United States Patent Application Publication 20250265153
Kind Code A1
Publication Date August 21, 2025
Inventor(s) NAKATSUKA; Hiroyasu et al.

INFORMATION PROCESSING SYSTEM CONTROLLING MULTIPLE MEMORY SYSTEMS

Abstract

According to one embodiment, an information processing system includes a host and memory systems. A first memory system stores first data in a nonvolatile memory. A second memory system stores second data in a nonvolatile memory. The host transmits first update data to the first memory system and transmits second update data to the second memory system. The first memory system generates first XOR data by performing an XOR operation on at least the first data and the first update data, and transmits the first XOR data to the second memory system. The second memory system generates second XOR data by performing an XOR operation on the second data, the second update data, and the first XOR data, and transmits the second XOR data to a third memory system.

Inventors: **NAKATSUKA; Hiroyasu (Machida Tokyo, JP), NAGAI; Koichi (Ota Tokyo, JP)**

Applicant: **Kioxia Corporation** (Tokyo, JP)

Family ID: **1000008578123**

Assignee: **Kioxia Corporation (Tokyo, JP)**

Appl. No.: **19/200766**

Filed: **May 07, 2025**

Foreign Application Priority Data

JP	2023-007397	Jan. 20, 2023

Related U.S. Application Data

parent US continuation 18207932 20230609 parent-grant-document US 12321233 child US 19200766

Publication Classification

Int. Cl.: **G06F11/10** (20060101); **G06F11/07** (20060101)

U.S. Cl.:

CPC **G06F11/1076** (20130101); **G06F11/0787** (20130101); **G06F11/1088** (20130101);

Background/Summary

CROSS-REFERENCE TO RELATED APPLICATIONS
[0001] This application is based upon and claims the benefit of priority from Japanese Patent Application No. 2023-007397, filed Jan. 20, 2023, the entire contents of which are incorporated herein by reference.
FIELD
[0002] Embodiments described herein relate generally to an information processing system that includes a memory system including a nonvolatile memory.
BACKGROUND
[0003] In recent years, memory systems that include a nonvolatile memory and information processing systems that include a host and multiple memory systems are widely used. As one of such memory systems, a solid state drive (SSD) that includes a NAND flash memory is known. The SSD is used as a main storage for various computing devices.
[0004] Redundant Arrays of Independent (Inexpensive) Disks (RAID) may be used to improve a fault tolerance of an information processing system. The RAID is a technology for improving redundancy of stored data and access performance by using multiple memory systems for storing data. For example, in RAID-5, data to be written and a parity (e.g., error correction code: ECC) for the data are distributed and stored to the memory systems. As a result, for example, even when a memory system which stores a portion of the data fails, the portion of the data stored in the failed memory system can be restored by using the other portions of the data and the parity that are stored in the other memory systems.
[0005] In a case where data stored in the memory systems that are configured as RAID is updated, or, in a case where data stored in a failed memory system is restored and the RAID is reconstructed (rebuilt), a load on a host that is connected to the memory systems may increase. In addition, in such a case, data transfer to a specific memory system may be congested. An insufficient bandwidth of a bus to the specific memory system may degrade performance of the whole of the information processing system.

Description

BRIEF DESCRIPTION OF THE DRAWINGS
[0006] FIG. **1** is a block diagram illustrating an example of a configuration of an information processing system according to a first embodiment.
[0007] FIG. **2** is a block diagram illustrating an example of a configuration of a memory system included in the information processing system according to the first embodiment.
[0008] FIG. **3** is a diagram illustrating an example of a sequential write operation in the information processing system according to the first embodiment.
[0009] FIG. **4** is a diagram illustrating a first update operation in an information processing system

according to a comparative embodiment.
[0010] FIG. **5** is a diagram illustrating a second update operation in the information processing system according to the comparative embodiment.
[0011] FIG. **6** is a diagram illustrating a third update operation in the information processing system according to the comparative embodiment.
[0012] FIG. **7** is a diagram illustrating an example of a fourth update operation in the information processing system according to the first embodiment.
[0013] FIG. **8** is a sequence diagram illustrating a specific example of the fourth update operation in the information processing system according to the first embodiment.
[0014] FIG. **9** is a diagram illustrating a specific example of an operation in a first memory system of the information processing system according to the first embodiment.
[0015] FIG. **10** is a diagram illustrating an example of a fifth update operation in the information processing system according to the first embodiment.
[0016] FIG. **11** is a sequence diagram illustrating a specific example of the fifth update operation in the information processing system according to the first embodiment.
[0017] FIG. **12** is a diagram illustrating an example of a first rebuild operation in an information processing system according to a second embodiment.
[0018] FIG. **13** is a sequence diagram illustrating a specific example of the first rebuild operation in the information processing system according to the second embodiment.
[0019] FIG. **14** is a diagram illustrating an example of a second rebuild operation in the information processing system according to the second embodiment.
[0020] FIG. **15** is a sequence diagram illustrating a specific example of the second rebuild operation in the information processing system according to the second embodiment.
[0021] FIG. **16** is a diagram illustrating a specific example of an operation in a first memory system of the information processing system according to the second embodiment.

DETAILED DESCRIPTION

[0022] In general, according to one embodiment, an information processing system includes a host and a plurality of memory systems. The plurality of memory systems each include a controller and a nonvolatile memory. The plurality of memory systems includes at least a first memory system, a second memory system, and a third memory system. The controller of the first memory system stores first data in a first nonvolatile memory. The controller of the second memory system stores second data in a second nonvolatile memory. The first data and the second data constitute at least a part of an error correction code frame. In a case where the first data and the second data are updated, the host transmits, to the first memory system, first update data updated from the first data, and transmits, to the second memory system, second update data updated from the second data. The controller of the first memory system generates first exclusive-logical-OR data by performing an exclusive-logical-OR operation on at least the first data and the first update data, and transmits the first exclusive-logical-OR data to the second memory system. The controller of the second memory system generates second exclusive-logical-OR data by performing an exclusive-logical-OR operation on the second data, the second update data, and the first exclusive-logical-OR data, and transmits the second exclusive-logical-OR data to the third memory system.
[0023] Various embodiments will be described hereinafter with reference to the accompanying drawings.

First Embodiment

[0024] First, a configuration of an information processing system **1** according to a first embodiment will be described with reference to FIG. **1**. The information processing system **1** includes a host device **2**, multiple memory systems **3**, and a switch **4**.
[0025] The host device **2** may be a storage server that stores a large amount of various data in the memory systems **3**, or may be a server or a personal computer. Hereinafter, the host device **2** is also referred to as a host **2**.

[0026] The memory systems **3** are memory systems that are configured as Redundant Arrays of Independent Disks (RAID). Hereinafter, a case where the memory systems **3** are configured as RAID-5 will be explained. In addition, FIG. **1** illustrates a case where the memory systems **3** are four memory systems **3-1**, **3-2**, **3-3**, and **3-4**. The number of memory systems **3** is, for example, any number of three or more. Hereinafter, one memory system **3** that is not specified among the memory systems **3** is also referred to as a memory system **3**.
[0027] The memory system **3** is a semiconductor storage device configured to write data to a nonvolatile memory such as a NAND flash memory and read data from the nonvolatile memory. The memory system **3** is also referred to as a storage device. The memory system **3** is realized as, for example, a solid state drive (SSD).
[0028] The memory system **3** may be used as a storage of the host **2**. The memory system **3** may be provided inside the host **2** or may be connected to the host **2** via a cable or a network.
[0029] The switch **4** is a device that connects the host **2** and the memory systems **3** to each other. The switch **4** includes a control circuit that controls communication between the host **2** and the memory systems **3**.
[0030] An interface for connecting the host **2** and the memory systems **3** via the switch **4** conforms to standards such as PCI Express™ (PCIe™) and NVM Express™ (NVMe™). Hereinafter, the switch **4** is also referred to as a PCIe switch **4**.
[0031] An example of a configuration of each of the host **2** and the memory system **3** will be described below.
(Configuration Example of Host **2**)
[0032] The host **2** includes, for example, a central processing unit (CPU) **21** and a random access memory (RAM) **22**.
[0033] The CPU **21** is, for example, at least one processor. The CPU **21** controls operations of various components of the host **2**. In addition, the CPU **21** controls communication between the host **2** and the memory system **3**. The CPU **21** transmits various commands to the memory system **3**. The commands transmitted to the memory system **3** include, for example, a read command, a write command, an XOR command, and an XOR/write command. The XOR command is a command for requesting an exclusive logical OR (XOR) operation on two or more pieces of data. The XOR/write command is a command for requesting an XOR operation on two or more pieces of data and a write operation of data. Note that the host **2** may include a control circuit (interface) that controls communication between the host **2** and the memory system **3**. The CPU **21** communicates with the memory system **3** via the control circuit.
[0034] The RAM **22** is a volatile memory. The RAM **22** is realized, for example, as a dynamic random access memory (DRAM) or a static random access memory (SRAM). A storage area of the RAM **22** is allocated, for example, as a buffer area in which data is temporarily stored. The buffer area stores, for example, data to be written to the memory system **3** and data read from the memory system **3**.
(Configuration Example of Memory System **3**)
[0035] FIG. **2** is a block diagram illustrating an example of a configuration of the memory system **3**.
[0036] The memory system **3** includes, for example, a nonvolatile memory **5**, a DRAM **6**, and a controller **7**.
[0037] The nonvolatile memory **5** is, for example, a NAND flash memory. Hereinafter, the nonvolatile memory **5** is referred to as a NAND flash memory **5**.
[0038] The NAND flash memory **5** includes multiple blocks B**0**, B**1**, B**2**, . . . , and Bm−1. Each of the blocks B**0**, B**1**, B**2**, . . . , and Bm−1 includes multiple pages P**0**, . . . , and Pn−1. The blocks each function as the minimum unit of a data erase operation. The block may also be referred to as an erasure block or a physical block. Each of the pages P**0**, . . . , and Pn−1 includes memory cells connected to a single word line. The pages each function as a unit of a data write operation and a

data read operation. Note that the word line may function as a unit of a data write operation and a data read operation.
[0039] The tolerable maximum number of program/erase cycles (maximum number of P/E cycles) for each of the blocks is limited. One P/E cycle of a block includes a data erase operation to erase data stored in all memory cells in the block and a data write operation (also referred to as a data program operation) to write data in each page of the block.
[0040] The DRAM **6** is a volatile memory. The DRAM **6** includes, for example, a storage area of firmware (FW), a cache area for a logical-to-physical address conversion table **31**, and a buffer area for temporarily storing data.
[0041] The FW is a program for controlling an operation of the controller **7**. The FW is loaded from the NAND flash memory **5** to the DRAM **6**, for example.
[0042] The logical-to-physical address conversion table **31** manages mapping between each logical address and each physical address of the NAND flash memory **5**. The logical address is an address used by the host **2** for addressing the memory system **3**. The logical address is, for example, a logical block address (LBA).
[0043] The controller **7** functions as a memory controller configured to control the NAND flash memory **5**.
[0044] The controller **7** may function as a flash translation layer (FTL) configured to execute data management and block management of the NAND flash memory **5**. The data management executed by the FTL includes (1) management of mapping data indicative of a relationship between each logical address and each physical address of the NAND flash memory **5**, and (2) process to hide a difference between data read operations/data write operations in units of page and data erase operations in units of block. The block management includes management of defective blocks, wear leveling, and garbage collection.
[0045] Management of mapping between each logical address and each physical address is executed by using the logical-to-physical address conversion table **31**. The controller **7** uses the logical-to-physical address conversion table **31** to manage the mapping between each logical address and each physical address in a certain management size. A physical address corresponding to a logical address indicates a physical memory location in the NAND flash memory **5** to which data of the logical address is written. The controller **7** manages multiple storage areas that are obtained by logically dividing the storage area of the NAND flash memory **5**, using the logical-to-physical address conversion table **31**. The multiple storage areas correspond to multiple logical addresses, respectively. In other words, each of the storage areas is identified by one logical address. The logical-to-physical address conversion table **31** may be loaded from the NAND flash memory **5** to the DRAM **6** when the memory system **3** is boot up.
[0046] The data write operation into one page is executable only once in a single P/E cycle. Thus, the controller **7** writes updated data corresponding to a logical address not to an original physical memory location in which previous data corresponding to the logical address is stored but to a different physical memory location. Then, the controller **7** updates the logical-to-physical address conversion table **31** to associate the logical address with this different physical memory location rather than the original physical memory location, and to invalidate the previous data. Data to which the logical-to-physical address conversion table **31** refers (that is, data associated with a logical address) will be referred to as valid data. Furthermore, data not associated with any logical address in the logical-to-physical address conversion table **31** will be referred to as invalid data. The valid data is data to possibly be read by the host **2** later. The invalid data is data not to be read by the host **2** anymore.
[0047] The controller **7** includes, for example, a host interface (host I/F) **11**, a NAND interface (NAND I/F) **12**, a DRAM interface (DRAM I/F) **13**, and a CPU **14**. The host I/F **11**, the NAND I/F **12**, the DRAM I/F **13**, and the CPU **14** are connected via, for example, a bus **10**.
[0048] The host I/F **11** functions as a circuit that receives various commands and data from the host

2 via the PCIe switch **4**. In addition, the host I/F **11** functions as a circuit that transmits responses to commands and data, to the host **2** via the PCIe switch **4**. Further, the host I/F **11** may function as a circuit that receives various commands, data, and responses to commands, from another memory system **3** via the PCIe switch **4**. In addition, the host I/F **11** may function as a circuit that transmits commands, data, and responses to commands, to another memory system **3** via the PCIe switch **4**.
[0049] The NAND I/F **12** electrically connects the controller **7** and the NAND flash memory **5**. The NAND I/F **12** conforms to an interface standard such as a toggle double data rate (DDR) and an open NAND flash Interface (ONFI).
[0050] The NAND I/F **12** functions as a NAND control circuit configured to control the NAND flash memory **5**. The NAND I/F **12** may be connected to memory chips in the NAND flash memory **5** via multiple channels. By operating the memory chips in parallel, it is possible to broaden an access bandwidth between the controller **7** and the NAND flash memory **5**.
[0051] The DRAM I/F **13** functions as a DRAM control circuit configured to control access to the DRAM **6**.
[0052] The CPU **14** is a processor configured to control the host I/F **11**, the NAND I/F **12**, and the DRAM I/F **13**. The CPU **14** performs various processes by executing the FW loaded from the NAND flash memory **5** onto the DRAM **6**. The FW is a control program including instructions for causing the CPU **14** to execute various processes. The CPU **14** may perform command processes to process various commands from the host **2**. The operation of the CPU **14** is controlled by the FW executed by the CPU **14**.
[0053] The function of each unit in the controller **7** may be realized by dedicated hardware in the controller **7** or may be realized by the CPU **14** executing the FW.
[0054] The CPU **14** functions as, for example, a command reception module **141**, a read processing module **142**, an XOR processing module **143**, and a write processing module **144**. The CPU **14** functions as these modules, for example, by executing the FW.
[0055] The command reception module **141** receives a command transmitted from the host **2** or another memory system **3**. The command reception module **141** controls the read processing module **142**, the XOR processing module **143**, and the write processing module **144** on the basis of the received command. Specifically, the command reception module **141** instructs the read processing module **142** to read data from the NAND flash memory **5**. The command reception module **141** instructs the XOR processing module **143** to perform an XOR operation on two or more pieces of data. The command reception module **141** instructs the write processing module **144** to write data into the NAND flash memory **5**. The command reception module **141** transmits a response to a command, to the host **2** or the memory system **3** that has transmitted the command. In addition, the command reception module **141** may transmit a command and data to another memory system **3**.
[0056] In response to the instruction from the command reception module **141**, the read processing module **142** reads data from the NAND flash memory **5**.
[0057] In response to the instruction from the command reception module **141**, the XOR processing module **143** performs an XOR operation on two or more pieces of data.
[0058] In response to the instruction from the command reception module **141**, the write processing module **144** writes data into the NAND flash memory **5**.
[0059] An example of a specific operation by the command reception module **141**, the read processing module **142**, the XOR processing module **143**, and the write processing module **144** will be described later with reference to FIG. **9**.
[0060] Here, a sequential write operation in the information processing system **1** will be described.
[0061] FIG. **3** illustrates an example of a sequential write operation in the information processing system **1**. The sequential write operation is an operation in which the host **2** continuously writes data to all the four memory systems **3-1**, **3-2**, **3-3**, and **3-4**.
[0062] The DRAM **22** of the host **2** stores user data **51**D to be written to the memory systems **3**.

[0063] When an amount of the user data **51**D to be written to the memory systems **3** has reached a specific unit, the CPU **21** of the host **2** generates an error correction code (ECC) for the user data **51**D to be written. The specific unit corresponds to, for example, a total amount of a plurality pieces of data each capable of being written in one data write operation in each of three memory systems **3**. The amount of data that is capable of being written in one data write operation in one memory system **3** is also referred to as a write unit. The ECC is data for correcting user data in which an error has occurred. The ECC is, for example, an XOR parity. Hereinafter, a case where the ECC is the XOR parity will be explained. In addition, the XOR parity is simply referred to as a parity.
[0064] Specifically, the CPU **21** acquires three pieces of user data **511**, **512**, and **513** of the write units (that is, first user data **511**, second user data **512**, and third user data **513**) that are obtained by dividing the user data **51**D of the specific unit. Then, the CPU **21** performs an XOR operation on the three pieces of user data **511**, **512**, and **513**, thereby generating a parity **51**P. The CPU **21** stores the generated parity **51**P in the DRAM **22**, for example. Each of the three pieces of user data **511**, **512**, and **513** and the parity **51**P have the same data length. The three pieces of user data **511**, **512**, and **513** and the parity **51**P constitute one ECC frame **51**E. The ECC frame is a data unit including a parity and user data that is protected by the parity. That is, the three pieces of user data **511**, **512**, and **513** are protected by the parity **51**P. The parity **51**P is updated in response to at least one of the three pieces of user data **511**, **512**, and **513** being updated.
[0065] Next, the CPU **21** writes the three pieces of user data **511**, **512**, and **513** and the parity **51**P to the four memory systems **3-1**, **3-2**, **3-3**, and **3-4**, respectively, via the PCIe switch **4**.
[0066] Specifically, for example, the CPU **21** transmits a write command for requesting the writing of the first user data **511**, to the first memory system **3-1** ((**1**) in FIG. **3**). The CPU **21** transmits a write command for requesting the writing of the second user data **512**, to the second memory system **3-2** ((**2**) in FIG. **3**). The CPU **21** transmits a write command for requesting the writing of the third user data **513**, to the third memory system **3-3** ((**3**) in FIG. **3**). The CPU **21** transmits a write command for requesting the writing of the parity **51**P, to the fourth memory system **3-4** ((**4**) in FIG. **3**). Note that one of the four memory systems **3-1**, **3-2**, **3-3**, and **3-4** to which each of the first user data **511**, the second user data **512**, the third user data **513**, and the parity **51**P is to be written is determined by the CPU **21** according to a specific rule, for example. The write destinations of the first user data **511**, the second user data **512**, the third user data **513**, and the parity **51**P are not limited to the above-described examples. For example, the CPU **21** may transmit the write commands such that the first user data **511** is written to the second memory system **3-2**, the second user data **512** is written to the third memory system **3-3**, the third user data **513** is written to the fourth memory system **3-4**, and the parity **51**P is written to the first memory system **3-1**.
[0067] In response to receiving the write command from the host **2**, the first memory system **3-1** receives the first user data **511** stored in the DRAM **22** of the host **2** via the PCIe switch **4**. Then, the first memory system **3-1** writes the first user data **511** into the NAND flash memory **5** of the first memory system **3-1**.
[0068] In response to receiving the write command from the host **2**, the second memory system **3-2** receives the second user data **512** stored in the DRAM **22** of the host **2** via the PCIe switch **4**. Then, the second memory system **3-2** writes the second user data **512** into the NAND flash memory **5** of the second memory system **3-2**.
[0069] In response to receiving the write command from the host **2**, the third memory system **3-3** receives the third user data **513** stored in the DRAM **22** of the host **2** via the PCIe switch **4**. Then, the third memory system **3-3** writes the third user data **513** into the NAND flash memory **5** of the third memory system **3-3**.
[0070] In response to receiving the write command from the host **2**, the fourth memory system **3-4** receives the parity **51**P stored in the DRAM **22** of the host **2** via the PCIe switch **4**. Then, the fourth memory system **3-4** writes the parity **51**P into the NAND flash memory **5** of the fourth memory

system **3-4**.
[0071] By the sequential write operation described above, in the information processing system **1**, the first user data **511**, the second user data **512**, the third user data **513**, and the parity **51**P, which constitute one ECC frame **51**E, are written in the four memory systems **3-1**, **3-2**, **3-3**, and **3-4** in a distributed manner. As a result, for example, even when any one of the four memory systems **3-1**, **3-2**, **3-3**, and **3-4** fails, the data stored in the failed memory system **3** can be restored by using the data stored in the other memory systems **3**.
[0072] Note that in the sequential write operation, four input/outputs (I/Os) are performed between the host **2** and the memory systems **3**. The four I/Os are I/Os related to four write operations from the host **2** to the memory systems **3** (i.e., write I/Os). The number of times of input/output is a criterion related to use of a bus bandwidth by communication between the host **2** and the memory systems **3** via the PCIe switch **4**. In addition, the parity **51**P is generated by using resources of the host **2** (more specifically, the CPU **21** and the DRAM **22**).
[0073] Next, an operation for updating a portion of the user data **51**D written in the memory systems **3** will be described. The operation for updating a portion of the user data **51**D written in the memory systems **3** is referred to as an update operation. The update operation includes a random write operation.
[0074] First, an update operation in an information processing system **1**A according to a comparative embodiment will be described with reference to FIGS. **4** to **6**. The information processing system **1**A according to the comparative embodiment includes a host **2**A, multiple memory systems **3**A, and a PCIe switch **4**A. The host **2**A and the memory systems **3**A are capable of communicating with each other via the PCIe switch **4**A. The host **2**A includes a CPU **21**A and a DRAM **22**A. Each of the memory systems **3**A includes a NAND flash memory. The memory systems **3**A are configured as RAID-5. The memory systems **3**A include four memory systems **3-1**A, **3-2**A, **3-3**A, and **3-4**A. Here, it is assumed that the first user data **511**, the second user data **512**, the third user data **513**, and the parity **51**P are stored respectively in the four memory systems **3-1**A, **3-2**A, **3-3**A, and **3-4**A by the same operation as the sequential write operation described above with reference to FIG. **3**.
[0075] FIG. **4** illustrates a first update operation in the information processing system **1**A according to the comparative embodiment. The first update operation is an operation for updating the first user data **511**, which is stored in the first memory system **3-1**A, with first update user data **511**-U.
[0076] The DRAM **22**A of the host **2**A stores the first update user data **511**-U.
[0077] The CPU **21**A of the host **2**A transmits a read command for requesting the reading of the first user data **511** to the first memory system **3-1**A ((**1**) in FIG. **4**).
[0078] In response to receiving the read command from the host **2**A, the first memory system **3-1**A reads the first user data **511** from the NAND flash memory of the first memory system **3-1**A. Then, the first memory system **3-1**A transmits the first user data **511** to the host **2**A.
[0079] The CPU **21**A of the host **2**A stores the first user data **511** that has been received from the first memory system **3-1**A, in the DRAM **22**A. The CPU **21**A performs an XOR operation on the first user data **511** and the first update user data **511**-U, thereby generating first XOR data **521**. The first XOR data **521** is stored in the DRAM **22**A, for example. Then, the CPU **21**A transmits a write command for requesting the writing of the first update user data **511**-U, to the first memory system **3-1**A ((**2**) in FIG. **4**).
[0080] In response to receiving the write command from the host **2**A, the first memory system **3-1**A receives the first update user data **511**-U stored in the DRAM **22**A of the host **2**A. Then, the first memory system **3-1**A writes the first update user data **511**-U into the NAND flash memory of the first memory system **3-1**A.
[0081] Next, the CPU **21**A of the host **2**A transmits a read command for requesting the reading of the parity **51**P to the fourth memory system **3-4**A ((**3**) in FIG. **4**).
[0082] In response to receiving the read command from the host **2**A, the fourth memory system **3-**

4A reads the parity **51**P from the NAND flash memory of the fourth memory system **3-4**A. Then, the fourth memory system **3-4**A transmits the parity **51**P to the host **2**A.

[0083] The CPU **21**A of the host **2**A stores the parity **51**P that has been received from the fourth memory system **3-4**A, in the DRAM **22**A. The CPU **21**A performs an XOR operation on the first XOR data **521** and the parity **51**P that are stored in the DRAM **22**A, thereby generating an update parity **51**P-U. The update parity **51**P-U is a parity changed according to the update from the first user data **511** to the first update user data **511**-U. That is, the first update user data **511**-U, the second user data **512**, the third user data **513**, and the update parity **51**P-U constitute an ECC frame. The CPU **21**A transmits a write command for requesting the writing of the update parity **51**P-U, to the fourth memory system **3-4**A ((**4**) in FIG. **4**).

[0084] In response to receiving the write command from the host **2**A, the fourth memory system **3-4**A receives the update parity **51**P-U stored in the DRAM **22**A of the host **2**A. Then, the fourth memory system **3-4**A writes the update parity **51**P-U into the NAND flash memory of the fourth memory system **3-4**A.

[0085] By the first update operation described above, in the information processing system **1**A, the first user data **511** stored in the first memory system **3-1**A is updated with the first update user data **511**-U, and the parity **51**P stored in the fourth memory system **3-4**A is updated with the update parity **51**P-U.

[0086] In the first update operation, four I/Os from the host **2**A to the memory systems **3**A are performed. The four I/Os include two write I/Os and I/Os related to two read operations (i.e., two read I/Os) from the host **2**A to the memory systems **3**A. In addition, the update parity **51**P-U is generated by using resources of the host **2**A (more specifically, the CPU **21**A and the DRAM **22**A).

[0087] FIG. **5** illustrates a second update operation in the information processing system **1**A according to the comparative embodiment. The second update operation is an operation for updating the first user data **511**, which is stored in the first memory system **3-1**A, with the first update user data **511**-U while reducing the resources of the host **2**A being used.

[0088] The DRAM **22**A of the host **2**A stores the first update user data **511**-U.

[0089] The CPU **21**A of the host **2**A transmits a command for requesting an XOR operation and the writing of the first update user data **511**-U (i.e., XOR/write command), to the first memory system **3-1**A ((**1**) in FIG. **5**).

[0090] In response to receiving the XOR/write command from the host **2**A, the first memory system **3-1**A receives the first update user data **511**-U from the host **2**A. In addition, the first memory system **3-1**A reads the first user data **511** from the NAND flash memory of the first memory system **3-1**A. The first memory system **3-1**A performs an XOR operation on the first user data **511** and the first update user data **511**-U, thereby generating the first XOR data **521**. The first XOR data **521** is stored in an internal buffer **61-1**A of the first memory system **3-1**A. Then, the first memory system **3-1**A writes the first update user data **511**-U into the NAND flash memory of the first memory system **3-1**A.

[0091] Next, the CPU **21**A of the host **2**A transmits an XOR/write command for requesting an XOR operation and the writing of the update parity **51**P-U, to the fourth memory system **3-4**A ((**2**) in FIG. **5**).

[0092] In response to receiving the XOR/write command from the host **2**A, the fourth memory system **3-4**A receives the first XOR data **521** from the first memory system **3-1**A. The fourth memory system **3-4**A reads the parity **51**P from the NAND flash memory of the fourth memory system **3-4**A. The fourth memory system **3-4**A performs an XOR operation on the first XOR data **521** and the parity **51**P, thereby generating the update parity **51**P-U. Then, the fourth memory system **3-4**A writes the update parity **51**P-U into the NAND flash memory of the fourth memory system **3-4**A.

[0093] By the second update operation described above, in the information processing system **1**A, the first user data **511** stored in the first memory system **3-1**A is updated with the first update user

data **511**-U, and the parity **51**P stored in the fourth memory system **3-4**A is updated with the update parity **51**P-U.
[0094] In the second update operation, two I/Os from the host **2**A to the memory systems **3**A are performed. The two I/Os are two write I/Os from the host **2**A to the memory systems **3**A. In the second update operation, any read I/O does not occur. Therefore, in the second update operation, the number of times of read I/Os is reduced as compared with the first update operation described above with reference to FIG. **4**. In addition, the update parity **51**P-U is generated by using the resources of the memory systems **3**A. Therefore, in the second update operation, the load on the host **2**A is reduced. In other words, the offloading of the update operation from the host **2**A to the memory systems **3**A can be realized.
[0095] FIG. **6** illustrates a third update operation in the information processing system **1**A according to the comparative embodiment. The third update operation is an operation for updating the first user data **511** stored in the first memory system **3-1**A with the first update user data **511**-U, and updating the third user data **513** stored in the third memory system **3-3**A with third update user data **513**-U, while reducing the resources of the host **2**A being used. That is, in the third update operation, the two pieces of user data **511** and **513** that are stored in the two memory systems **3-1**A and **3-3**A, respectively, are updated.
[0096] The DRAM **22**A of the host **2**A stores the first update user data **511**-U and the third update user data **513**-U.
[0097] The CPU **21**A of the host **2**A transmits an XOR/write command (first XOR/write command) for requesting an XOR operation and the writing of the first update user data **511**-U, to the first memory system **3-1**A ((**1-1**) in FIG. **6**). In addition, the CPU **21**A transmits an XOR/write command (second XOR/write command) for requesting an XOR operation and the writing of the third update user data **513**-U, to the third memory system **3-3**A ((**1-2**) in FIG. **6**).
[0098] In response to receiving the first XOR/write command from the host **2**A, the first memory system **3-1**A receives the first update user data **511**-U from the host **2**A. The first memory system **3-1**A reads the first user data **511** from the NAND flash memory of the first memory system **3-1**A. The first memory system **3-1**A performs an XOR operation on the first user data **511** and the first update user data **511**-U, thereby generating the first XOR data **521**. The first XOR data **521** is stored in the internal buffer **61-1**A of the first memory system **3-1**A. Then, the first memory system **3-1**A writes the first update user data **511**-U into the NAND flash memory of the first memory system **3-1**A.
[0099] In addition, in response to receiving the second XOR/write command from the host **2**A, the third memory system **3-3**A receives the third update user data **513**-U from the host **2**A. The third memory system **3-3**A reads the third user data **513** from the NAND flash memory of the third memory system **3-3**A. The third memory system **3-3**A performs an XOR operation on the third user data **513** and the third update user data **513**-U, thereby generating second XOR data **522**. The second XOR data **522** is stored in an internal buffer **61-3**A of the third memory system **3-3**A. Then, the third memory system **3-3**A writes the third update user data **513**-U into the NAND flash memory of the third memory system **3-3**A.
[0100] Next, the CPU **21**A of the host **2**A transmits an XOR/write command (third XOR/write command) for requesting an XOR operation and the writing of preliminary XOR data **51**P-P (described later), to the fourth memory system **3-4**A ((**2-1**) in FIG. **6**). Further, the CPU **21**A of the host **2**A transmits an XOR/write command (fourth XOR/write command) for requesting an XOR operation and the writing of the update parity **51**P-U, to the fourth memory system **3-4**A ((**2-2**) in FIG. **6**). The host **2**A may transmit the fourth XOR/write command to the fourth memory system **3-4**A before receiving a response to the third XOR/write command.
[0101] In response to receiving the third XOR/write command from the host **2**A, the fourth memory system **3-4**A receives the first XOR data **521** from the first memory system **3-1**A. The fourth memory system **3-4**A reads the parity **51**P from the NAND flash memory of the fourth

memory system **3-4**A. The fourth memory system **3-4**A performs an XOR operation on the first XOR data **521** and the parity **51**P, thereby generating the preliminary XOR data **51**P-P. Then, the fourth memory system **3-4**A writes the preliminary XOR data **51**P-P into the NAND flash memory of the fourth memory system **3-4**A.

[0102] In response to receiving the fourth XOR/write command from the host **2**A, the fourth memory system **3-4**A receives the second XOR data **522** from the third memory system **3-3**A. The fourth memory system **3-4**A reads the preliminary XOR data **51**P-P from the NAND flash memory of the fourth memory system **3-4**A. The fourth memory system **3-4**A performs an XOR operation on the preliminary XOR data **51**P-P and the second XOR data **522**, thereby generating the update parity **51**P-U. Then, the fourth memory system **3-4**A writes the update parity **51**P-U into the NAND flash memory of the fourth memory system **3-4**A.

[0103] Note that when having received the fourth XOR/write command prior to the third XOR/write command, the fourth memory system **3-4**A performs an XOR operation on the second XOR data **522** and the parity **51**P, thereby generating the preliminary XOR data **51**P-P. In this case, in response to receiving the third XOR/write command, the fourth memory system **3-4**A performs an XOR operation on the preliminary XOR data **51**P-P and the first XOR data **521**, thereby generating the update parity **51**P-U.

[0104] By the third update operation described above, in the information processing system **1**A, the first user data **511** stored in the first memory system **3-1**A is updated with the first update user data **511**-U, the third user data **513** stored in the third memory system **3-3**A is updated with the third update user data **513**-U, and the parity **51**P stored in the fourth memory system **3-4**A is updated with the update parity **51**P-U.

[0105] In the third update operation, similarly to the second update operation described above with reference to FIG. **5**, the update parity **51**P-U is generated by using the resources of the memory systems **3**A. Therefore, in the third update operation, the offloading of the update operation from the host **2**A to the memory systems **3**A can be realized.

[0106] However, in the third update operation, the first XOR data **521** is transmitted from the first memory system **3-1**A to the fourth memory system **3-4**A, and the second XOR data **522** is transmitted from the third memory system **3-3**A to the fourth memory system **3-4**A. A transmission request of the first XOR data **521** to the fourth memory system **3-4**A and a transmission request of the second XOR data **522** to the fourth memory system **3-4**A may be generated simultaneously. In this case, one of the transmission requests is made to wait until transmission for the other of the transmission requests is completed. That is, data transfer to the fourth memory system **3-4**A that stores the parity **51**P may be congested. In addition, a bus connecting each memory system **3**A and the PCIe switch **4**A has a bus bandwidth smaller than that of a bus connecting the host **2** and the PCIe switch **4**. Therefore, an insufficient bus bandwidth for transferring data to the fourth memory system **3-4**A may degrade performance of the whole of the information processing system **1**A.

[0107] In contrast, in the information processing system **1** according to the first embodiment, while the offloading of the update operation from the host **2** to the memory systems **3** can be realized, the congestion of data transfer to the memory system **3** that stores the parity **51**P is also prevented. As a result, in the information processing system **1**, performance of the whole of the information processing system **1** can be improved while reducing the load on the host **2**.

[0108] Two examples of the update operation in the information processing system **1** will be described with reference to FIGS. **7** to **11**.

[0109] FIG. **7** illustrates an example of a fourth update operation in the information processing system **1**. The fourth update operation is an operation for updating user data stored in each of two or more memory systems **3** while reducing the resources of the host **2** being used. Here, a case where the fourth update operation is executed for updating the first user data **511** stored in the first memory system **3-1** with the first update user data **511**-U and updating the third user data **513** stored in the third memory system **3-3** with the third update user data **513**-U will be explained.

Note that it is assumed that the four memory systems **3-1**, **3-2**, **3-3**, and **3-4** store the first user data **511**, the second user data **512**, the third user data **513**, and the parity **51**P, respectively, by the sequential write operation described above with reference to FIG. **3**. The first user data **511**, the second user data **512**, the third user data **513**, and the parity **51**P constitute one ECC frame **51**E.
[0110] The DRAM **22** of the host **2** stores the first update user data **511**-U and the third update user data **513**-U.
[0111] The CPU **21** of the host **2** transmits an XOR/write command (first XOR/write command) for requesting an XOR operation and the writing of the first update user data **511**-U, to the first memory system **3-1** ((**1**) in FIG. **7**). The first XOR/write command may include a logical address of the first user data **511**.
[0112] In response to receiving the first XOR/write command from the host **2**, the first memory system **3-1** receives the first update user data **511**-U from the host **2**. The first memory system **3-1** reads the first user data **511** from the NAND flash memory **5** of the first memory system **3-1**. The first memory system **3-1** performs an XOR operation on the first user data **511** and the first update user data **511**-U, thereby generating the first XOR data **521**. The first XOR data **521** is stored in an internal buffer **61-1** of the first memory system **3-1**. For example, a part of the storage area of the DRAM **6** is allocated as the internal buffer **61-1**. The internal buffer **61-1** is, for example, a controller memory buffer (CMB) that is capable of being accessed by the other memory systems **3** via the PCIe switch **4**. Since the internal buffer **61-1** is used to store the first XOR data **521**, a write amplification factor (WAF) does not increase. The WAF is a value obtained by dividing the amount of data actually written into the NAND flash memory **5** by the amount of data written into the NAND flash memory **5** in accordance with requests from the host **2**. The first memory system **3-1** writes the first update user data **511**-U into the NAND flash memory **5** of the first memory system **3-1**.
[0113] Next, the CPU **21** of the host **2** transmits an XOR/write command (second XOR/write command) for requesting an XOR operation and the writing of the third update user data **513**-U, to the third memory system **3-3** ((**2**) in FIG. **7**). The second XOR/write command may include a logical address of the third user data **513** and an identifier (for example, an address) of the internal buffer **61-1** of the first memory system **3-1** (more specifically, the storage location of the first XOR data **521**).
[0114] In response to receiving the second XOR/write command from the host **2**, the third memory system **3-3** receives the third update user data **513**-U from the host **2** and receives the first XOR data **521** from the first memory system **3-1**. In addition, the third memory system **3-3** reads the third user data **513** from the NAND flash memory **5** of the third memory system **3-3**. The third memory system **3-3**A performs an XOR operation on the third user data **513**, the third update user data **513**-U, and the first XOR data **521**, thereby generating third XOR data **523**. The third XOR data **523** is stored in an internal buffer **61-3** of the third memory system **3-3**. Then, the third memory system **3-3** writes the third update user data **513**-U into the NAND flash memory **5** of the third memory system **3-3**.
[0115] Next, the CPU **21** of the host **2** transmits an XOR/write command (third XOR/write command) for requesting an XOR operation and the writing of the update parity **51**P-U, to the fourth memory system **3-4** ((**3**) in FIG. **7**). The third XOR/write command may include a logical address of the parity **51**P and an identifier (for example, an address) of the internal buffer **61-3** of the third memory system **3-3** (more specifically, the storage location of the third XOR data **523**).
[0116] In response to receiving the third XOR/write command from the host **2**, the fourth memory system **3-4** receives the third XOR data **523** from the third memory system **3-3**. The fourth memory system **3-4** reads the parity **51**P from the NAND flash memory **5** of the fourth memory system **3-4**. The fourth memory system **3-4** performs an XOR operation on the third XOR data **523** and the parity **51**P, thereby generating the update parity **51**P-U. Then, the fourth memory system **3-4** writes the update parity **51**P-U into the NAND flash memory **5** of the fourth memory system **3-4**.

[0117] By the fourth update operation described above, in the information processing system **1**, the first user data **511** stored in the first memory system **3-1** is updated with the first update user data **511**-U, the third user data **513** stored in the third memory system **3-3** is updated with the third update user data **513**-U, and the parity **51**P stored in the fourth memory system **3-4** is updated with the update parity **51**P-U.
[0118] In the fourth update operation, the update parity **51**P-U is generated by using the resources of the memory systems **3**. Therefore, in the fourth update operation, the offloading of the update operation from the host **2** to the memory systems **3** can be realized.
[0119] Further, in the fourth update operation, the first XOR data **521** is transmitted from the first memory system **3-1** to the third memory system **3-3**, and the third XOR data **523** is transmitted from the third memory system **3-3** to the fourth memory system **3-4**. As a result, it is possible to prevent the congestion of data transfer to the fourth memory system **3-4** storing the parity **51**P.
[0120] Therefore, in the information processing system **1** that performs the fourth update operation, performance of the whole of the information processing system **1** can be improved while reducing the load on the host **2**.
[0121] FIG. **8** is a sequence diagram illustrating a specific example of the fourth update operation in the information processing system **1**.
[0122] First, the host **2** transmits the first XOR/write command and the first update user data **511**-U to the first memory system **3-1** (A**1**). The host **2** may transmit the first update user data **511**-U to the first memory system **3-1** that has received the first XOR/write command.
[0123] In response to receiving the first XOR/write command and the first update user data **511**-U, the first memory system **3-1** reads the first user data **511** from the NAND flash memory **5** of the first memory system **3-1** (A**2**). The first memory system **3-1** performs an XOR operation on the first user data **511** and the first update user data **511**-U, thereby generating the first XOR data **521** (A**3**). The first memory system **3-1** writes the first update user data **511**-U into the NAND flash memory **5** of the first memory system **3-1**, and invalidates the first user data **511** (A**4**). Specifically, the first memory system **3-1** updates the logical-to-physical address conversion table **31** so as to associate the logical address corresponding to the first user data **511** with the physical memory location in which the first update user data **511**-U is stored, instead of the physical memory location in which the first user data **511** is stored. As a result, the first user data **511** is invalidated.
[0124] Then, the first memory system **3-1** transmits a response to the first XOR/write command to the host **2** (A**5**). Note that the first memory system **3-1** may transmit the response to the first XOR/write command to the host **2** not only when the first update user data **511**-U has been written in the NAND flash memory **5** but also when it is guaranteed that the first update user data **511**-U is made non-volatile. Specifically, for example, in a case where the first memory system **3-1** has a power loss protection (PLP) function, the first memory system **3-1** may transmit the response to the host **2** in response to generating the first XOR data **521** and storing the first update user data **511**-U in a write buffer (for example, the DRAM **6**). As a result, the first memory system **3-1** can transmit the response to the host **2** earlier than a case of transmitting the response in response to writing the first update user data **511**-U into the NAND flash memory **5**. The PLP function is a function for writing, into the NAND flash memory **5**, user data or the like that is stored in the write buffer and has not yet been written into the NAND flash memory **5** by using energy of charges stored in a power storage device of the memory system **3** when power supply from an external power supply to the memory system **3** is lost.
[0125] Next, in response to receiving the response to the first XOR/write command, the host **2** transmits the second XOR/write command and the third update user data **513**-U to the third memory system **3-3** (A**6**).
[0126] In response to receiving the second XOR/write command and the third update user data **513**-U, the third memory system **3-3** receives the first XOR data **521** from the first memory system **3-1** (A**7**). Specifically, the third memory system **3-3** receives, for example, the first XOR data **521**

that has been read and transmitted from a specific storage area (for example, the internal buffer **61-1**) in the first memory system **3-1**. The location in the first memory system **3-1** where the first XOR data **521** is stored is defined in advance, for example. The third memory system **3-3** reads the third user data **513** from the NAND flash memory **5** of the third memory system **3-3** (A**8**). The third memory system **3-3** performs an XOR operation on the third user data **513**, the third update user data **513**-U, and the first XOR data **521**, thereby generating the third XOR data **523** (A**9**). The third memory system **3-3** writes the third update user data **513**-U into the NAND flash memory **5** of the third memory system **3-3**, and invalidates the third user data **513** (A**10**). Then, the third memory system **3-3** transmits a response to the second XOR/write command to the host **2** (A**11**). Note that the third memory system **3-3** may transmit the response to the second XOR/write command to the host **2** when it is guaranteed that the third update user data **513**-U is made non-volatile.
[0127] In response to receiving the response to the second XOR/write command, the host **2** transmits the third XOR/write command to the fourth memory system **3-4** (A**12**).
[0128] In response to receiving the third XOR/write command, the fourth memory system **3-4** receives the third XOR data **523** from the third memory system **3-3** (A**13**). The fourth memory system **3-4** reads the parity **51**P from the NAND flash memory **5** of the fourth memory system **3-4** (A**14**). The fourth memory system **3-4** performs an XOR operation on the parity **51**P and the third XOR data **523**, thereby generating the update parity **51**P-U (A**15**). The fourth memory system **3-4** writes the update parity **51**P-U into the NAND flash memory **5** of the fourth memory system **3-4** (A**16**). Then, the fourth memory system **3-4** transmits a response to the third XOR/write command to the host **2** (A**17**). Note that the fourth memory system **3-4** may transmit the response to the third XOR/write command to the host **2** when it is guaranteed that the update parity **51**P-U is made non-volatile.
[0129] By the fourth update operation described above, in the information processing system **1**, the offloading of the update operation from the host **2** to the memory systems **3** can be realized, and it is possible to prevent the congestion of data transfer to the fourth memory system **3-4** storing the parity **51**P.
[0130] Note that the operation in each of the first memory system **3-1**, the second memory system **3-2**, the third memory system **3-3**, and the fourth memory system **3-4** is realized by, for example, the CPU **14** that functions as the command reception module **141**, the read processing module **142**, the XOR processing module **143**, and the write processing module **144** described above with reference to FIG. **2**.
[0131] As an example, an operation in the first memory system **3-1** in a case where the fourth update operation is performed will be described.
[0132] FIG. **9** illustrates an example of a specific operation in the first memory system **3-1** in a case where the fourth update operation is performed.
[0133] The command reception module **141** receives the first XOR/write command from the host **2** ((**1**) in FIG. **9**). In response to receiving the first XOR/write command, the command reception module **141** receives the first update user data **511**-U from the host **2** ((**2**) in FIG. **9**). On the basis of the first XOR/write command, the command reception module **141** sends a read instruction to read the first user data **511** to the read processing module **142** ((**3**) in FIG. **9**).
[0134] In response to the read instruction, the read processing module **142** reads the first user data **511** from the NAND flash memory **5** ((**4**) in FIG. **9**). Then, the read processing module **142** sends the read first user data **511** to the XOR processing module **143** ((**5**) in FIG. **9**).
[0135] Next, the command reception module **141** sends, to the XOR processing module **143**, the first update user data **511**-U, and an XOR instruction to perform an XOR operation on the first user data **511** and the first update user data **511**-U ((**6**) in FIG. **9**).
[0136] In response to the XOR instruction, the XOR processing module **143** performs an XOR operation on the first user data **511** and the first update user data **511**-U, thereby generating the first XOR data **521**. The XOR processing module **143** stores the generated first XOR data **521** in the

internal buffer **61-1** ((**7**) in FIG. **9**).
[0137] Then, the command reception module **141** sends, to the write processing module **144**, the first update user data **511**-U and a write instruction to write the first update user data **511**-U ((**8**) in FIG. **9**).
[0138] In response to the write instruction, the write processing module **144** writes the first update user data **511**-U into the NAND flash memory **5** ((**9**) in FIG. **9**). Then, the write processing module **144** updates the logical-to-physical address conversion table **31** such that a logical address, which is associated with the physical memory location where the first user data **511** is stored, is associated with the physical memory location where the first update user data **511**-U is stored ((**10**) in FIG. **9**). As a result, the first user data **511** is invalidated, and user data corresponding to the logical address is updated to the first update user data **511**-U.
[0139] Then, the command reception module **141** transmits a response to the first XOR/write command to the host **2** ((**11**) in FIG. **9**).
[0140] Similarly, the operation in each of the second memory system **3-2**, the third memory system **3-3**, and the fourth memory system **3-4** may be realized by the CPU **14** that functions as the command reception module **141**, the read processing module **142**, the XOR processing module **143**, and the write processing module **144**.
[0141] FIG. **10** illustrates an example of a fifth update operation in the information processing system **1**. Similarly to the fourth update operation, the fifth update operation is executed for updating the first user data **511** stored in the first memory system **3-1** with the first update user data **511**-U and updating the third user data **513** stored in the third memory system **3-3** with the third update user data **513**-U.
[0142] The DRAM **22** of the host **2** stores the first update user data **511**-U and the third update user data **513**-U.
[0143] The CPU **21** of the host **2** transmits the first XOR/write command for requesting an XOR operation and the writing of the first update user data **511**-U, to the first memory system **3-1** ((**1-1**) in FIG. **10**). In addition, the CPU **21** of the host **2** transmits a second XOR/write command for requesting an XOR operation and the writing of the third update user data **513**-U, to the third memory system **3-3** ((**1-2**) in FIG. **10**).
[0144] In response to receiving the first XOR/write command from the host **2**, the first memory system **3-1** receives the first update user data **511**-U from the host **2**. The first memory system **3-1** reads the first user data **511** from the NAND flash memory **5** of the first memory system **3-1**. The first memory system **3-1** performs an XOR operation on the first user data **511** and the first update user data **511**-U, thereby generating the first XOR data **521**. The first XOR data **521** is stored in the internal buffer **61-1** of the first memory system **3-1**. Then, the first memory system **3-1** writes the first update user data **511**-U into the NAND flash memory **5** of the first memory system **3-1**.
[0145] In response to receiving the second XOR/write command from the host **2**, the third memory system **3-3** receives the third update user data **513**-U from the host **2**. The third memory system **3-3** reads the third user data **513** from the NAND flash memory **5** of the third memory system **3-3**. The third memory system **3-3** performs an XOR operation on the third user data **513** and the third update user data **513**-U, thereby generating preliminary XOR data **523**-P. The preliminary XOR data **523**-P is stored in the internal buffer **61-3** of the third memory system **3-3**. Then, the third memory system **3-3** writes the third update user data **513**-U into the NAND flash memory **5** of the third memory system **3-3**. The operation by the third memory system **3-3** according to the second XOR/write command is performed, for example, in parallel with the operation by the first memory system **3-1** according to the first XOR/write command.
[0146] Next, the CPU **21** of the host **2** transmits an XOR command for requesting an XOR operation on the first XOR data **521** and the preliminary XOR data **523**-P to the third memory system **3-3** ((**2**) in FIG. **10**).
[0147] In response to receiving the XOR command from the host **2**, the third memory system **3-3**

receives the first XOR data **521** from the first memory system **3-1**. The third memory system **3-3** performs an XOR operation on the first XOR data **521** and the preliminary XOR data **523**-P, thereby generating the third XOR data **523**. The third XOR data **523** is stored in the internal buffer **61-3** of the third memory system **3-3**.

[0148] Next, the CPU **21** of the host **2** transmits a third XOR/write command for requesting an XOR operation and the writing of the update parity **51**P-U, to the fourth memory system **3-4** ((**3**) in FIG. **10**).

[0149] In response to receiving the third XOR/write command from the host **2**, the fourth memory system **3-4** receives the third XOR data **523** from the third memory system **3-3**. The fourth memory system **3-4** reads the parity **51**P from the NAND flash memory **5** of the fourth memory system **3-4**. The fourth memory system **3-4** performs an XOR operation on the third XOR data **523** and the parity **51**P, thereby generating the update parity **51**P-U. Then, the fourth memory system **3-4** writes the update parity **51**P-U into the NAND flash memory **5** of the fourth memory system **3-4**.

[0150] By the fifth update operation described above, in the information processing system **1**, the first user data **511** stored in the first memory system **3-1** is updated with the first update user data **511**-U, the third user data **513** stored in the third memory system **3-3** is updated with the third update user data **513**-U, and the parity **51**P stored in the fourth memory system **3-4** is updated with the update parity **51**P-U.

[0151] In the fifth update operation, the update parity **51**P-U is generated by using the resources of the memory systems **3**. Therefore, in the fifth update operation, the offloading of the update operation from the host **2** to the memory systems **3** can be realized.

[0152] In addition, in the fifth update operation, the first XOR data **521** is transmitted from the first memory system **3-1** to the third memory system **3-3**, and the third XOR data **523** is transmitted from the third memory system **3-3** to the fourth memory system **3-4**. As a result, it is possible to prevent the congestion of data transfer to the fourth memory system **3-4** storing the parity **51**P.

[0153] Further, in the fifth update operation, the operation by the first memory system **3-1** according to the first XOR/write command and the operation by the third memory system **3-3** according to the second XOR/write command are performed in parallel. As a result, the operation by the first memory system **3-1** and the operation by the third memory system **3-3** can be partially parallelized. Therefore, in the information processing system **1**, the time required for the fifth update operation may be shorter than the time required for the fourth update operation.

[0154] As described above, in the information processing system **1** that performs the fifth update operation, performance of the whole of the information processing system **1** can be improved while reducing the load on the host **2**.

[0155] FIG. **11** is a sequence diagram illustrating a specific example of the fifth update operation in the information processing system **1**.

[0156] First, the host **2** transmits the first XOR/write command and the first update user data **511**-U to the first memory system **3-1** (B**1**). The host **2** may transmit the first update user data **511**-U to the first memory system **3-1** that has received the first XOR/write command. In addition, the host **2** transmits the second XOR/write command and the third update user data **513**-U to the third memory system **3-3** (B**2**). The host **2** may transmit the third update user data **513**-U to the third memory system **3-3** that has received the second XOR/write command.

[0157] In response to receiving the first XOR/write command and the first update user data **511**-U, the first memory system **3-1** reads the first user data **511** from the NAND flash memory **5** of the first memory system **3-1** (B**3**). The first memory system **3-1** performs an XOR operation on the first user data **511** and the first update user data **511**-U, thereby generating the first XOR data **521**, and stores the first XOR data **521** in the internal buffer **61-1** (B**4**). The first memory system **3-1** writes the first update user data **511**-U into the NAND flash memory **5** of the first memory system **3-1**, and invalidates the first user data **511** (B**5**). Then, the first memory system **3-1** transmits a response to the first XOR/write command to the host **2** (B**6**).

[0158] In response to receiving the second XOR/write command and the third update user data **513**-U, the third memory system **3-3** reads the third user data **513** from the NAND flash memory **5** of the third memory system **3-3** (B**7**). The third memory system **3-3** performs an XOR operation on the third user data **513** and the third update user data **513**-U, thereby generating the preliminary XOR data **523**-P, and stores the generated preliminary XOR data **523**-P in the internal buffer **61-3** (B**8**). The third memory system **3-3** writes the third update user data **513**-U into the NAND flash memory **5** of the third memory system **3-3**, and invalidates the third user data **513** (B**9**). Then, the third memory system **3-3** transmits a response to the second XOR/write command to the host **2** (B**10**).
[0159] Next, in response to receiving the response to the first XOR/write command and the response to the second XOR/write command, the host **2** transmits an XOR command to the third memory system **3-3** (B**11**).
[0160] In response to receiving the XOR command, the third memory system **3-3** receives the first XOR data **521** from the first memory system **3-1** (B**12**). The third memory system **3-3** performs an XOR operation on the first XOR data **521** and the preliminary XOR data **523**-P, thereby generating the third XOR data **523**, and stores the third XOR data **523** in the internal buffer **61-3** (B**13**). Then, the third memory system **3-3** transmits a response to the XOR command to the host **2** (B**14**). The subsequent operation from B**15** to B**20** is similar to the operation from A**12** to A**17** in the fourth update operation described above with reference to FIG. **8**.
[0161] By the fifth update operation described above, in the information processing system **1**, the offloading of the update operation from the host **2** to the memory systems **3** can be realized, and it is possible to prevent the congestion of data transfer to the fourth memory system **3-4** storing the parity **51**P. In addition, by at least partially parallelizing the operation in the first memory system **3-1** and the operation in the third memory system **3-3**, the time required for the fifth update operation may be made shorter than the time required for the fourth update operation.
Second Embodiment
[0162] In the information processing system **1** according to the first embodiment, in a case where the multiple memory systems **3** are configured as RAID, an update operation is performed such that at least a part of the user data **51**D included in one ECC frame **51**E is updated across two or more memory systems **3**. In contrast, in an information processing system **1** according to a second embodiment, in a case where multiple memory systems **3** are configured as RAID, an operation (rebuild operation) of restoring data stored in a failed memory system **3** and reconstructing the RAID is performed.
[0163] A configuration of the information processing system **1** according to the second embodiment is similar to that of the information processing system **1** according to the first embodiment. The second embodiment is different in that the rebuild operation is performed in the host **2** and the memory systems **3**. Hereinafter, the differences from the first embodiment will be mainly described.
[0164] FIG. **12** illustrates an example of a first rebuild operation in the information processing system **1**. The first rebuild operation is an operation of restoring data stored in a failed memory system **3** and reconstructing the RAID while reducing resources of the host **2** being used. Here, a case where the second memory system **3-2** fails will be explained. A fifth memory system **3-5** is a memory system **3** to be replaced from the failed second memory system **3-2**. In other words, in the information processing system **1**, in order to cope with the failure of the second memory system **3-2**, the rebuild operation is performed such that the RAID is reconstructed by the first memory system **3-1**, the third memory system **3-3**, the fourth memory system **3-4**, and the fifth memory system **3-5**.
[0165] It is assumed that the first user data **511**, the second user data **512**, the third user data **513**, and the parity **51**P are stored respectively in the four memory systems **3-1**, **3-2**, **3-3**, and **3-4** by the sequential write operation described above with reference to FIG. **3**. The first user data **511**, the

second user data **512**, the third user data **513**, and the parity **51**P constitute one ECC frame **51**E.
[0166] When the second memory system **3-2** fails, the host **2** transmits an XOR command (first XOR command) for requesting an XOR operation to the first memory system **3-1** ((**1**) in FIG. **12**). The first XOR command may include a logical address of the first user data **511**.
[0167] In response to the first XOR command from the host **2**, the first memory system **3-1** reads the first user data **511** from the NAND flash memory **5** of the first memory system **3-1**. The first memory system **3-1** performs an XOR operation on the first user data **511** and dummy data **531**, thereby generating fourth XOR data **524**. The fourth XOR data **524** is stored in the internal buffer **61-1**, for example. The dummy data **531** is a data string in which all bits are zero. The dummy data **531** and the first user data **511** have the same data length. Since all bits of the data string of the dummy data **531** are zero, the fourth XOR data **524** is the same as the first user data **511**. The first memory system **3-1** may store the first user data **511** as it is in the internal buffer **61-1** instead of performing the XOR operation on the first user data **511** and the dummy data **531**.
[0168] Then, the host **2** transmits a second XOR command to the third memory system **3-3** ((**2**) in FIG. **12**). The second XOR command may include a logical address of the third user data **513** and an identifier (for example, an address) of the internal buffer **61-1** of the first memory system **3-1** (more specifically, the storage location of the fourth XOR data **524**).
[0169] In response to the second XOR command from the host **2**, the third memory system **3-3** receives the fourth XOR data **524** from the first memory system **3-1**. The third memory system **3-3** reads the third user data **513** from the NAND flash memory **5** of the third memory system **3-3**. The third memory system **3-3** performs an XOR operation on the third user data **513** and the fourth XOR data **524**, thereby generating fifth XOR data **525**. The fifth XOR data **525** is stored in the internal buffer **61-3**, for example.
[0170] Next, the host **2** transmits a third XOR command to the fourth memory system **3-4** ((**3**) in FIG. **12**). The third XOR command may include a logical address of the parity **51**P and an identifier (for example, an address) of the internal buffer **61-3** of the third memory system **3-3** (more specifically, the storage location of the fifth XOR data **525**).
[0171] In response to the third XOR command from the host **2**, the fourth memory system **3-4** receives the fifth XOR data **525** from the third memory system **3-3**. The fourth memory system **3-4** reads the parity **51**P from the NAND flash memory **5** of the fourth memory system **3-4**. The fourth memory system **3-4** performs an XOR operation on the parity **51**P and the fifth XOR data **525**, thereby generating the second user data **512**. That is, the second user data **512** stored in the NAND flash memory **5** of the failed second memory system **3-2** is restored. The generated second user data **512** is stored in an internal buffer **61-4** of the fourth memory system **3-4**, for example.
[0172] Then, the host **2** transmits a write command for requesting the writing of the second user data **512**, to the fifth memory system **3-5** ((**4**) in FIG. **12**). The write command may include a logical address of the second user data **512** and an identifier (for example, an address) of the internal buffer **61-4** of the fourth memory system **3-4** (more specifically, the storage location of the second user data **512**).
[0173] In response to receiving the write command from the host **2**, the fifth memory system **3-5** receives the second user data **512** from the fourth memory system **3-4**. Then, the fifth memory system **3-5** writes the second user data **512** into the NAND flash memory **5** of the fifth memory system **3-5**.
[0174] By the first rebuild operation described above, in the information processing system **1**, the second user data **512** stored in the failed second memory system **3-2** can be restored and stored in the fifth memory system **3-5** that is replaced from the second memory system **3-2**. Then, by similarly repeating the first rebuild operation, it is possible to restore all data (that is, user data and parities) stored in the second memory system **3-2** and store the restored data in the fifth memory system **3-5**. As a result, in the information processing system **1**, the RAID can be reconstructed by the first memory system **3-1**, the third memory system **3-3**, the fourth memory system **3-4**, and the

fifth memory system **3-5**.
[0175] In the first rebuild operation, the second user data **512** is restored by using the resources of the memory systems **3**. Therefore, in the first rebuild operation, the offloading of the rebuild operation from the host **2** to the memory systems **3** can be realized.
[0176] Note that as another operation for restoring the second user data **512**, for example, an operation can be considered in which one of the first memory system **3-1**, the third memory system **3-3**, the fourth memory system **3-4**, and the fifth memory system **3-5** acquires the first user data **511**, the third user data **513**, and the parity **51**P, and performs an XOR operation to restore the second user data **512**. However, in this operation, data transfer to one memory system **3** that restores the second user data **512** is congested.
[0177] In contrast, in the first rebuild operation, the fourth XOR data **524** is transmitted from the first memory system **3-1** to the third memory system **3-3**, the fifth XOR data **525** is transmitted from the third memory system **3-3** to the fourth memory system **3-4**, and the second user data **512** is transmitted from the fourth memory system **3-4** to the fifth memory system **3-5**. As a result, it is possible to prevent the congestion of data transfer to a specific one of the first memory system **3-1**, the third memory system **3-3**, the fourth memory system **3-4**, and the fifth memory system **3-5**.
[0178] Therefore, in the information processing system **1** that performs the first rebuild operation, performance of the whole of the information processing system **1** can be improved while reducing the load on the host **2**.
[0179] FIG. **13** is a sequence diagram illustrating a specific example of the first rebuild operation in the information processing system **1**.
[0180] First, the host **2** transmits the first XOR command to the first memory system **3-1** (C**1**).
[0181] In response to receiving the first XOR command, the first memory system **3-1** reads the first user data **511** from the NAND flash memory **5** of the first memory system **3-1** (C**2**). The first memory system **3-1** performs an XOR operation on the first user data **511** and the dummy data **531**, thereby generating the fourth XOR data **524** (=the first user data **511**), and stores the fourth XOR data **524** in the internal buffer **61-1** (C**3**). Then, the first memory system **3-1** transmits a response to the first XOR command to the host **2** (C**4**).
[0182] In response to receiving the response to the first XOR command, the host **2** transmits the second XOR command to the third memory system **3-3** (C**5**).
[0183] In response to receiving the second XOR command, the third memory system **3-3** receives the fourth XOR data **524** from the first memory system **3-1** (C**6**). The third memory system **3-3** reads the third user data **513** from the NAND flash memory **5** of the third memory system **3-3** (C**7**). The third memory system **3-3** performs an XOR operation on the third user data **513** and the fourth XOR data **524**, thereby generating the fifth XOR data **525**, and stores the fifth XOR data **525** in the internal buffer **61-3** (C**8**). Then, the third memory system **3-3** transmits a response to the second XOR command to the host **2** (C**9**).
[0184] In response to receiving the response to the second XOR command, the host **2** transmits the third XOR command to the fourth memory system **3-4** (C**10**).
[0185] In response to receiving the third XOR command, the fourth memory system **3-4** receives the fifth XOR data **525** from the third memory system **3-3** (C**11**). The fourth memory system **3-4** reads the parity **51**P from the NAND flash memory **5** of the fourth memory system **3-4** (C**12**). The fourth memory system **3-4** performs an XOR operation on the parity **51**P and the fifth XOR data **525**, thereby generating the second user data **512**, and stores the second user data **512** in the internal buffer **61-4** (C**13**). Then, the fourth memory system **3-4** transmits a response to the third XOR command to the host **2** (C**14**).
[0186] In response to receiving the response to the third XOR command, the host **2** transmits the write command to the fifth memory system **3-5** (C**15**).
[0187] In response to receiving the write command, the fifth memory system **3-5** receives the second user data **512** from the fourth memory system **3-4** (C**16**). The fifth memory system **3-5**

writes the second user data **512** into the NAND flash memory **5** of the fifth memory system **3-5** (C**17**). Then, the fifth memory system **3-5** transmits a response to the write command to the host **2** (C**18**).
[0188] By the first rebuild operation described above, in the information processing system **1**, the offloading of the rebuild operation from the host **2** to the memory systems **3** can be realized, and it is possible to prevent the congestion of data transfer to a specific memory system **3**.
[0189] FIG. **14** illustrates an example of a second rebuild operation in the information processing system **1**. Similarly to the first rebuild operation, the second rebuild operation is an operation for reconstructing the RAID by the first memory system **3-1**, the third memory system **3-3**, the fourth memory system **3-4**, and the fifth memory system **3-5** in order to cope with a failure of the second memory system **3-2**.
[0190] When the second memory system **3-2** fails, the host **2** transmits a first XOR command to the first memory system **3-1** ((**1**) in FIG. **14**). The first XOR command may further include an identifier of the third memory system **3-3**.
[0191] In response to the first XOR command from the host **2**, the first memory system **3-1** reads the first user data **511** from the NAND flash memory **5** of the first memory system **3-1**. The first memory system **3-1** performs an XOR operation on the first user data **511** and the dummy data **531**, thereby generating the fourth XOR data **524**. The fourth XOR data **524** is stored in the internal buffer **61-1**, for example. Then, the first memory system **3-1** transmits a second XOR command to the third memory system **3-3** ((**2**) in FIG. **14**). The second XOR command may further include an identifier of the fourth memory system **3-4**.
[0192] In response to the second XOR command from the first memory system **3-1**, the third memory system **3-3** receives the fourth XOR data **524** from the first memory system **3-1**. The third memory system **3-3** reads the third user data **513** from the NAND flash memory **5** of the third memory system **3-3**. The third memory system **3-3** performs an XOR operation on the third user data **513** and the fourth XOR data **524**, thereby generating the fifth XOR data **525**. The fifth XOR data **525** is stored in the internal buffer **61-3**, for example. Then, the third memory system **3-3** transmits a third XOR command to the fourth memory system **3-4** ((**3**) in FIG. **14**). The third XOR command may further include an identifier of the fifth memory system **3-5**.
[0193] In response to the third XOR command from the third memory system **3-3**, the fourth memory system **3-4** receives the fifth XOR data **525** from the third memory system **3-3**. The fourth memory system **3-4** reads the parity **51**P from the NAND flash memory **5** of the fourth memory system **3-4**. The fourth memory system **3-4** performs an XOR operation on the parity **51**P and the fifth XOR data **525**, thereby generating the second user data **512**. That is, the second user data **512** stored in the NAND flash memory **5** of the failed second memory system **3-2** is restored. The generated second user data **512** is stored in the internal buffer **61-4**, for example. Then, the fourth memory system **3-4** transmits a write command for requesting the writing of the second user data **512**, to the fifth memory system **3-5** ((**4**) in FIG. **14**).
[0194] In response to the write command from the fourth memory system **3-4**, the fifth memory system **3-5** receives the second user data **512** from the fourth memory system **3-4**. Then, the fifth memory system **3-5** writes the second user data **512** into the NAND flash memory **5** of the fifth memory system **3-5**.
[0195] By the second rebuild operation described above, in the information processing system **1**, the second user data **512** stored in the failed second memory system **3-2** can be restored and stored in the fifth memory system **3-5** that is replaced from the second memory system **3-2**. Then, by similarly repeating the second rebuild operation, all data stored in the second memory system **3-2** can be restored, and the restored data can be stored in the fifth memory system **3-5**. As a result, in the information processing system **1**, the RAID can be reconstructed by the first memory system **3-1**, the third memory system **3-3**, the fourth memory system **3-4**, and the fifth memory system **3-5**.
[0196] In addition, in the second rebuild operation, the second user data **512** is restored by using

the resources of the memory systems **3**. Further, not only data (that is, the fourth XOR data **524**, the fifth XOR data **525**, and the second user data **512**) but also commands are transferred between the memory systems **3**. Accordingly, in the second rebuild operation, the host **2** needs to transmit only the first XOR command to the first memory system **3-1**. Therefore, in the second rebuild operation, the offloading of the rebuild operation from the host **2** to the memory systems **3** can be realized.
[0197] Further, in the second rebuild operation, the fourth XOR data **524** is transmitted from the first memory system **3-1** to the third memory system **3-3**, the fifth XOR data **525** is transmitted from the third memory system **3-3** to the fourth memory system **3-4**, and the second user data **512** is transmitted from the fourth memory system **3-4** to the fifth memory system **3-5**. As a result, it is possible to prevent the congestion of data transfer to a specific one of the first memory system **3-1**, the third memory system **3-3**, the fourth memory system **3-4**, and the fifth memory system **3-5**.
[0198] Therefore, in the information processing system **1** that performs the second rebuild operation, performance of the whole of the information processing system **1** can be improved while reducing the load on the host **2**.
[0199] FIG. **15** is a sequence diagram illustrating a specific example of the second rebuild operation in the information processing system **1**.
[0200] First, the host **2** transmits the first XOR command to the first memory system **3-1** (D**1**).
[0201] In response to receiving the first XOR command, the first memory system **3-1** reads the first user data **511** from the NAND flash memory **5** of the first memory system **3-1** (D**2**). The first memory system **3-1** performs an XOR operation on the first user data **511** and the dummy data **531**, thereby generating the fourth XOR data **524**, and stores the fourth XOR data **524** in the internal buffer **61-1** (D**3**). Then, the first memory system **3-1** transmits the second XOR command and the fourth XOR data **524** to the third memory system **3-3** (D**4**).
[0202] In response to receiving the second XOR command and the fourth XOR data **524**, the third memory system **3-3** reads the third user data **513** from the NAND flash memory **5** of the third memory system **3-3** (D**5**). The third memory system **3-3** performs an XOR operation on the third user data **513** and the fourth XOR data **524**, thereby generating the fifth XOR data **525**, and stores the fifth XOR data **525** in the internal buffer **61-3** (D**6**). Then, the third memory system **3-3** transmits the third XOR command and the fifth XOR data **525** to the fourth memory system **3-4** (D**7**).
[0203] In response to receiving the third XOR command and the fifth XOR data **525**, the fourth memory system **3-4** reads the parity **51**P from the NAND flash memory **5** of the fourth memory system **3-4** (D**8**). The fourth memory system **3-4** performs an XOR operation on the parity **51**P and the fifth XOR data **525**, thereby generating the second user data **512**, and stores the second user data **512** in the internal buffer **61-4** (D**9**). Then, the fourth memory system **3-4** transmits the write command and the second user data **512** to the fifth memory system **3-5** (D**10**).
[0204] In response to receiving the write command and the second user data **512**, the fifth memory system **3-5** writes the second user data **512** into the NAND flash memory **5** of the fifth memory system **3-5** (D**11**). Then, the fifth memory system **3-5** transmits a response to the write command to the fourth memory system **3-4** (D**12**).
[0205] In response to receiving the response to the write command, the fourth memory system **3-4** transmits a response to the third XOR command to the third memory system **3-3** (D**13**). In response to receiving the response to the third XOR command, the third memory system **3-3** transmits a response to the second XOR command to the first memory system **3-1** (D**14**). In response to receiving the response to the second XOR command, the first memory system **3-1** transmits a response to the first XOR command to the host **2** (D**15**).
[0206] By the second rebuild operation described above, in the information processing system **1**, the offloading of the rebuild operation from the host **2** to the memory systems **3** can be realized, and it is possible to prevent the congestion of data transfer to a specific memory system **3**.
[0207] Note that the operation in each of the first memory system **3-1**, the third memory system **3-**

3, the fourth memory system **3-4**, and the fifth memory system **3-5** is realized by, for example, the CPU **14** that functions as the command reception module **141**, the read processing module **142**, the XOR processing module **143**, and the write processing module **144** described above with reference to FIG. **2**.
[0208] As an example, an operation in the first memory system **3-1** in a case where the second rebuild operation is performed will be described.
[0209] FIG. **16** illustrates an example of a specific operation in the first memory system **3-1** in a case where the second rebuild operation is performed.
[0210] The command reception module **141** receives the first XOR command from the host **2** ((**1**) in FIG. **16**). On the basis of the first XOR command, the command reception module **141** sends a read instruction to read the first user data **511** to the read processing module **142** ((**2**) in FIG. **16**).
[0211] In response to the read instruction, the read processing module **142** reads the first user data **511** from the NAND flash memory **5** ((**3**) in FIG. **16**). Then, the read processing module **142** sends the read first user data **511** to the XOR processing module **143** ((**4**) in FIG. **16**).
[0212] Next, the command reception module **141** sends, to the XOR processing module **143**, the dummy data **531** and an XOR instruction to perform an XOR operation on the first user data **511** and the dummy data **531** ((**5**) in FIG. **16**).
[0213] In response to the XOR instruction, the XOR processing module **143** performs an XOR operation on the first user data **511** and the dummy data **531**, thereby generating the fourth XOR data **524**. The XOR processing module **143** stores the generated fourth XOR data **524** in the internal buffer **61-1** ((**6**) in FIG. **16**).
[0214] Next, the command reception module **141** transmits the second XOR command to the third memory system **3-3** ((**7**) in FIG. **16**). Thereafter, the command reception module **141** receives a response to the second XOR command from the third memory system **3-3** ((**8**) in FIG. **16**). In response to receiving the response to the second XOR command, the command reception module **141** transmits a response to the first XOR command to the host **2** ((**9**) in FIG. **16**).
[0215] Similarly, the operation in each of the third memory system **3-3**, the fourth memory system **3-4**, and the fifth memory system **3-5** may be realized by the CPU **14** that functions as the command reception module **141**, the read processing module **142**, the XOR processing module **143**, and the write processing module **144**.
[0216] As described above, according to the first and second embodiments, it is possible to improve the whole performance while reducing the load on the host **2**.
[0217] In the information processing system **1** according to the first embodiment, the controller **7** of the first memory system **3-1** stores the first user data **511** in the NAND flash memory **5** of the first memory system **3-1**. The controller **7** of the third memory system **3-3** stores the third user data **513** in the NAND flash memory **5** of the third memory system **3-3**. The first user data **511** and the third user data **513** constitute at least a part of one error correction code frame **51**E. In a case where the first user data **511** and the third user data **513** are updated, the host **2** transmits, to the first memory system **3-1**, the first update user data **511**-U updated from the first user data **511**, and transmits, to the third memory system **3-3**, the third update user data **513**-U updated from the third user data **513**. The controller **7** of the first memory system **3-1** generates the first XOR data **521** by performing an XOR operation on at least the first user data **511** and the first update user data **511**-U, and transmits the first XOR data **521** to the third memory system **3-3**. The controller **7** of the third memory system **3-3** generates the third XOR data **523** by performing an XOR operation on the third user data **513**, the third update user data **513**-U, and the first XOR data **521**, and transmits the third XOR data **523** to the fourth memory system **3-4**.
[0218] As a result, the offloading of the update operation from the host **2** to the memory systems **3** can be realized, and, for example, it is possible to prevent the congestion of data transfer to the fourth memory system **3-4** storing the parity **51**P.
[0219] In addition, in the information processing system **1** according to the second embodiment,

the controller **7** of the second memory system **3-2** stores the second user data **512** in the NAND flash memory **5** of the second memory system **3-2**. The controller **7** of the third memory system **3-3** stores the third user data **513** in the NAND flash memory **5** of the third memory system **3-3**. The controller **7** of the fourth memory system **3-4** stores the parity **51**P in the NAND flash memory **5** of the fourth memory system **3-4**. The second user data **512**, the third user data **513**, and the parity **51**P constitute at least a part of one error correction code frame **51**E. When the second memory system **3-2** fails, the controller **7** of the third memory system **3-3** generates the fifth XOR data **525** by performing an XOR operation on the fourth XOR data **524** and the third user data **513**, and transmits the fifth XOR data **525** to the fourth memory system **3-4**. The controller **7** of the fourth memory system **3-4** generates the second user data **512** by performing an XOR operation on the fifth XOR data **525** and the parity **51**P, and transmits the generated second user data **512** to the fifth memory system **3-5** that is replaced from the failed second memory system **3-2**.
[0220] As a result, in the information processing system **1**, the offloading of the rebuild operation from the host **2** to the memory systems **3** can be realized, and it is possible to prevent the congestion of data transfer to a specific memory system **3**.
[0221] Each of the various functions described in the first and second embodiments may be realized by a circuit (e.g., processing circuit). An exemplary processing circuit may be a programmed processor such as a central processing unit (CPU). The processor executes computer programs (instructions) stored in a memory thereby performs the described functions. The processor may be a microprocessor including an electric circuit. An exemplary processing circuit may be a digital signal processor (DSP), an application specific integrated circuit (ASIC), a microcontroller, a controller, or other electric circuit components. The components other than the CPU described according to the embodiments may be realized in a processing circuit.
[0222] While certain embodiments have been described, these embodiments have been presented by way of example only, and are not intended to limit the scope of the inventions. Indeed, the novel devices and methods described herein may be embodied in a variety of other forms; furthermore, various omissions, substitutions and changes in the form of the embodiments described herein may be made without departing from the spirit of the inventions. The accompanying claims and their equivalents are intended to cover such forms or modification as would fall within the scope and spirit of the inventions.

Claims

1-18. (canceled)
19. A memory system capable of communicating with a host and one or more external memory systems, the memory system comprising: a nonvolatile memory; and a controller electrically connected to the nonvolatile memory and configured to: store first data in the nonvolatile memory; receive, from the host, a first request and first update data updated from the first data; receive, from a first memory system that is one of the one or more external memory systems, first exclusive-logical-OR data; and in response to receiving the first request: generate second exclusive-logical-OR data by performing an exclusive-logical-OR operation on the first data, the first update data, and the first exclusive-logical-OR data; transmit, to the host, a first response to the first request; and transmit, to a second memory system that is another one of the one or more external memory systems, the second exclusive-logical-OR data.
20. The memory system according to claim 19, wherein a first controller of the first memory system is configured to store second data in a first nonvolatile memory of the first memory system, and the first data and the second data constitute at least a part of an error correction code frame.
21. The memory system according to claim 20, wherein a second controller of the second memory system is configured to store a parity in a second nonvolatile memory of the second memory system, and the first data, the second data, and the parity constitute at least a part of the error

correction code frame.
22. The memory system according to claim 21, wherein the first exclusive-logical-OR data is generated in the first memory system by performing an exclusive-logical-OR operation on at least the second data and second update data updated from the second data.
23. The memory system according to claim 22, wherein an update parity is generated in the second memory system by performing an exclusive-logical-OR operation on the parity and the second exclusive-logical-OR data, and the first update data, the second update data, and the update parity constitute at least a part of the error correction code frame that is updated.
24. The memory system according to claim 19, wherein the first memory system includes a volatile memory, and the first request specifies an address of a first location of the volatile memory of the first memory system where the first exclusive-logical-OR data is stored.
25. The memory system according to claim 24, wherein the controller is further configured to receive, from the first memory system, the first exclusive-logical-OR data by accessing the first location of the volatile memory of the first memory system based on the address specified in the first request.
26. The memory system according to claim 19, wherein the controller is further configured to: store the first update data in the nonvolatile memory; and invalidate the first data.
27. The memory system according to claim 26, wherein the first data is stored in a second location of the nonvolatile memory, the first update data is stored in a third location of the nonvolatile memory, and the controller is further configured to: manage a logical-to-physical address conversion table; and in invalidating the first data, update the logical-to-physical address conversion table such that a logical address of the first data is mapped from a physical address of the second location to a physical address of the third location.
28. The memory system according to claim 19, further comprising a volatile memory, wherein the controller is further configured to: store the second exclusive-logical-OR data in the volatile memory; and transmit, to the second memory system, the second exclusive-logical-OR data from the volatile memory.
29. A method of controlling a nonvolatile memory, comprising: storing first data in the nonvolatile memory; receiving, from a host, a first request and first update data updated from the first data; receiving, from a first memory system that is one of one or more external memory systems, first exclusive-logical-OR data; and in response to receiving the first request: generating second exclusive-logical-OR data by performing an exclusive-logical-OR operation on the first data, the first update data, and the first exclusive-logical-OR data; transmitting, to the host, a first response to the first request; and transmitting, to a second memory system that is another one of the one or more external memory systems, the second exclusive-logical-OR data.
30. The method according to claim 29, wherein a first controller of the first memory system is configured to store second data in a first nonvolatile memory of the first memory system, and the first data and the second data constitute at least a part of an error correction code frame.
31. The method according to claim 30, wherein a second controller of the second memory system is configured to store a parity in a second nonvolatile memory of the second memory system, and the first data, the second data, and the parity constitute at least a part of the error correction code frame.
32. The method according to claim 31, wherein the first exclusive-logical-OR data is generated in the first memory system by performing an exclusive-logical-OR operation on at least the second data and second update data updated from the second data.
33. The method according to claim 32, wherein an update parity is generated in the second memory system by performing an exclusive-logical-OR operation on the parity and the second exclusive-logical-OR data, and the first update data, the second update data, and the update parity constitute at least a part of the error correction code frame that is updated.
34. The method according to claim 29, wherein the first memory system includes a volatile memory, and the first request specifies an address of a first location of the volatile memory of the

first memory system where the first exclusive-logical-OR data is stored.
35. The method according to claim 34, further comprising: accessing the first location of the volatile memory of the first memory system based on the address specified in the first request, to receive the first exclusive-logical-OR data from the first memory system.
36. The method according to claim 29, further comprising: storing the first update data in the nonvolatile memory; and invalidating the first data.
37. The method according to claim 36, wherein the first data is stored in a second location of the nonvolatile memory, the first update data is stored in a third location of the nonvolatile memory, and the method further comprises: managing a logical-to-physical address conversion table; and in invalidating the first data, updating the logical-to-physical address conversion table such that a logical address of the first data is mapped from a physical address of the second location to a physical address of the third location.
38. The method according to claim 29, further comprising: storing the second exclusive-logical-OR data in a volatile memory; and transmitting, to the second memory system, the second exclusive-logical-OR data from the volatile memory.