



(19) **United States**

(12) **Patent Application Publication**  
**Borchers et al.**

(10) **Pub. No.: US 2010/0262773 A1**

(43) **Pub. Date: Oct. 14, 2010**

(54) **DATA STRIPING IN A FLASH MEMORY DATA STORAGE DEVICE**

**Publication Classification**

(75) Inventors: **Albert T. Borchers**, Santa Cruz, CA (US); **Andrew T. Swing**, Los Gatos, CA (US); **Robert S. Sprinkle**, Mountain View, CA (US)

(51) **Int. Cl.**  
**G06F 12/00** (2006.01)  
**G06F 12/02** (2006.01)

(52) **U.S. Cl.** ..... **711/114; 711/103; 711/E12.001; 711/E12.008**

Correspondence Address:

**BRAKE HUGHES BELLERMANN LLP**  
**c/o CPA Global**  
**PO Box 52050**  
**Minneapolis, MN 55402 (US)**

(73) Assignee: **Google Inc.**, Mountain View, CA (US)

(21) Appl. No.: **12/537,738**

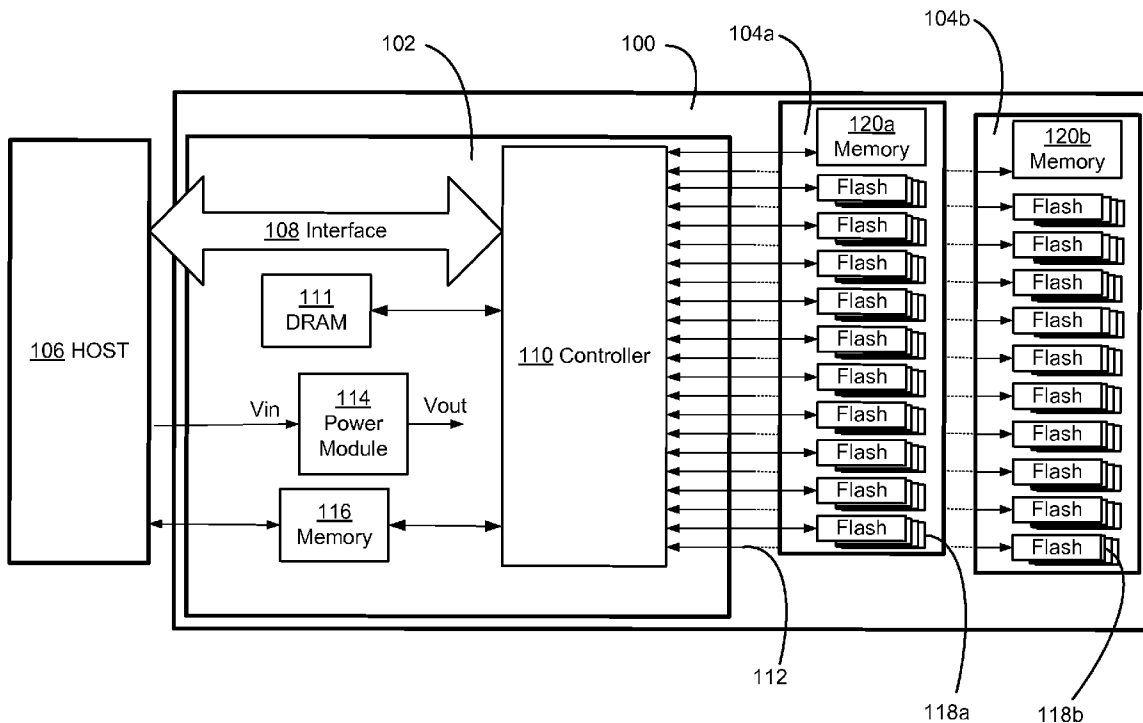
(22) Filed: **Aug. 7, 2009**

**Related U.S. Application Data**

(60) Provisional application No. 61/187,835, filed on Jun. 17, 2009, provisional application No. 61/167,709, filed on Apr. 8, 2009.

(57) **ABSTRACT**

A method is disclosed for striping data from a host to a data storage device that includes a plurality of memory chips and a plurality of physical channels for communication of data between the host and the plurality of memory chips, where each channel is operably connected to a different plurality of the memory chips. The method includes determining a number of physical channels in the plurality of channels, determining a first channel chunk size with which to write data to memory chips connected to separate channels, segmenting, via the host, logically sequential data into first channel chunk size segments, and striping data to different channels of the data storage device in first channel chunk size units.



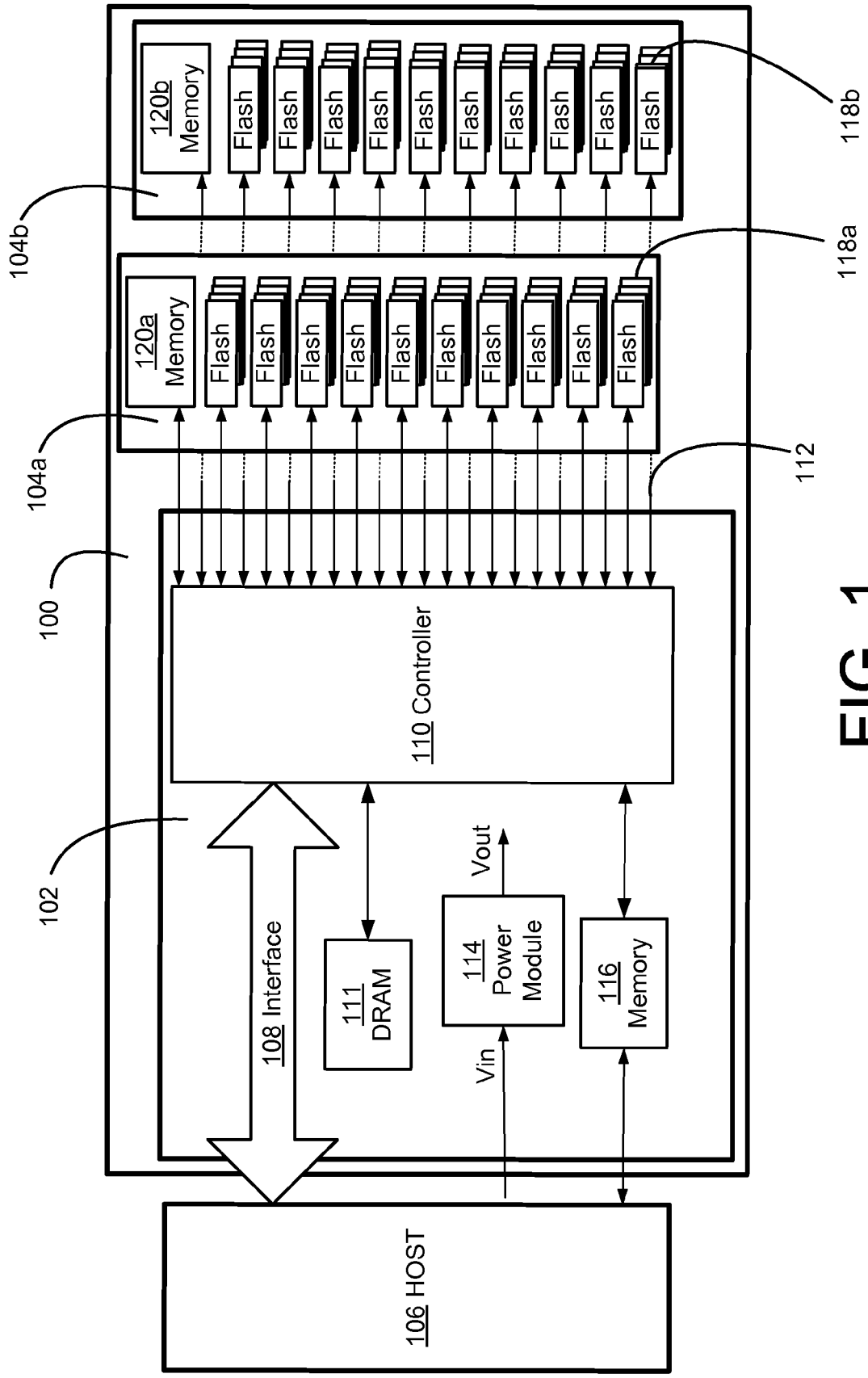


FIG. 1

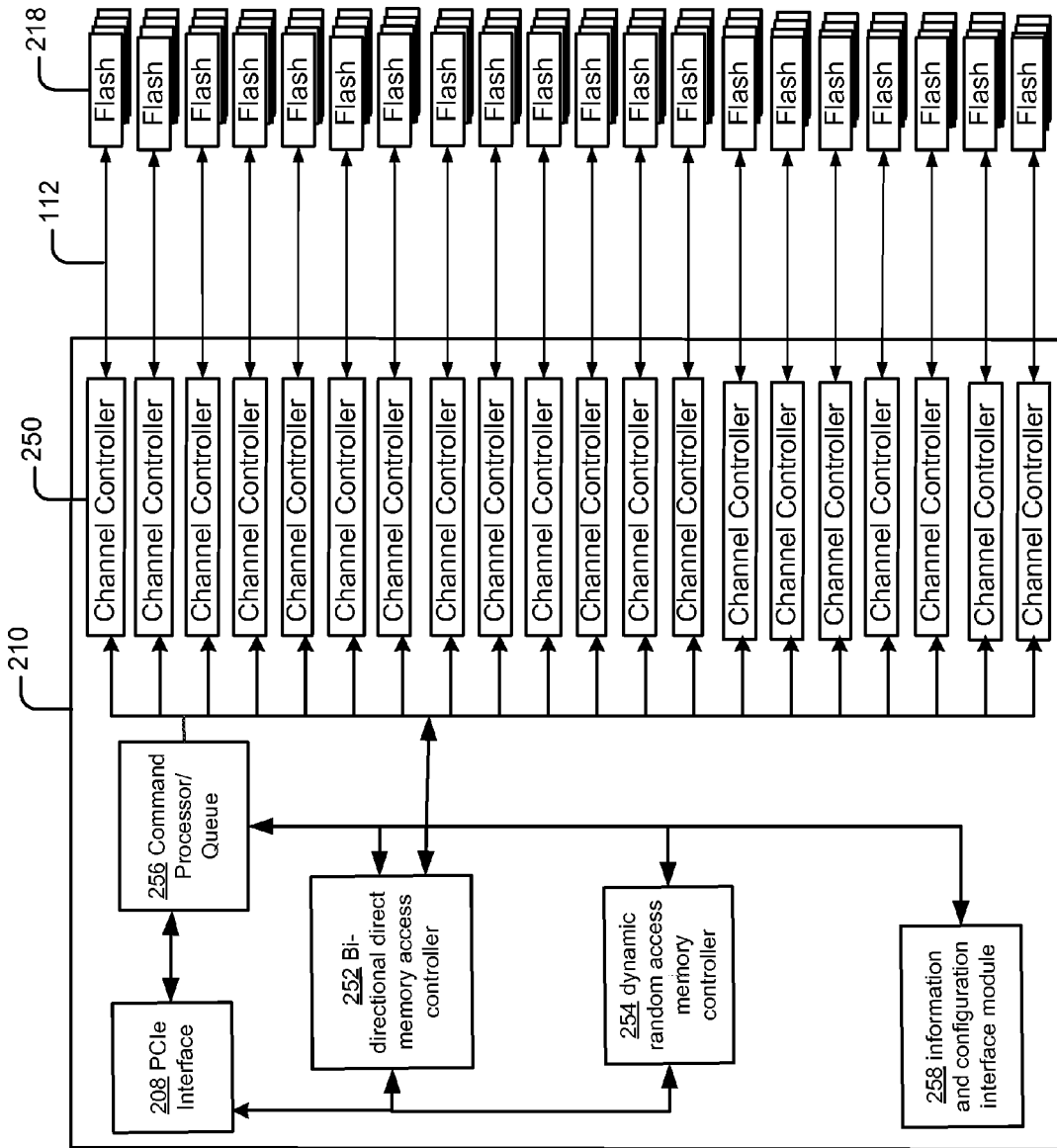


FIG. 2

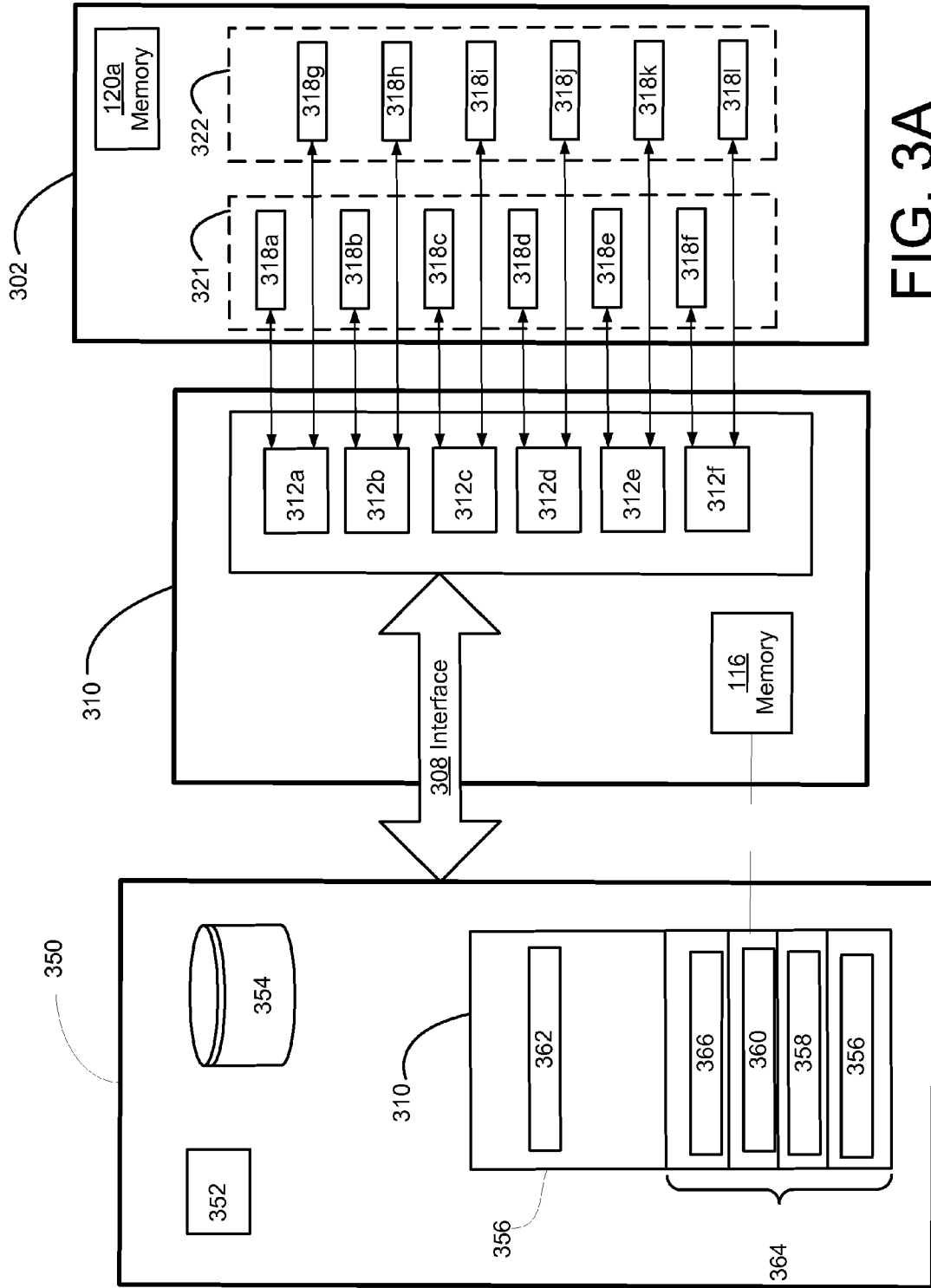


FIG. 3A



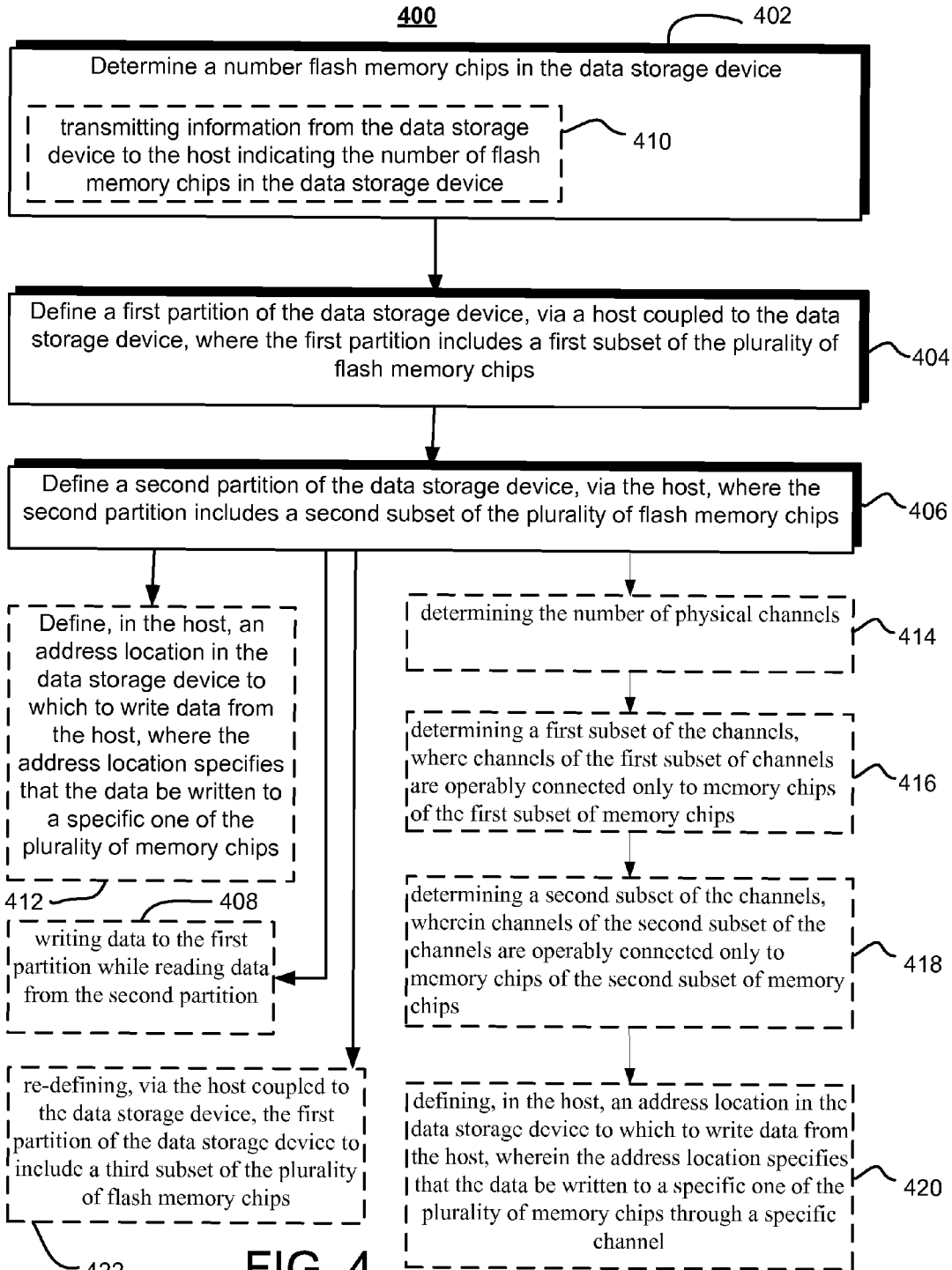


FIG. 4

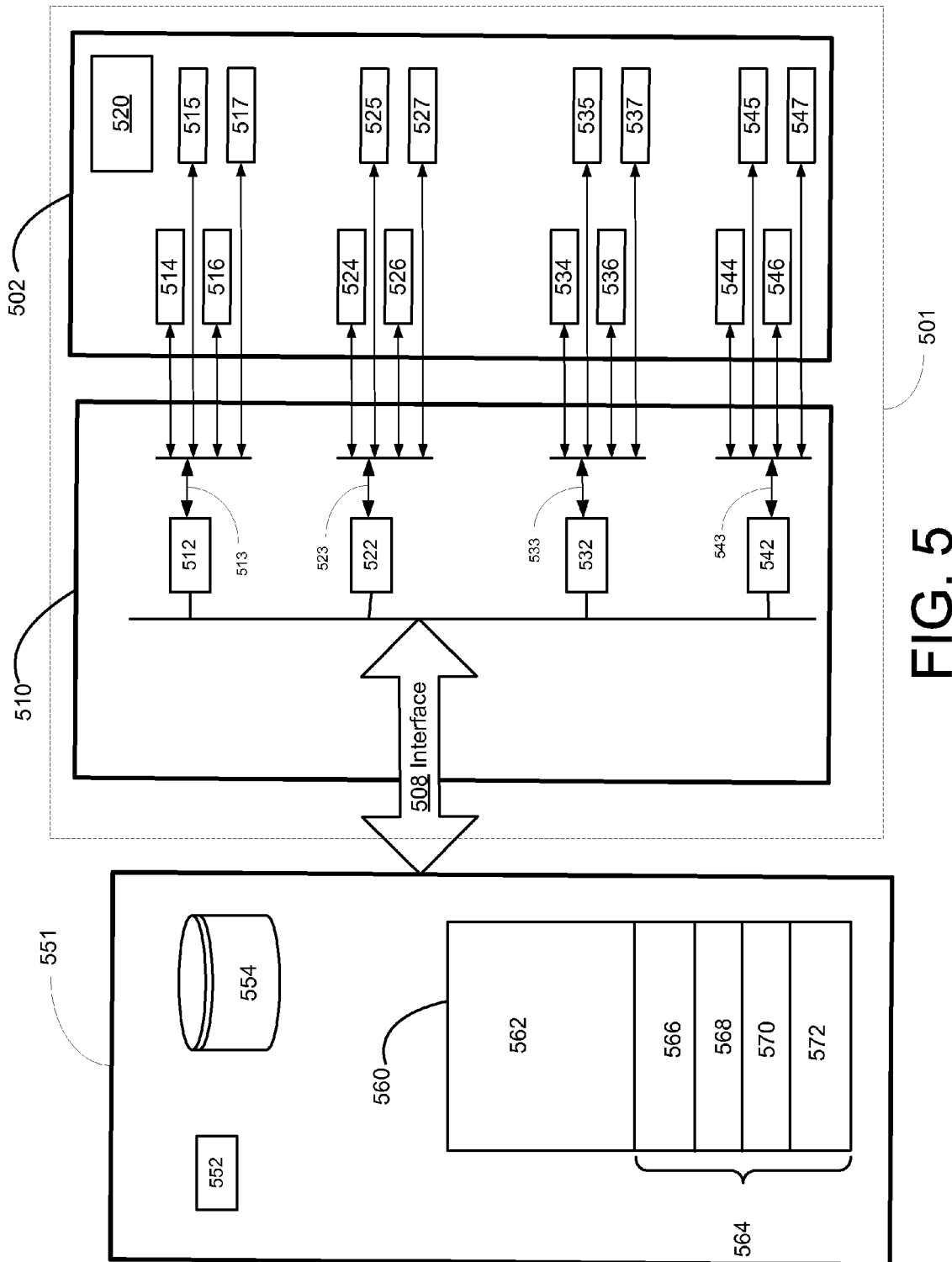


FIG. 5

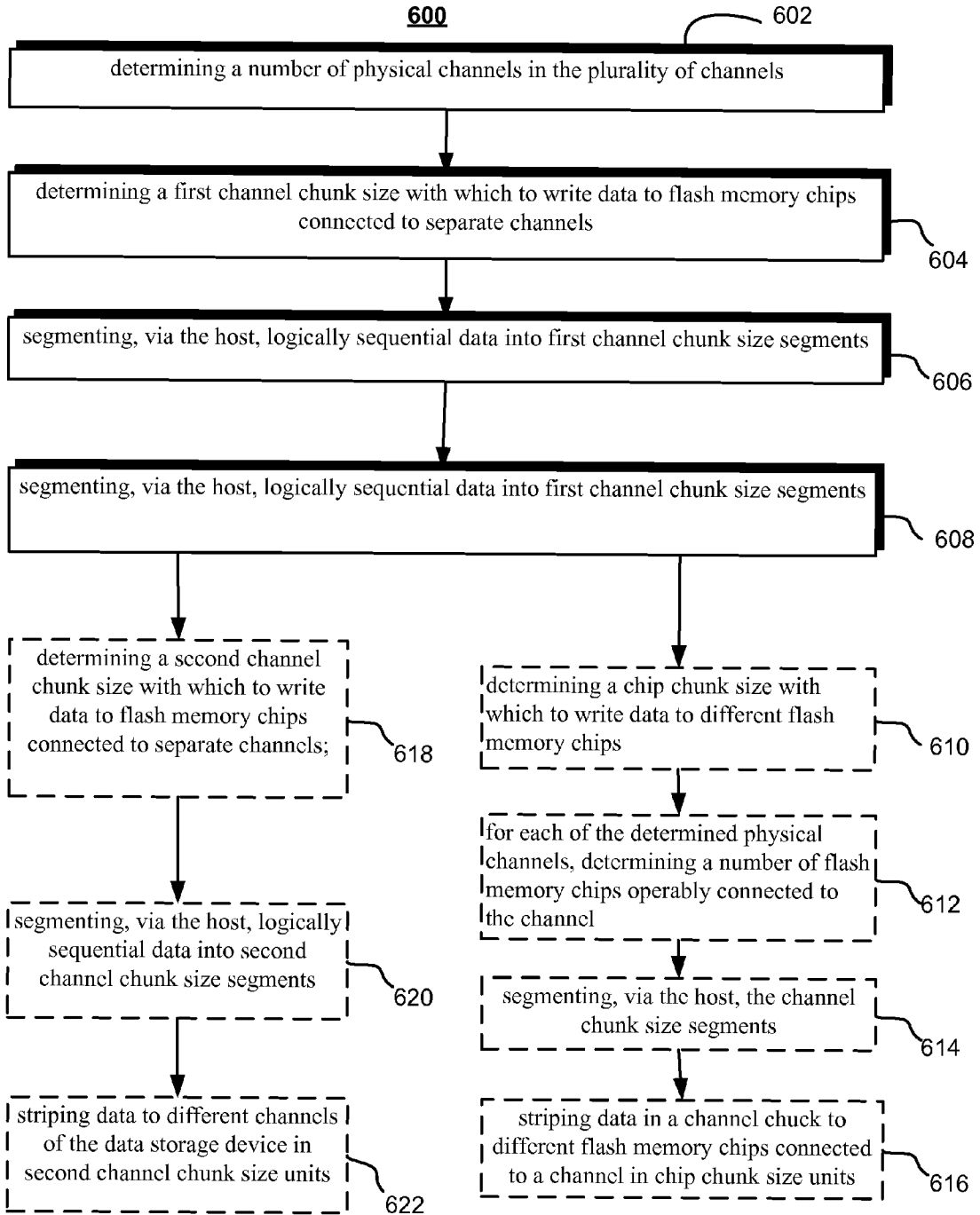


FIG. 6



**DATA STRIPING IN A FLASH MEMORY DATA STORAGE DEVICE**

**CROSS REFERENCE TO RELATED APPLICATIONS**

[0001] This application claims the benefit of U.S. Provisional Application No. 61/167,709, filed Apr. 8, 2009, and titled "Data Storage Device" and U.S. Provisional Application No. 61/187,835, filed Jun. 17, 2009, and titled "Partitioning and Striping in a Flash Memory Data Storage Device," both of which are hereby incorporated by reference in entirety.

**TECHNICAL FIELD**

[0002] This description relates to a data storage device.

**BACKGROUND**

[0003] Data storage devices may be used to store data. A data storage device may be used with a computing device to provide for the data storage needs of the computing device. In certain instances, it may be desirable to store large amounts of data on a data storage device. Also, it may be desirable to execute commands quickly to read data from and to write data to the data storage device.

**SUMMARY**

[0004] In a general aspect, a method is disclosed for striping data from a host to a data storage device that includes a plurality of memory chips and a plurality of physical channels for communication of data between the host and the plurality of memory chips, where each channel is operably connected to a different plurality of the memory chips. The method includes determining a number of physical channels in the plurality of channels, determining a first channel chunk size with which to write data to memory chips connected to separate channels, segmenting, via the host, logically sequential data into first channel chunk size segments, and striping data to different channels of the data storage device in first channel chunk size units.

[0005] According to one general aspect, another method is disclosed for striping data from a host to a data storage device that includes a plurality of memory chips and a plurality of physical channels for communication of data between the host and the plurality of memory chips, where each channel is operably connected to a different plurality of the memory chips. The method includes determining a number of physical channels in the plurality of channels, and for each of the determined physical channels, determining a number of memory chips operably connected to the channel. A first channel chunk size with which to write data to memory chips connected to separate channels is determined, and a chip chunk size with which to write data to different memory chips is determined. Logically sequential data is segmented, via the host, into first channel chunk size segments, and the first channel chunk size segments are segmented, via the host, into chip chunk size segments. Data is striped to different channels of the data storage device in first channel chunk size units, and data in a first channel chunk sized segment is striped to different memory chips connected to a channel in chip chunk size units.

[0006] Implementations can include one or more of the following features. For example, the logically sequential data can consist of a data file. Data can be written to a first channel

while reading data from a second channel. Determining the number of physical channels in the plurality of channels can include transmitting information from the data storage device to the host indicating the number of channels in the data storage device or can include reading data stored on the host indicating the number of channels in the data storage device. In addition, a second channel chunk size with which to write data to memory chips connected to separate channels can be determined and logically sequential data can be segmented, via the host, into second channel chunk size segments. Data can be striped to different channels of the data storage device in second channel chunk size units, can data in a second channel chunk sized segment can be striped to different memory chips connected to a channel in chip chunk size units. The first channel chunk sized can be determined based on input from a user entered via the host.

[0007] In another general aspect, an apparatus can include a host and a data storage device that includes a plurality of memory chips and a plurality of physical channels for communication of data between the host and the plurality of memory chips, where each channel is operably connected to a different plurality of the memory chips. The host can be coupled to the data storage device via an interface and can include an initialization engine configured to determine a first channel chunk size with which to write data to memory chips connected to separate channels, a segmentation engine configured to segment logically sequential data into first channel chunk size segments, and a striping engine configured to stripe data to different channels of the data storage device in first channel chunk size units.

[0008] Implementations can include one or more of the following features. For example, the data storage device can include a storage medium configured to store the number of channels, where the data storage device is configured to transmit, upon receiving a command from the host, information from the data storage device to the host indicating the number of channels in the data storage device. The host can further include an address assignment engine configured to assign a memory address to data to be written to the data storage device, where the assigned memory address specifies that the data be written to a specific one of the channels. The striping engine can be configured to write data to a first channel while reading data from a second channel. The initialization engine can be further configured to determine a second channel chunk size, different from the first channel chunk size, with which to write data to memory chips connected to separate channels, where the segmentation engine is further configured to segment logically sequential data into second channel chunk size segments, and where the striping engine is further configured to stripe data to different channels of the data storage device in second channel chunk size units. The initialization engine can be further configured to determine a chip chunk size with which to write data to different memory chips, where the segmentation engine is further configured to segment the first channel chunk size segments into chip chunk size segments, and where the striping engine is further configured to stripe data to different chips connected to a channel in first channel chunk size units.

[0009] The details of one or more implementations are set forth in the accompanying drawings and the description below. Other features will be apparent from the description and drawings, and from the claims.

**BRIEF DESCRIPTION OF THE DRAWINGS**

[0010] FIG. 1 is an exemplary block diagram of a data storage device.

[0011] FIG. 2 is an exemplary block diagram of a FPGA controller that can be used in the data storage device of FIG. 1.

[0012] FIG. 3A is an exemplary block diagram of exemplary computing devices for use with the data storage device of FIG. 1.

[0013] FIG. 3B is an exemplary block diagram of exemplary computing devices for use with the data storage device of FIG. 1.

[0014] FIG. 4 is an exemplary flowchart illustrating an example process of partitioning the data storage device of FIG. 1.

[0015] FIG. 5 is an exemplary block diagram of an example implementation of the data storage device of FIG. 1.

[0016] FIG. 6 is an exemplary flowchart illustrating example operations of the data storage device of FIG. 1.

#### DETAILED DESCRIPTION

[0017] This document describes an apparatus, system(s) and techniques for data storage. Such a data storage apparatus may include a controller board having a controller that may be used with one or more different memory boards, with each of the memory boards having multiple flash memory chips. The data storage apparatus may communicate with a host using an interface on the controller board. In this manner, the controller on the controller board may be configured to receive commands from the host using the interface and to execute those commands using the flash memory chips on the memory boards.

[0018] FIG. 1 is a block diagram of a data storage device 100. The data storage device 100 may include a controller board 102 and one or more memory boards 104a and 104b. The data storage device 100 may communicate with a host 106 over an interface 108. The interface 108 may be between the host 106 and the controller board 102. The controller board 102 may include a controller 110, a DRAM 111, multiple channels 112, a power module 114, and a memory module 116. The memory boards 104a and 104b may include multiple flash memory chips 118a and 118b on each of the memory boards. The memory boards 104a and 104b also may include a memory device 120a and 120b.

[0019] In general, the data storage device 100 may be configured to store data on the flash memory chips 118a and 118b. The host 106 may write data to and read data from the flash memory chips 118a and 118b, as well as cause other operations to be performed with respect to the flash memory chips 118a and 118b. The reading and writing of data between the host 106 and the flash memory chips 118a and 118b, as well as the other operations, may be processed through and controlled by the controller 110 on the controller board 102. The controller 110 may receive commands from the host 106 and cause those commands to be executed using the flash memory chips 118a and 118b on the memory boards 104a and 104b. The communication between the host 106 and the controller 110 may be through the interface 108. The controller 110 may communicate with the flash memory chips 118a and 118b using the channels 112.

[0020] The controller board 102 may include DRAM 111. The DRAM 111 may be operably coupled to the controller 110 and may be used to store information. For example, the DRAM 111 may be used to store logical address to physical address maps and bad block information. The DRAM 111 also may be configured to function as a buffer between the host 106 and the flash memory chips 118a and 118b.

[0021] In one exemplary implementation, the controller board 102 and each of the memory boards 104a and 104b are physically separate printed circuit boards (PCBs). The memory board 104a may be on one PCB that is operably connected to the controller board 102 PCB. For example, the memory board 104a may be physically and/or electrically connected to the controller board 102. Similarly, the memory board 104b may be a separate PCB from the memory board 104a and may be operably connected to the controller board 102 PCB. For example, the memory board 104b may be physically and/or electrically connected to the controller board 102.

[0022] The memory boards 104a and 104b each may be separately disconnected and removable from the controller board 102. For example, the memory board 104a may be disconnected from the controller board 102 and replaced with another memory board (not shown), where the other memory board is operably connected to controller board 102. In this example, either or both of the memory boards 104a and 104b may be swapped out with other memory boards such that the other memory boards may operate with the same controller board 102 and controller 110.

[0023] In one exemplary implementation, the controller board 102 and each of the memory boards 104a and 104b may be physically connected in a disk drive form factor. The disk drive form factor may include different sizes such as, for example, a 3.5" disk drive form factor and a 2.5" disk drive form factor.

[0024] In one exemplary implementation, the controller board 102 and each of the memory boards 104a and 104b may be electrically connected using a high density ball grid array (BGA) connector. Other variants of BGA connectors may be used including, for example, a fine ball grid array (FBGA) connector, an ultra fine ball grid array (UBGA) connector and a micro ball grid array (MBGA) connector. Other types of electrical connection means also may be used.

[0025] The interface 108 may include a high speed interface between the controller 110 and the host 106. The high speed interface may enable fast transfers of data between the host 106 and the flash memory chips 118a and 118b. In one exemplary implementation, the high speed interface may include a Peripheral Component Interconnect Express ("PCIe") interface. For instance, the PCIe interface may be a PCIe x4 interface or a PCIe x8 interface. The PCIe interface 108 may include a PCIe connector cable assembly to the host 106. In this example, the 110 may include an interface controller configured to interface between the host 106 and the interface 108. The interface controller may include a PCIe endpoint controller. Other high speed interfaces, connectors, and connector assemblies also may be used.

[0026] In one exemplary implementation, the communication between the controller board 102 and the flash memory chips 118a and 118b on the memory boards 104a and 104b may be arranged and configured into multiple channels 112. Each of the channels 112 may communicate with one or more flash memory chips 118a and 118b. The controller 110 may be configured such that commands received from the host 106 may be executed by the controller 110 using each of the channels 112 simultaneously or at least substantially simultaneously. In this manner, multiple commands may be executed simultaneously on different channels 112, which may improve throughput of the data storage device 100.

[0027] In the example of FIG. 1, twenty (20) channels 112 are illustrated. The completely solid lines illustrate the ten

(10) channels between the controller **110** and the flash memory chips **118a** on the memory board **104a**. The mixed solid and dashed lines illustrate the ten (10) channels between the controller **110** and the flash memory chips **118b** on the memory board **104b**. As illustrated in FIG. 1, each of the channels **112** may support multiple flash memory chips. For instance, each of the channels **112** may support up to 32 flash memory chips. In one exemplary implementation, each of the 20 channels may be configured to support and communicate with 6 flash memory chips. In this example, each of the memory boards **104a** and **104b** would include 60 flash memory chips each. Depending on the type and the number of the flash memory chips **118a** and **118b**, the data storage **100** device may be configured to store up to and including multiple terabytes of data.

[0028] The controller **110** may include a microcontroller, a FPGA controller, other types of controllers, or combinations of these controllers. In one exemplary implementation, the controller **110** is a microcontroller. The microcontroller may be implemented in hardware, software, or a combination of hardware and software. For example, the microcontroller may be loaded with a computer program product from memory (e.g., memory module **116**) including instructions that, when executed, may cause the microcontroller to perform in a certain manner. The microcontroller may be configured to receive commands from the host **106** using the interface **108** and to execute the commands. For instance, the commands may include commands to read, write, copy and erase blocks of data using the flash memory chips **118a** and **118b**, as well as other commands.

[0029] In another exemplary implementation, the controller **110** is a FPGA controller. The FPGA controller may be implemented in hardware, software, or a combination of hardware and software. For example, the FPGA controller may be loaded with firmware from memory (e.g., memory module **116**) including instructions that, when executed, may cause the FPGA controller to perform in a certain manner. The FPGA controller may be configured to receive commands from the host **106** using the interface **108** and to execute the commands. For instance, the commands may include commands to read, write, copy and erase blocks of data using the flash memory chips **118a** and **118b**, as well as other commands.

[0030] The memory module **116** may be configured to store data, which may be loaded to the controller **110**. For instance, the memory module **116** may be configured to store one or more images for the FPGA controller, where the images include firmware for use by the FPGA controller. The memory module **116** may interface with the host **106** to communicate with the host **106**. The memory module **116** may interface directly with the host **106** and/or may interface indirectly with the host **106** through the controller **110**. For example, the host **106** may communicate one or more images of firmware to the memory module **116** for storage. In one exemplary implementation, the memory module **116** includes an electrically erasable programmable read-only memory (EEPROM). The memory module **116** also may include other types of memory modules.

[0031] The memory boards **104a** and **104b** may be configured to operate with different types of flash memory chips **118a** and **118b**. In one exemplary implementation, the flash memory chips **118a** and the flash memory chips **118b** may be the same type of flash memory chips including requiring the same voltage from the power module **114** and being from the

same flash memory chip vendor. The terms vendor and manufacturer are used interchangeably throughout this document.

[0032] In another exemplary implementation, the flash memory chips **118a** on the memory board **104a** may be a different type of flash memory chip from the flash memory chips **118b** on the memory board **104b**. For example, the memory board **104a** may include SLC NAND flash memory chips and the memory board **104b** may include MLC NAND flash memory chips. In another example, the memory board **104a** may include flash memory chips from one flash memory chip manufacturer and the memory board **104b** may include flash memory chips from a different flash memory chip manufacturer. The flexibility to have all the same type of flash memory chips or to have different types of flash memory chips enables the data storage device **100** to be tailored to different applications being used by the host **106**.

[0033] In another exemplary implementation, the memory boards **104a** and **104b** may include different types of flash memory chips on the same memory board. For example, the memory board **104a** may include both SLC NAND chips and MLC NAND chips on the same PCB. Similarly, the memory board **104b** may include both SLC NAND chips and MLC NAND chips. In this manner, the data storage device **100** may be advantageously tailored to meet the specifications of the host **106**.

[0034] In another exemplary implementation, the memory board **104a** and **104b** may include other types of memory devices, including non-flash memory chips. For instance, the memory boards **104a** and **104b** may include random access memory (RAM) such as, for instance, dynamic RAM (DRAM) and static RAM (SRAM) as well as other types of RAM and other types of memory devices. In one exemplary implementation, the both of the memory boards **104a** and **104b** may include RAM. In another exemplary implementation, one of the memory boards may include RAM and the other memory board may include flash memory chips. Also, one of the memory boards may include both RAM and flash memory chips.

[0035] The memory modules **120a** and **120b** on the memory boards **104a** and **104b** may be used to store information related to the flash memory chips **118a** and **118b**, respectively. In one exemplary implementation, the memory modules **120a** and **120b** may store device characteristics of the flash memory chips. The device characteristics may include whether the chips are SLC chips or MLC chips, whether the chips are NAND or NOR chips, a number of chip selects, a number of blocks, a number of pages per block, a number of bytes per page and a speed of the chips.

[0036] In one exemplary implementation, the memory modules **120a** and **120b** may include serial EEPROMs. The EEPROMs may store the device characteristics. The device characteristics may be compiled once for any given type of flash memory chip and the appropriate EEPROM image may be generated with the device characteristics. When the memory boards **104a** and **104b** are operably connected to the controller board **102**, then the device characteristics may be read from the EEPROMs such that the controller **110** may automatically recognize the types of flash memory chips **118a** and **118b** that the controller **110** is controlling. Additionally, the device characteristics may be used to configure the controller **110** to the appropriate parameters for the specific type or types of flash memory chips **118a** and **118b**.

[0037] As discussed above, the controller **110** may include a FPGA controller. Referring to FIG. 2, an exemplary block

diagram of a FPGA controller **210** is illustrated. The FPGA controller may be configured to operate in the manner described above with respect to controller **110** of FIG. **1**. The FPGA controller **210** may include multiple channel controllers **250** to connect the multiple channels **112** to the flash memory chips **218**. The flash memory chips **218** are illustrated as multiple flash memory chips that connect to each of the channel controllers **250**. The flash memory chips **218** are representative of the flash memory chips **118a** and **118b** of FIG. **1**, which are on the separate memory boards **104a** and **104b** of FIG. **1**. The separate memory boards are not shown in the example of FIG. **2**. The FPGA controller **210** may include a PCIe interface module **208**, a bi-directional direct memory access (DMA) controller **252**, a dynamic random access memory (DRAM) controller **254**, a command processor/queue **256** and an information and configuration interface module **258**.

**[0038]** Information may be communicated with a host (e.g., host **106** of FIG. **1**) using an interface. In this example, FIG. **2**, the FPGA controller **210** includes a PCIe interface to communicate with the host and a PCIe interface module **208**. The PCIe interface module **208** may be arranged and configured to receive commands from the host and to send commands to the host. The PCIe interface module **208** may provide data flow control between the host and the data storage device. The PCIe interface module **208** may enable high speed transfers of data between the host and the controller **210** and ultimately the flash memory chips **218**. In one exemplary implementation, the PCIe interface and the PCIe interface module **208** may include a 64-bit bus. The bi-directional direct memory access (DMA) controller **252** may be arranged and configured to control the operation of the bus between the PCIe interface module **208** and the command processor/queue **256**.

**[0039]** The bi-directional DMA controller **252** may be configured to interface with the PCIe interface **208**, and each of the channel controllers **250**. The bi-directional DMA controller **252** enables bi-directional direct memory access between the host **106** and the flash memory chips **218**.

**[0040]** The DRAM controller **254** may be arranged and configured to control the translation of logical to physical addresses. For example, in an implementation in which the host addresses the memory space using logical addresses, the DRAM controller **254** may assist the command processor/queue **256** with the translation of the logical addresses used by the host to the actual physical addresses in the flash memory chips **218** related to data being written to or read from the flash memory chips **218**. A logical address received from the host may be translated to a physical address for a location in one of the flash memory chips **218**. Similarly, a physical address for a location in one of the flash memory chips **218** may be translated to a logical address and communicated to the host.

**[0041]** The command processor/queue **256** may be arranged and configured to receive the commands from the host through the PCIe interface module **208** and to control the execution of the commands through the channel controllers **250**. The command processor/queue **256** may maintain a queue for a number of commands to be executed and order the commands using an ordered list to ensure that the oldest commands may be processed first. The command processor **100** may maintain the order of the commands designated for the same flash memory chip and may reorder the commands designated for different flash memory chips. In this manner,

multiple commands may be executed simultaneously and each of the channels **112** may be used simultaneously or at least substantially simultaneously.

**[0042]** The command processor/queue **256** may be configured to process commands for different channels **112** out of order and preserve per-channel command ordering. For instance, commands that are received from the host and that are designated for different channels may be processed out of order by the command processor/queue **256**. In this manner, the channels may be kept busy. Commands that are received from the host for processing on the same channel may be processed in the order that the commands were received from the host by the command processor/queue **256**. In one exemplary implementation, the command processor/queue **256** may be configured to maintain a list of commands received from the host in an oldest-first sorted list to ensure timely execution of the commands.

**[0043]** The channel controllers **250** may be arranged and configured to process commands from the command processor/queue **256**. Each of the channel controllers **250** may be configured to process commands for multiple flash memory chips **218**. In one exemplary implementation, each of the channel controllers **250** may be configured to process commands for up to and including 32 flash memory chips **218**.

**[0044]** The channel controllers **250** may be configured to process the commands from the command processor/queue **256** in order as designated by the command processor/queue **256**. Examples of the commands that may be processed include, but are not limited to, reading a flash page, programming a flash page, copying a flash page, erasing a flash block, reading a flash block's metadata, mapping a flash memory chip's bad blocks, and resetting a flash memory chip.

**[0045]** The information and configuration interface module **258** may be arranged and configured to interface with a memory module (e.g., memory module **116** of FIG. **1**) to receive configuration information for the FPGA controller **210**. For example, the information and configuration interface module **258** may receive one or more images from the memory module to provide firmware to the FPGA controller **210**. Modifications to the images and to the firmware may be provided by the host to the controller **210** through the information and configuration interface module **258**. Modifications received through the information and configuration interface module **258** may be applied to any of the components of the controller **210** including, for example, the PCIe interface module **208**, the bi-directional direct memory access (DMA) controller **252**, the DRAM controller **254**, the command processor/queue **256** and the channel controllers **250**. The information and configuration interface module **258** may include one or more registers, which may be modified as necessary by instructions from the host.

**[0046]** The FPGA controller **210** may be arranged and configured to cooperate and process commands in conjunction with the host. The FPGA controller **210** may perform or at least assist in performing error correction, bad block management, logical to physical mapping, garbage collection, wear levelling, partitioning and low level formatting related to the flash memory chips **218**.

**[0047]** FIG. **3A** is a schematic block diagram of an apparatus **300** including a data storage device **302** having a plurality of flash memory chips **318a**, **318b**, **318c**, **318d**, **318e**, **318f**, **318g**, **318h**, **318i**, **318j**, **318k**, **318l** that are organized into a first partition **321** and a second partition **322**. The first and second partition **321** and **322** define different physical

areas of storage space in the data storage device 302, such that directories and files of different categories can be stored in the different partitions, or so that one partition can be used for different purposes than the other partition. The first partition can include a first subset of the flash memory chips 318a-f, while the second partition can include a second subset of the flash memory chips 318g-l, where there are not any flash memory chips that are part of both partitions. That is, the boundary between the partitions 321 and 322 is drawn between individual flash memory chips to ensure that an individual flash memory chip does not belong to more than one partition.

**[0048]** Organizing the data storage device into two or more partitions can serve a number of purposes. For example, operating system file stored on one partition can be kept separate from user files stored on another partition. Cache and log files that can change size dynamically and rapidly, potentially making a file system full, can be stored on one partition and kept separate from other files stored on a different partition. Partitions can be used for multi-booting setups, which allow users to have more than one operating system on a single computer. For example, a user could install Linux, Mac OS X, and Microsoft Windows or operating systems on different partitions of the same data storage device and have a choice of booting into any operating system (supported by the hardware) at power-up. Partitions can be used to protect or isolate files to make it easier to recover a corrupted file system or operating system installation. For example if one partition is corrupted but none of the other file systems are affected, the data on the storage device may still be salvageable. Using a separate partition for read-only data also reduces the chances of the file system on that partition becoming corrupted. Partitions also can raise overall computer performance on systems where smaller file systems are more efficient. For example, large hard drives with only one NTFS file system typically have a very large sequentially-accessed Master File Table (MFT), and it generally takes more time to read this MFT than the smaller MFTs of smaller partitions.

**[0049]** In another example embodiment, the data storage device 302 may be used to store large amounts of data (e.g., many Gigabytes or Terabytes of data) that must be read quickly from the data storage device and supplied to the host. For example, the data storage device can be used to cache large volumes of publicly accessible information (e.g., a large corpus of web pages from the World Wide Web, a large library of electronic versions of books, or digital information representing a large volume of telecommunications, etc.) that can be fetched by the host in response to a query. Thus, it can be important that the relevant data be accessed and returned very quickly in response to a read command issued by the host. However, the information stored in the data storage device also may need to be constantly updated to keep the information up to date as the relevant information changes. For example, if the information on the storage device relates to a corpus of web pages, the information stored on the storage device may need to be updated as the web pages change and as new web pages are created.

**[0050]** In such a system, a partitioned flash memory data storage device 302 can offer exceptional performance. In a flash memory storage device, write operations to a flash memory chip take much longer (e.g., 10-100 times longer) than read operations from a flash memory chip.

**[0051]** Therefore, organizing the chips 318a-l of the data storage device into two or more partitions, where the parti-

tions are defined at boundaries between different chips, offers a way to ensure fast read operations while also allowing the information stored on the data storage device to be updated in real time. For example, both partitions 321 and 322 can be used to store a corpus of data (e.g., a corpus of web pages) to be served in response to queries and the individual partitions can alternate between serving the requests and being updated with new information. For instance, in a first time period the first partition 321 can be used to provide the information to the host (e.g., information that may be requested in response to a user query), while the data on the second partition 322 is updated (e.g., in response to changes or additions to the web pages of the corpus). Then, in a second time period, the recently updated second partition 322 can be used to provide the information to the host, while the data on the first partition 321 is updated. This process can be repeated so that data is always served from a partition that acts as a read-only device, and therefore provides very fast responses to read commands from the host without being slowed down by write commands, while the other partition is being updated with new information. Defining the partitions such that an individual flash memory chip is included in only one partition ensures that no flash chip will have data written to it and read from it at substantially the same time, which would cause a delay in responding to a read request from the host 350.

**[0052]** As discussed above, the memory chips 318a-1 can be connected to a controller that may include a FPGA controller 310. The FPGA controller may be configured to operate in the manner described above with respect to controller 110 of FIG. 1 or of FPGA 210 of FIG. 2. The FPGA controller 310 may include multiple channel controllers 312a, 312b, 312c, 312d, 312e, 312f to connect the multiple channels 112 to the flash memory chips 318a-1. Of course, as described above, the storage device can include more than 12 flash memory chips, more than six channel controllers, and many more than two flash memory chips may be operably connected to a channel controller across a physical channel. Thus, the implementation shown in FIGS. 3A and 3B is merely schematic for clarity of illustration.

**[0053]** In one implementation, channel controllers 312a, 312b, 312c, 312d, 312e, 312f can control channels that are operably connected to flash memory chips that are part of each partition 321 and 322. For example, channel controller 312a can be operably connected to memory chip 318a, which is part of the first partition 321, and also to memory chip 318g, which is part of the second partition 322. In such a configuration, at least one memory chip in the first partition 321 is connected to each communication channel between the data storage device 302 and the host, and at least one memory chip in the second partition 322 is connected to each communication channel between the data storage device 302 and the host 350. Such a configuration results in maximum parallelism of communication between a partition 321 or 322 and the host, which can result in fast read access and fast write times from and to the data storage device 302.

**[0054]** In another implementation, approximately half the channel controllers can be operably connected to flash memory chips in a first partition and approximately half the channel controllers can be operably connected to flash memory chips in the second partition.

**[0055]** In another implementation, shown in FIG. 3B, flash memory chips 318a, 318b, 318c, 318d, 318e, 318f, 318g, 318h, 318i, 318j, 318k, 318l can be organized into a first partition 331, a second partition 322, a third partition 333, and

a fourth partition 334, where the different partitions define different physical areas of storage space in the data storage device 302, such that directories and files of different categories can be stored in the different partitions, or so that one partition can be used for different purposes than the other partition. The first partition 331 can include a first subset of the flash memory chips 318a-c. The second partition 332 can include a second subset of the flash memory chips 318d-f. The third partition 333 can include a third subset of the flash memory chips 318g-i. The fourth partition 334 can include a fourth subset of the flash memory chips 318j-l. Among the different partitions 331, 332, 333, and 334 there are not any individual flash memory chips whose physical memory address space is part of two or more partitions. That is, the boundaries between the partitions 331, 332, 333, and 334 are drawn between individual flash memory chips to ensure that an individual flash memory chip does not belong to more than one partition.

[0056] In the system of FIG. 3B, a partitioned flash memory data storage device 302 can offer exceptional performance, e.g., when used to store a corpus of data (e.g., a corpus of web pages) to be served in response to queries, and the individual partitions can alternate between serving the requests and being updated with new information. For instance, in a first time period the first, second, and third partitions 331, 332, and 333 can be used to provide the information to the host (e.g., information that may be requested in response to a user query), while the data on the fourth partition 334 is updated (e.g., in response to changes or additions to the web pages of the corpus). Then, in a second time period, the recently updated fourth partition 334, along with the second and third partitions 332 and 332 can be used to provide the information to the host, while the data on the first partition 331 is updated. Thus, data on each partition can be updated in round robin fashion, while query requests are served by the other partitions. This process can be repeated so that data is always served from partitions that act as read-only devices, and therefore provides very fast responses to read commands from the host without being slowed down by write commands, while the other partition is being updated with new information. Defining four partitions results in redundancy of information stored on the data storage device, so that if a partition, channel, or individual memory chip fails, such that one partition is no longer usable, the remaining three partitions can continue to be used to provide a data storage device in which each of the remaining partitions takes turns being updated while the other remaining partitions serve data requests.

[0057] As described above, the data storage device 302 can be connected to a host 350 through an interface 308, which can be a high speed interface, such as, for example a PCIe interface. The host can include, for example, a processor 352, a first memory 354, a second memory 356, and a partition engine 360. The first memory 354 can include, for example, a non-volatile memory device (e.g., a hard disk) adapted for storing machine-readable, executable code instructions that can be executed by the processor 352. The code instructions stored on the first memory 354 can be loaded into the second memory (e.g., a volatile memory, such as, a random access memory) 356 where they can be executed by the processor 352 to create the memory device detection engine 358 and the partition engine 360. The second memory can include logical blocks of "user space" devoted to user mode applications and logical blocks of "kernel space" 364 devoted to running the lower-level resources that user-level applications must con-

trol to perform their functions. The memory device detection engine 358 and the partition engine 360 can reside in the kernel space 364 of the second memory 356.

[0058] The configuration detection engine 358 can be configured to detect the number of flash memory chips 318 on the data storage device 302, and the partition engine 360 can be configured to define the first partition 321 and the second partition 322 of the data storage device. Thus, the configuration detection engine 358 and the partition engine 360, which run on the host 350, can be used by the host to discover hardware device properties of the data storage device 302 and then to define, via the host, the partitions 321 and 322. In one implementation, the configuration detection engine 358 can issue a query command to the data storage device, and in response to the query command the data storage device can return information to the host about, for example, the number of flash memory chips 318, the size (e.g., as measured in bytes) of each chip, the number of channels in the data storage device, the flash memory chips to which each the channel controller 312a-e is operably connected. Such information can be stored on the EEPROM 116 on the FPGA 310 and/or on the EEPROM 120a of the flash board of the data storage device 302. The configuration detection engine can poll the EEPROM 116 or the EEPROM 120a (e.g., during a boot-up operation of the host 350) to cause the data storage device to return such information to the host 350. In another implementation, the host may poll the flash memory chips 318 to provide the information about, for example, the number of flash memory chips 318, the size (e.g., as measured in bytes) of each chip, the number of channels in the data storage device, the flash memory chips to which each the channel controller 312a-e is operably connected.

[0059] The partition engine 360 can receive the information from the memory device detection engine 358 about the number of flash chips 318, the size of each flash chip, the number of channels and the memory chips to which each channels is operably connected, and, based on this information, the partition engine can define a first partition 321 and second partition 322 in the data storage device 302. The partition engine running on the host 350 can define the first partition to include memory blocks drawn from a first subset of the memory chips 318 and the second partition memory blocks drawn from a second subset of the memory chips 318, where the first subset does not include any individual flash chips of the second subset and the second subset does not include any individual flash chips of the first subset. The partition engine 360 then can map the physical memory block addresses (which may include, for example, a unique channel number, a unique flash memory chip number, and a block address within the flash memory chip) to logical addresses that can be used by application programs running in the user space, such that the user space applications running on the host 350 can read data from the data storage device 302 and write data to the data storage device 302 with reference to the logical space addresses.

[0060] After a partition scheme of multiple partitions has been defined and data has been stored on the flash memory chips of the data storage device 100, the device can store information about the partitioning scheme, e.g., on the memory 116, so that when the device is booted at a later time, it can communicate the partitioning scheme to the host 106 for the host to use. For example, the device may maintain information about the physical configuration of the data storage device, including a number of flash memory chips in the

device and about the partitioning scheme, including which flash memory storage chips and channels are associated with which partitions on the memory 116. Then, when the system including the host 106 and the data storage device 100 is booted, the storage device 100 can communicate this information to the host 106, e.g., in response to a read operation performed by the configuration detection engine 358 of the host 106. The partitioning engine 360 of the host 106 then can define the partitions for the operating system and applications running on the host. For example, the partitioning engine 360 can define a first and second partition based on the information read from the storage device 100, where the first and second partitions do not include any of the same memory chips. The partitioning engine 360 also can allocate a logical to physical memory map for the first and second partitions, so that they user-level application programs can use logical addresses that then are mapped to physical memory addresses of the flash memory chips of the storage device 100.

[0061] The partition engine 360 also can be used to re-define the first partition of the data storage device to include a third subset of the plurality of flash memory chips, where the third subset is different from the first subset, and where the third subset does not include any flash memory chips of the second subset and wherein the second subset does not include any flash memory chips of the third subset. For example, with reference to FIG. 3A and FIG. 3B, a user may decide that the original partition scheme shown in FIG. 3A does not suit his or her needs, and therefore may use the host to re-define the partitions 321 and 322 (e.g., to include more or fewer flash memory chips in the particular partitions) or to add additional partitions to the scheme. In one implementation, the first partition 321 can be re-defined as partitions 331 and 333. Allowing the user to define the partitions through the host rather than forcing the user to accept a partition scheme that is pre-defined by, or pre-loaded in, the controller 310 gives the user flexibility to define partitions as he or she desires and to change the partition scheme when the need arises. In another implementation, the imminent failure of one of the flash memory chips, e.g., 318a, may be detected by the host, and in response to this information, the partition engine may re-define the first partition 321 to exclude the flash memory chip 318a from the partition, i.e., as the originally defined first partition but for the memory chip 318a. Thus, any number of partitions can be defined (up to the number of flash memory chips 118a and 118b in the storage device 100), and different partitions within a partition scheme can include different numbers of flash memory chips and can include different amounts of memory space.

[0062] The host also may include an address assignment engine 366 that can exist in the kernel 364 and that can assign physical memory addresses to data to be written to the data storage device 302. For example, an application running in user space 362 may call for data to be written from the host 350 to the data storage device 302, and the user space application may specify that the data be written to a particular logical memory address. The address assignment engine 366 may translate logical addresses into physical addresses that can include, for example, a particular channel that the data should be written to, a particular flash memory chip operably connected to the specified channel to which the data should be written, and a particular physical block address of the specified memory chip to which the data should be written. In such an implementation, the translation of logical addresses to physical memory space addresses can be performed by the

address assignment engine 366, such that the role of the DRAM controller 254 of the FPGA 210 is reduced or irrelevant.

[0063] FIG. 4 is an exemplary flowchart illustrating an example process 400 of partitioning the data storage device of FIG. 1, where the device includes a plurality of flash memory chips. The process 400 can include determining a number of flash memory chips in the data storage device (402). For example, the configuration detection engine can query the data storage device to gather information about the number of flash memory chips in the data storage device. A first partition of the data storage device can be defined, via a host coupled to the data storage device, where the first partition includes a first subset of the plurality of flash memory chips (404). A second partition of the data storage device can be defined, via the host, where the second partition includes a second subset of the plurality of flash memory chips (406). As a result of this process it is ensured that the first subset does not include any flash memory chips of the second subset and that the second subset does not include any flash memory chips of the first subset.

[0064] Optionally, the process 400 can include writing data to the first partition while reading data from the second partition (408). Determining the number flash memory chips in the data storage device can include transmitting information from the data storage device to the host indicating the number of flash memory chips in the data storage device (410). An address location in the data storage device to which to write data from the host can be defined in the host, where the address location specifies that the data be written to a specific one of the plurality of memory chips (412).

[0065] When the data storage device includes a plurality of physical channels for communication of data between the host and the plurality of flash memory chips, with each channel being operably connected to a different plurality of the memory chips, the process 400 can further include determining the number of physical channels (414), determining a first subset of the channels, where channels of the first subset of the channels are operably connected only to memory chips of the first subset of memory chips (416), determining a second subset of the channels, where channels of the second subset of the channels are operably connected only to memory chips of the second subset of memory chips (418), and defining, in the host, an address location in the data storage device to which to write data from the host, wherein the address location specifies that the data be written to a specific one of the plurality of memory chips through a specific channel (420). In addition, the process 400 can include re-defining, via the host coupled to the data storage device, the first partition of the data storage device to include a third subset of the plurality of flash memory chips (422).

[0066] FIG. 5 is an exemplary block diagram of an example implementation of an apparatus 500 in which a host 551 can control the striping of data across different channels 513, 523, 533, 543 to flash memory chips 514, 515, 516, 517, 524, 525, 526, 527, 534, 535, 536, 537, 544, 545, 546, and 547 of a data storage device 501 and/or across the different flash memory chips. For example, logically sequential data (e.g., data of a file) can be broken up into segments and the segments can be assigned to different channels 513, 523, 533, 543 or to different flash memory chips 514, 515, 516, 517, 524, 525, 526, 527, 534, 535, 536, 537, 544, 545, 546, and 547. By segmenting the data and striping it across different channels and/or flash memory chips, e.g., in a round-robin fashion, different



segments of the logically sequential data can be written to different physical devices (e.g., channels or flash memory chips) concurrently. Because the time required to write data to a flash memory chip is non-trivial compared to the time for a processor 552 of the host 551 to process data packets destined for the storage device 501, striping the data across different channels 513, 523, 533, 543 or to different flash memory chips 514, 515, 516, 517, 524, 525, 526, 527, 534, 535, 536, 537, 544, 545, 546, and 547 can speed the writing of the logically sequential data from the host 551 to the storage device 501. Similarly, reading striped logically sequential data back from the storage device 501, where the data has been striped across different channels or chips, can be faster than reading the data across a single channel or from a single chip. Thus, when reading back striped data, a first segment of logically sequential data can be read back to the host 551 (e.g., from a chip 514) across a first channel 513, while the next segment of the logically sequential data is being fetched from another chip (e.g., chip 524) for transmission across a second channel 514.

[0067] When striping logically sequential data to particular flash memory chips 514, 515, 516, 517, 524, 525, 526, 527, 534, 535, 536, 537, 544, 545, 546, and 547 using “chip striping” the host can specify the destination memory address for a particular segment, where the specified address can include a particular flash memory chip to which the segment is to be written. Thus, the striping of data to particular chips can be placed under the control of the host 551. This can provide a great degree of parallelism when writing data to and reading data from the storage device 501. For example, an implementation of the storage device that includes 12 channels and 20 flash memory chips per channel, a file can be striped across all 240 flash memory chips, which means that 240 write or read operations can be performed when accessing the chips in a round-robin fashion before a chip needs to be accessed a second time. This high degree of parallelism results can result in a high data throughput rate between the host 551 and the storage device 501, such that data can be read from and written to the storage device very quickly.

[0068] When striping logically sequential data across particular channels 513, 523, 533, 543 using “channel striping” the host can specify the destination memory address for a particular segment, where the specified address can include a particular channel to which the segment is to be written. Then, at the time the segment is written to the specified channel the particular flash memory chip operably connected to the specified channel to which the segment is written can be chosen dynamically by the host 551 based on the current run-time state of the chips, e.g., base on chip space availability of the different chips. Channel striping can be more impervious to chip failures than chip striping, because if single chip can fail when using channel striping, the storage device can continue to operate. In addition, channel striping offers advantages over chip striping in terms of write performance, as opposed to read performance, because the optimum chip of all chips operably connected to a specified channel is selected dynamically for writing data. Because of the write performance advantages of channel striping, garbage collection, which involves several write operations, can be performed efficiently when using channel striping.

[0069] Thus, the striping of data to particular chips can be placed under the control of the host 551. This can provide a great degree of parallelism when writing data to and reading data from the storage device 501. For example, an imple-

mentation of the storage device that includes 12 channels and 20 flash memory chips per channel, a file can be striped across all 240 flash memory chips, which means that 240 write or read operations can be performed when accessing the chips in a round-robin fashion before a chip needs to be accessed a second time. This high degree of parallelism results can result in a high data throughput rate between the host 551 and the storage device 501, such that data can be read from and written to the storage device very quickly.

[0070] Similar to the embodiment discussed above in connection with FIG. 3, the flash memory chips 514, 515, 516, 517, 524, 525, 526, 527, 534, 535, 536, 537, 544, 545, 546, and 547 can be connected to a controller that may include a FPGA controller 510. The FPGA controller 510 may be configured to operate in the manner described above with respect to controller 110 of FIG. 1, the FPGA 210 of FIG. 2, or the FPGA 310 of FIG. 3. The FPGA controller 510 may include multiple channel controllers 512, 522, 532, 542 that are operably connected via respective physical channels 513, 523, 533, 543 to respective groups of flash memory chips: 514, 515, 516, and 517; 524, 525, 526, and 527; 534, 535, 536, and 537; and 544, 545, 546, and 547. Of course, as described above, the storage device can include many more than 16 flash memory chips, many more than four channel controllers, and many more than four flash memory chips may be operably connected to a channel controller across a physical channel. Thus, the implementation shown in FIG. 5 is merely schematic for clarity of illustration.

[0071] As described above, the data storage device 501 can be connected to a host 551 through an interface 508, which can be a high speed interface, such as, for example a PCIe interface. The host can include, for example, a processor 552, a first memory 554, a second memory 560. The second memory 560 can include, for example, volatile memory (e.g., random access memory) into which executable instructions are loaded for fast execution by the processor 552. The first memory 454 can include, for example, a non-volatile memory device (e.g., a hard disk) adapted for storing machine-readable, executable code instructions that can be executed by the processor 552. The code instructions stored on the first memory 554 can be loaded into the second memory (e.g., a volatile memory, such as, a random access memory) 560 where they can be executed by the processor 552 to stripe data using “chip striping,” “channel striping” or a combination of both. The second memory can include logical blocks of “user space” 562 devoted to user mode applications and logical blocks of “kernel space” 564 devoted to running the lower-level resources that user-level applications must control to perform their functions. Within the kernel space 564 of the second memory 560 can reside an initialization engine 566 for setting up a striping scheme, a segmentation engine 568 for segmenting logically sequential data into segments, a striping engine 570 for striping the data across distinct physical elements (e.g., channels or chips) of the storage device 501, and an address assignment engine 572 for assigning addresses to the data segments.

[0072] An initialization engine 566 can be configured to determine a first channel chunk size with which to write data to flash memory chips connected to separate channels. In one implementation, the initialization engine can receive and determine the first channel chunk size based on information about the page size of data that is written to the flash memory chips in the storage device 501 and based on information about the number of flash memory chips that are connected to channels



in the storage device **501**. For example, if the storage device includes 12 channels and 20 flash memory chips are connected to each channel, and the page size is 4K, then the initialization engine may determine an appropriate channel chunk size to be some multiple of 4K (e.g., 8K, 32K, 40K, or 80K). The initialization engine **566** can receive this information about the physical configuration of the storage device **501** from a storage medium (e.g., an EEPROM) **520** that stores information about, for example, the number of physical channels **513**, **523**, **533**, **543** in the device **501**, the number of flash memory chips **514**, **515**, **516**, **517**, **524**, **525**, **526**, **527**, **534**, **535**, **536**, **537**, **544**, **545**, **546**, and **547** in the device, the type of flash memory chips (e.g., single-level cell (“SLC”) flash or multilevel cell (“MLC”) flash) in the storage device, and the page size of data written to the chips. The host **550** can transmit a command to the storage device **501** to request the transfer of such information about the physical parameters of the storage device (e.g., the number of channels, number of chips, type of chips, and page size), and in response to the command the storage device **501** can transmit the information back to the host **550**.

**[0073]** When logically sequential data is written to the storage device **501** using a channel striping technique, the logically sequential data can be segmented in channel chunk size units. For example, a segmentation engine **568** can divide logically sequential data (e.g., a data file) into multiple segments whose can be, for example, equal to the channel chunk size determined by the initialization engine **566**. In one implementation, the segmentation engine **568** can receive logically sequential data and can output segments that are sized according to the specified channel chunk sizes. A striping engine **570** then can control the striping of the logically sequential data to different channels of the data storage device **501** in first channel chunk size units. For example, an address assignment engine **572** can assign a memory address to the data segments, where the assigned memory address specifies that the segment be written to a specific one of the physical channels of the storage device **501**.

**[0074]** The striping engine **570** can tag each segment with an address (which may be assigned by the address assignment engine **572**) that will cause the particular segment to be written to a particular channel **513**, **523**, **533**, **543** that is indicated by the address. For example, a first channel chunk of data can be written to channel **513**, a second channel chunk of data can be written to channel **523**, a third channel chunk of data can be written to channel **533**, and a fourth channel chunk of data can be written to channel **543**. When a channel chunk size unit of data addressed to a particular channel (e.g., channel **513**) arrives at a channel controller (e.g., channel controller **512**) associated with the particular channel, then, if channel striping is used and the channel chunk size unit of data is not addressed to a particular flash memory chip connected to the channel, the channel controller can write portions of the channel chunk size unit to different flash memory chips. The individual flash memory chip selected for each portion can be determined dynamically (e.g., by the host **550** or by the controller) based on the current run time state of each chip connected to the channel, e.g., based on the chip space availability of the chips connected to the channel. For example, if a write operation to flash memory chip **514** is still being performed when a channel chunk size unit of data arrives at controller **512**, then the portions of the channel chunk size unit of data may be written to flash memory chips **515**, **516**, and **517** until the write operation to chip **514** is completed.

**[0075]** Thus, by using channel striping when writing logically sequential data from the host to the storage device, data can be written to one channel while data is also being read from another channel. In addition, by using channel striping and dynamically determining the individual flash memory chips to which to write segments of logically sequential data within a particular channel, write performance of the system **500** can be enhanced, because data will be written preferentially to chips that are most ready to accept the data, so the time the host is kept waiting for a chip to be accessible is kept to a minimum. Furthermore, because garbage collection in flash memory is a write-intensive process, channel striping can improve performance of garbage collection.

**[0076]** An advantage of the host **550** controlling the initialization and execution of the data striping is that the host can control and change the parameters that are used to perform data striping, so that the host can setup and control the interaction with the storage device **501**. For example, a user of the host **550** may initially configure the host to use a first channel chunk size for striping data across different channels of the data storage device **501**, but as the user’s desires change, the apparatus **500** is used for a different application, different flash memory chips are used in the storage device, etc., a need may arise for using a different channel chunk size for striping data across the channels. In this case, the initialization engine may be further configured by the user to determine a second channel chunk size, different from the first channel chunk size, with which to write data to flash memory chips connected to separate channels. The segmentation engine can be further configured to segment logically sequential data into second channel chunk size segments, and the striping engine can be further configured to stripe data to different channels of the data storage device in second channel chunk size units.

**[0077]** In addition to determining a channel chunk size with which to stripe logically sequential data across different channels in segments, the initialization engine also can determine a chip chunk size with which to stripe logically sequential data across different chips, where the chip chunk size determines the amount of data to be written to a particular chip before beginning to write data to a different chip. Then, when striping logically sequential data across particular chips (e.g., chips **514**, **515**, **516**, and **517** that are connected to a particular channel **513**) using “chip striping” the host can specify the destination memory address for a particular segment, where the specified address can include a particular chip to which the segment is to be written. With chip striping, logically sequential data can be striped across different chips of the storage device **501** in chip chunk size unit. That is, after a chip chunk size data segment has been written to one flash memory chip the next chip chunk size unit can be written to a different chip. Thus, chip striping provides maximum parallelism in read and write operations from and to the storage device **501**. For example, in a storage device **501** having 12 channels and 20 chips per channel, segments of a data file can be written to 240 different chips before a chip is revisited. Therefore, chip striping offers advantages over channel striping in terms of read performance, because the high degree of parallelism that can be achieved with chip striping.

**[0078]** Thus, with chip striping the initialization engine **566** can be configured to determine a first chip chunk size with which to write data to flash memory chips of the storage device **501**. For example, based on information received from the storage device **501** about the number of flash memory chips in the storage device **501** and the page size used to write

data to the flash memory chips, the initialization engine 566 may determine an appropriate channel chunk size to be some multiple of the page size (e.g., 8K, 32K, 40K, 80K, 160K, 320K, etc.).

[0079] Then, when logically sequential data is written to the storage device 501 using a chip striping technique, the logically sequential data can be segmented in chip chunk size units for writing to the chips. For example, the segmentation engine 568 can divide logically sequential data (e.g., a data file) into multiple segments whose size can be, for example, equal to the chip chunk size determined by the initialization engine 566. In one implementation, the segmentation engine 568 can receive logically sequential data and can output segments that are sized according to the specified chip chunk sizes. A striping engine 570 then can control the striping of the logically sequential data to different chips of the data storage device 501 in chip chunk size units. For example, an address assignment engine 572 can assign a memory address to the data segments, where the assigned memory address specifies that the segment be written to a specific one of the chips of the storage device 501.

[0080] In another implementation, the segmentation engine 568 can receive logically sequential data and can output segments that are sized according to a specified channel chunk size and which are further subdivided into chip chunk size units. The striping engine 570 then can control the striping of the logically sequential data to different channels of the data storage device 501 in channel chunk size units and can control the striping of data to chips connected to the channel in chip chunk size units. For example, the address assignment engine 572 can assign a memory address to the data segments, where the assigned memory address specifies that the segment be written to a specific one of the channels and a specific one of the chips of the storage device 501.

[0081] The striping engine 570 can tag each segment with an address (which may be assigned by the address assignment engine 572) that will cause the particular segment to be written to a particular channel 513, 523, 533, 543 and to a particular flash memory chip 514, 515, 516, 517, 524, 525, 526, 527, 534, 535, 536, 537, 544, 545, 546, and 547 that is indicated by the address. For example, a first channel chunk of data can be written to channel 513, a second channel chunk of data can be written to channel 523, a third channel chunk of data can be written to channel 533, and a fourth channel chunk of data can be written to channel 543, whereas a first chip chunk of data of the first channel chunk can be written to chip 514, a second chip chunk of data of the first channel chunk can be written to chip 515, a third chip chunk of data of the first channel chunk can be written to chip 516, and a fourth chip chunk of data of the first channel chunk can be written to chip 517, and a first chip chunk of data of the second channel chunk can be written to chip 524, a second chip chunk of data of the second channel chunk can be written to chip 525, a third chip chunk of data of the second channel chunk can be written to chip 526, and a fourth chip chunk of data of the second channel chunk can be written to chip 527, etc.

[0082] Thus, by using chip striping when writing logically sequential data from the host to the storage device, data can be written to one chip while data is also being read from another chip. Then, when the logically sequential is read back from the multiple chips of the storage device 501, read operations can be performed in parallel from the different flash memory chips.

[0083] Partitioning and striping can be used in combination. For example, a first partition 104a of the flash memory chips in the storage device can be defined to use channel striping and a second partition 104b of the device can be defined to use chip striping. Thus, the first partition 104a may provide relatively better write performance, redundancy, and fault tolerance due to the use of channel striping techniques to write and read data between the host and the first partition, while the second partition may provide relatively better read performance due to the use of chip striping techniques to write and read data between the host and the second partition.

[0084] FIG. 6 is an exemplary flowchart illustrating example operations of the data storage device of FIG. 5. A process 600 of striping data from a host to a data storage device is shown. The device includes a plurality of flash memory chips, and the data storage device includes a plurality of physical channels for communication of data between the host and the plurality of flash memory chips. Each channel is operably connected to a different plurality of the memory chips. A number of physical channels in the plurality of channels can be determined (602), for example, by the initialization engine 566. A first channel chunk size with which to write data to flash memory chips connected to separate channels can be determined (604), for example, by the initialization engine 566. Logically sequential data can be segmented into first channel chunk size segments by the host (606), for example, by the segmentation engine 568 running on the host 550. Data can be striped to different channels of the data storage device in first channel chunk size units (608), for example, by the striping engine 570 in co-operation with the address assignment engine 570.

[0085] In one implementation, the process 600 may further include determining a chip chunk size with which to write data to different flash memory chips (610), and, for each of the determined physical channels, determining a number of flash memory chips operably connected to the channel (612). Channel chunk size segments can be segmented into chip chunk size segments by the host (614), and data in a channel chunk sized unit can be striped to different flash memory chips connected to a channel in chip chunk size units (616).

[0086] In another implementation, a second channel chunk size with which to write data to flash memory chips connected to separate channels can be determined (618). Logically sequential data can be segmented into second channel chunk size segments (620), and data can be striped to different channels of the data storage device in second channel chunk size units (622).

[0087] Implementations of the various techniques described herein may be implemented in digital electronic circuitry, or in computer hardware, firmware, software, or in combinations of them. Implementations may be implemented as a computer program product, i.e., a computer program tangibly embodied in an information carrier, e.g., in a machine-readable storage device, for execution by, or to control the operation of, data processing apparatus, e.g., a programmable processor, a computer, or multiple computers. A computer program, such as the computer program(s) described above, can be written in any form of programming language, including compiled or interpreted languages, and can be deployed in any form, including as a stand-alone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A computer program can be deployed to be executed on one computer or

on multiple computers at one site or distributed across multiple sites and interconnected by a communication network.

**[0088]** Method steps may be performed by one or more programmable processors executing a computer program to perform functions by operating on input data and generating output. Method steps also may be performed by, and an apparatus may be implemented as, special purpose logic circuitry, e.g., a FPGA or an ASIC (application-specific integrated circuit).

**[0089]** Processors suitable for the execution of a computer program include, by way of example, both general and special purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive instructions and data from a read-only memory or a random access memory or both. Elements of a computer may include at least one processor for executing instructions and one or more memory devices for storing instructions and data. Generally, a computer also may include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto-optical disks, or optical disks. Information carriers suitable for embodying computer program instructions and data include all forms of non-volatile memory, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices; magnetic disks, e.g., internal hard disks or removable disks; magneto-optical disks; and CD-ROM and DVD-ROM disks. The processor and the memory may be supplemented by, or incorporated in special purpose logic circuitry.

**[0090]** To provide for interaction with a user, implementations may be implemented on a computer having a display device, e.g., a cathode ray tube (CRT) or liquid crystal display (LCD) monitor, for displaying information to the user and a keyboard and a pointing device, e.g., a mouse or a trackball, by which the user can provide input to the computer. Other kinds of devices can be used to provide for interaction with a user as well; for example, feedback provided to the user can be any form of sensory feedback, e.g., visual feedback, auditory feedback, or tactile feedback; and input from the user can be received in any form, including acoustic, speech, or tactile input.

**[0091]** Implementations may be implemented in a computing system that includes a back-end component, e.g., as a data server, or that includes a middleware component, e.g., an application server, or that includes a front-end component, e.g., a client computer having a graphical user interface or a Web browser through which a user can interact with an implementation, or any combination of such back-end, middleware, or front-end components. Components may be interconnected by any form or medium of digital data communication, e.g., a communication network. Examples of communication networks include a local area network (LAN) and a wide area network (WAN), e.g., the Internet.

**[0092]** While certain features of the described implementations have been illustrated as described herein, many modifications, substitutions, changes and equivalents will now occur to those skilled in the art. It is, therefore, to be understood that the appended claims are intended to cover all such modifications and changes as fall within the scope of the embodiments.

What is claimed is:

1. A method of striping data from a host to a data storage device that includes a plurality of memory chips and a plurality of physical channels for communication of data

between the host and the plurality of memory chips, each channel being operably connected to a different plurality of the memory chips, the method comprising:

- determining a number of physical channels in the plurality of channels;
  - for each of the determined physical channels, determining a number of memory chips operably connected to the channel;
  - determining a first channel chunk size with which to write data to memory chips connected to separate channels;
  - determining a chip chunk size with which to write data to different memory chips;
  - segmenting, via the host, logically sequential data into first channel chunk size segments;
  - segmenting, via the host, the first channel chunk size segments into chip chunk size segments;
  - striping data to different channels of the data storage device in first channel chunk size units; and
  - striping data in a first channel chunk sized segment to different memory chips connected to a channel in chip chunk size units.
2. The method of claim 1, wherein the logically sequential data consists of a data file.
3. The method of claim 1, further comprising:  
writing data to a first channel while reading data from a second channel.
4. The method of claim 1, wherein determining the number of physical channels in the plurality of channels includes transmitting information from the data storage device to the host indicating the number of channels in the data storage device.
5. The method of claim 1, wherein determining the number of physical channels in the plurality of channels includes reading data stored on the host indicating the number of channels in the data storage device.
6. The method of claim 1, further comprising:  
determining a second channel chunk size with which to write data to memory chips connected to separate channels;  
segmenting, via the host, logically sequential data into second channel chunk size segments;  
striping data to different channels of the data storage device in second channel chunk size units; and  
striping data in a second channel chunk sized segment to different memory chips connected to a channel in chip chunk size units.
7. The method of claim 1, wherein the determining of the first channel chunk sized is based on input from a user entered via the host.
8. A method of striping data from a host to a data storage device that includes a plurality of memory chips and a plurality of physical channels for communication of data between the host and the plurality of memory chips, each channel being operably connected to a different plurality of the memory chips, the method comprising:  
determining a number of physical channels in the plurality of channels;  
determining a first channel chunk size with which to write data to memory chips connected to separate channels;  
segmenting, via the host, logically sequential data into first channel chunk size segments; and  
striping data to different channels of the data storage device in first channel chunk size units.

9. The method of claim 8, wherein the logically sequential data consists of a data file.

10. The method of claim 8, further comprising:  
writing data to a first channel while reading data from a second channel.

11. The method of claim 8, wherein determining the number of physical channels in the plurality of channels includes transmitting information from the data storage device to the host indicating the number of channels in the data storage device.

12. The method of claim 8, wherein determining the number of physical channels in the plurality of channels includes reading data stored on the host indicating the number of channels in the data storage device.

13. The method of claim 8, further comprising:  
determining a second channel chunk size with which to write data to memory chips connected to separate channels;  
segmenting, via the host, logically sequential data into second channel chunk size segments; and  
striping data to different channels of the data storage device in second channel chunk size units.

14. The method of claim 8, wherein the determining of the first channel chunk sized is based on input from a user entered via the host.

15. An apparatus comprising:  
a data storage device including a plurality of memory chips and a plurality of physical channels for communication of data between a host and the plurality of memory chips, each channel being operably connected to a different plurality of the memory chips;  
a host operably coupled to the data storage device via an interface, the host comprising:  
an initialization engine configured to determine a first channel chunk size with which to write data to memory chips connected to separate channels;  
a segmentation engine configured to segment logically sequential data into first channel chunk size segments;  
and

a striping engine configured to stripe data to different channels of the data storage device in first channel chunk size units.

16. The apparatus of claim 15, wherein the data storage device comprises a storage medium configured to store the number of channels and wherein the data storage device is configured to transmit, upon receiving a command from the host, information from the data storage device to the host indicating the number of channels in the data storage device.

17. The apparatus of claim 15, wherein the host further comprises an address assignment engine configured to assign a memory address to data to be written to the data storage device, wherein the assigned memory address specifies that the data be written to a specific one of the channels.

18. The apparatus of claim 15, wherein the logically sequential data consists of a data file.

19. The apparatus of claim 15, wherein the striping engine is configured to write data to a first channel while reading data from a second channel.

20. The apparatus of claim 15,  
wherein the initialization engine is further configured to determine a second channel chunk size, different from the first channel chunk size, with which to write data to memory chips connected to separate channels;  
wherein the segmentation engine is further configured to segment logically sequential data into second channel chunk size segments; and  
wherein the striping engine is further configured to stripe data to different channels of the data storage device in second channel chunk size units.

21. The apparatus of claim 15, further comprising:  
wherein the initialization engine is further configured to determine a chip chunk size with which to write data to different memory chips;  
wherein the segmentation engine is further configured to segment the first channel chunk size segments into chip chunk size segments; and  
wherein the striping engine is further configured to stripe data to different chips connected to a channel in first channel chunk size units.

\* \* \* \* \*