further comprising:

managing the additional cache memory via the peripheral device.

33. The method of claim 32, further comprising:

providing the peripheral device with a location of the additional cache memory assigned.

34. The method of claim 32, wherein managing the additional cache memory comprises:

evaluating whether data should be transferred from the cache of the peripheral device to the additional cache memory; and

transferring data from the cache of the peripheral device to the additional cache memory if the data should be transferred.

35. The method of claim 32, wherein managing the additional cache memory further comprises:

if the second device requests data, evaluating whether the requested data is in the cache of the peripheral device;

transferring the requested data to the second device if the requested data is in the cache of the peripheral device; and

evaluating whether the requested data is in the additional cache memory if the requested data is not in the cache of the peripheral device.

36. The method of claim 34, wherein managing the additional cache memory further comprises:

transferring the requested data from the additional cache memory to the cache of the peripheral device via first-party direct memory access if the requested data is in the additional cache memory.

37. The method of claim 32, wherein managing the additional cache memory further comprises:

providing the second device with a location of data requested by the second device in the memory space of the second device if the requested data is in the additional cache memory.

38. The method of claim 34, wherein evaluating whether data should be transferred comprises:

evaluating whether the cache of the peripheral device is full if new data is to be stored in the cache of the peripheral device.

39. The method of claim 34, wherein transferring data from the cache of the peripheral device comprises:

indicating to the second device an address of the second device that will be written to;

waiting to receive a signal from the second device indicating that the second device is ready to receive data at the indicated address; and

sending the data to the indicated address upon receipt of the signal from the second device.

40. The method of claim 34, wherein transferring data from the cache is performed with a serial advanced technology attachment (ATA) protocol, wherein the serial ATA protocol includes a first-party direct memory access mechanism.

41. The method of claim 35, wherein evaluating whether the requested data is in the additional cache memory comprises:

searching a cache table that is stored in the memory of the peripheral device.

42. The method of claim 35, wherein evaluating whether the requested data is in the additional cache memory comprises:

retrieving a cache table from the second device; and

searching the cache table for the requested data.

43. The method of claim 36, wherein transferring data from the additional cache memory is performed with a serial advanced technology attachment (ATA) protocol, wherein the serial ATA protocol includes a first-party direct memory access mechanism.

44. The method of claim 31, wherein the peripheral device is a hard disk drive, wherein querying the second device comprises querying a host computer whether memory is available in a memory space of the host computer; and wherein allocating additional cache memory comprises allocating additional cache memory from the memory space of the host computer for use by the hard disk drive if memory is available in the memory space of the host computer.

45. A method for virtually increasing a size of a cache of a peripheral device that is connectable to a host computer, comprising:

querying the host computer whether memory is available in a memory space of the host computer; and

allocating additional cache memory from the memory space of the host computer for use by the peripheral device if memory is available in the memory space of the host computer.

46. The method of claim 45, wherein the data storage device comprises a hard disc drive.

47. The method of claim 45, further comprising:

indicating to the host computer an address in the memory space of the host computer that will be read from if requested data is in the additional cache memory;

waiting to receive a signal from the host computer indicating that the data at the indicated address is ready to be read; and

reading the data from the indicated address upon receipt of the host signal.

48. A peripheral device connectable to a second device having a memory space, the peripheral device comprising:

a cache memory; and

a module that allocates additional cache memory from the memory space of the second device for use by the peripheral device.

49. The device of claim 48 wherein the module further manages operation of the additional cache memory.

50. The device of claim 48 wherein the module further stores a location of the additional cache memory.

51. The device of claim 48, wherein the module further evaluates whether data should be transferred from the cache of the peripheral device to the additional cache memory, and manages transfer of data from the cache of the peripheral device to the additional cache memory if the data should be transferred.

52. The device of claim 51, wherein the module further evaluates whether data requested by the host is in the cache of the peripheral device, transfers the requested data to the second device if the requested data is in the cache of the peripheral device, and evaluates whether the requested data is in the additional cache memory if the requested data is not in the cache of the peripheral device.

53. The device of claim 52, wherein the module transfers the requested data from the additional cache memory to the cache of the peripheral device via first-party direct memory access if the requested data is in the additional cache memory.

54. The device of claim 51, wherein the module evaluates whether the cache of the peripheral device is full if new data is to be stored in the cache of the peripheral device.

55. The device of claim 51, wherein the module indicates to the second device an address of the second device that will be written to, waits to receive a signal from the second device indicating that the second device is ready to receive data at the indicated address, and sends the data to the indicated address upon receipt of the signal.

56. The device of claim 51, wherein the data transferred from the cache is transferred via a serial advanced technology attachment (ATA) protocol, wherein the serial ATA protocol includes a first-party direct memory access mechanism.

57. The device of claim 48, wherein the module further provides the second device with a location of data requested by the second device if the requested data is in the additional cache memory.

58. The device of claim 48, wherein the module is adapted to search a cache table that is stored in the memory of the peripheral device.

59. The device of claim 48, wherein the peripheral device is a data storage device.

60. The device of claim 48, wherein the peripheral device is a hard disc drive.

61. The device of claim 48, wherein the second device is a host computer.

* * * * *