(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
14 December 2023 (14.12.2023)

WIPO | PCT

(10) International Publication Number
WO 2023/239508 A1

(51) International Patent Classification:
G06F 3/06 (2006.01)

(21) International Application Number:
PCT/US2023/021373

(22) International Filing Date:
08 May 2023 (08.05.2023)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
17/834,051 07 June 2022 (07.06.2022) US

(71) Applicant: **WESTERN DIGITAL TECHNOLOGIES, INC.** [US/US]; 5601 Great Oaks Parkway, San Jose, California 95119 (US).

(72) Inventors: **SEGEV, Amir**; c/o Western Digital Technologies, Inc., 5601 Great Oaks Parkway, San Jose, California 95119 (US). **BENISTY, Shay**; c/o Western Digital Technologies, Inc., 5601 Great Oaks Parkway, San Jose, California 95119 (US).

(74) Agent: **SANDERS, Jason A.** et al.; Sanders IP Law, 240 N. East Promontory, Suite 200, Farmington, Utah 84025 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CV, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, MG, MK, MN, MU, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, CV, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SC, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, ME, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report (Art. 21(3))*

(54) Title: DATA STORAGE DEVICE AND METHOD FOR HOST BUFFER MANAGEMENT

(57) Abstract: A data storage device and method for host buffer management are provided. In one embodiment, a data storage device is provided comprising a non-volatile memory and a controller. The controller is configured to receive a read command from a host; read data from the non-volatile memory; identify a location in a host memory buffer (HMB) in the host that is available to store the data; write the data to the location in the HMB; and inform the host of the location in the HMB that stores the data. Other embodiments are possible, and each of the embodiments can be used alone or together in combination.

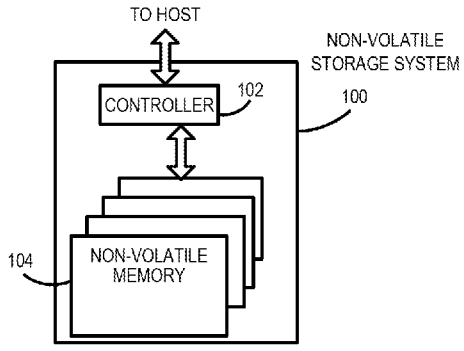


FIG. 1A

WO 2023/239508 A1

Data Storage Device and Method for Host Buffer Management

CROSS-REFERENCE TO RELATED APPLICATION(S)

[0001] This application claims the benefit of and hereby incorporates by reference, for all purposes, the entirety of the contents of U.S. Nonprovisional Application No. 17/834,051, filed June 7, 2022, and entitled "Data Storage Device and Method for Host Buffer Management".

Background

[0002] A host can read data from a memory in a data storage device. In some environments, during a read operation, a host allocates space in its volatile memory for buffers to store the requested data and for a pointer list with pointers that point to the allocated buffers. The data storage device reads a read command and the pointer list from the host's memory. After the data storage device reads the requested data from its memory, the data storage device writes the data in the buffer location in the host, as pointed to by the pointer list. The data storage device then informs the host that the read operation is complete. In response, the host reads the pointer lists to determine the location in the host's volatile memory where the data is stored, reads the data from that location, and then releases the space in the volatile memory that was allocated for the buffer and the pointer list.

Brief Description of the Drawings

[0003] Figure 1A is a block diagram of a data storage device of an embodiment.

[0004] Figure 1B is a block diagram illustrating a storage module of an embodiment.

[0005] Figure 1C is a block diagram illustrating a hierarchical storage system of an embodiment.

[0006] Figure 2A is a block diagram illustrating components of the controller of the data storage device illustrated in Figure 1A according to an embodiment.

[0007] Figure 2B is a block diagram illustrating components of the memory data storage device illustrated in Figure 1A according to an embodiment.

[0008] Figure 3 is a block diagram of a host and data storage device of an embodiment.

[0009] Figure 4 is a diagram illustrating a read command flow of an embodiment.

[0010] Figure 5 is a block diagram of a host and data storage device of an embodiment.

[0011] Figure 6 is a diagram illustrating a host memory buffer (HMB)-read command flow of an embodiment.

[0012] Figure 7 is an illustration of a completion queue entry layout of an embodiment.

Detailed Description

Overview

[0013] By way of introduction, the below embodiments relate to a data storage device and method for host buffer management. In one embodiment, a data storage device is provided comprising a non-volatile memory and a controller. The controller is configured to receive a read command from a host; read data from the non-volatile memory; identify a location in a host memory buffer (HMB) in the host that is available to store the data; write the data to the location in the HMB; and inform the host of the location in the HMB that stores the data.

[0014] In some embodiments, the controller, not the host, is responsible for managing where the data is written in the host.

[0015] In some embodiments, the controller comprises: a resource server configured to manage memory allocation in the HMB; and a direct memory access module configured to send a request to the resource server to identify the location in the HMB that is available to store the data.

[0016] In some embodiments, the controller is further configured to inform the host of the location in the HMB by storing a completion queue entry in a completion queue, wherein the completion queue entry comprises a pointer to the location.

[0017] In some embodiments, the controller is further configured to receive, from the host, an indication that the location in the HMB can be released.

[0018] In some embodiments, the controller is further configured to release the location in response to receiving the indication.

[0019] In some embodiments, the controller is further configured to inform the host of the location in the HMB by sending a pointer to the host; and receiving the indication from the host comprises receiving the pointer back from the host.

[0020] In some embodiments, the controller is further configured to receive the read command by fetching the read command from a submission queue in the host in response to determining a doorbell register indicates that the read command is available to be fetched from the submission queue.

[0021] In some embodiments, the data storage device further comprises a volatile memory, wherein the controller is further configured to store the data in the volatile memory after the data is read from the non-volatile memory but before the data is written to the location in the HMB.

[0022] In some embodiments, the controller is further configured to error correct the data prior to writing the data to the location in the HMB.

[0023] In some embodiments, the data storage device comprises a solid-state drive (SSD).

[0024] In some embodiments, the non-volatile memory comprises a three-dimensional memory.

[0025] In another embodiment, a method is provided that is performed in a host in communication with a data storage device. The method comprises: generating a request for data stored in the data storage device; receiving, from the data storage device, a pointer to a location in a memory in the host where the data storage device stored the data after executing the request; reading the data from the location in the memory; and informing the data storage device that the location in the memory can be released.

[0026] In some embodiments, the data storage device, not the host, is responsible for managing where the data is written in the host.

[0027] In some embodiments, a resource server in the data storage device is responsible for managing where the data is written in the host.

[0028] In some embodiments, the memory in the host comprises a host memory buffer (HMB).

[0029] In some embodiments, the pointer is received in a completion queue entry written in a completion queue by the data storage device.

[0030] In some embodiments, informing the data storage device that the location in the memory can be released comprises sending the pointer back to the data storage device.

[0031] In some embodiments, the method further comprises placing the read command in a submission queue; and informing the data storage device that the read command is ready to be fetched from the submission queue.

[0032] In another embodiment, a data storage device is provided comprising a memory and means for responding to a read command from a host by: reading data from the memory; selecting a location in host memory that is available to store the data; writing the data to the location in the host memory; and informing the host of the location in the host memory that stores the data.

[0033] Other embodiments are possible, and each of the embodiments can be used alone or together in combination. Accordingly, various embodiments will now be described with reference to the attached drawings.

Embodiments

[0034] The following embodiments relate to a data storage device (DSD). As used herein, a "data storage device" refers to a device that stores data. Examples of DSDs include, but are not limited to, hard disk drives (HDDs), solid state drives (SSDs), tape drives, hybrid drives, etc. Details of example DSDs are provided below.

[0035] Data storage devices suitable for use in implementing aspects of these embodiments are shown in Figures 1A-1C. Figure 1A is a block diagram illustrating a data storage device 100 according to an embodiment of the subject matter described herein. Referring to Figure 1A, data storage device 100 includes a controller 102 and non-volatile memory that may be made up of one or more non-volatile memory die 104. As used herein, the term die refers to the collection of non-volatile memory cells, and associated circuitry for managing the physical operation of those non-volatile memory cells, that are formed on a single semiconductor substrate. Controller 102 interfaces with a host system and transmits command sequences for read, program, and erase operations to non-volatile memory die 104.

[0036] The controller 102 (which may be a non-volatile memory controller (e.g., a flash, resistive random-access memory (ReRAM), phase-change memory (PCM), or magnetoresistive random-access memory (MRAM) controller)) can take the form of processing circuitry, a microprocessor or processor, and a computer-readable medium that stores computer-readable program code (e.g., firmware) executable by the (micro)processor, logic gates, switches, an application specific integrated circuit (ASIC), a programmable logic controller, and an embedded microcontroller, for example. The controller 102 can be configured with hardware and/or firmware to perform the various functions described below and shown in the flow diagrams. Also, some of the components shown as being internal to the controller can also be stored external to the controller, and other components can be used. Additionally, the phrase "operatively in communication with" could mean directly in communication with or indirectly (wired or wireless) in communication with through one or more components, which may or may not be shown or described herein.

[0037] As used herein, a non-volatile memory controller is a device that manages data stored on non-volatile memory and communicates with a host, such as a computer or electronic device. A non-volatile memory controller can have various functionality in addition to the specific functionality described herein. For example, the non-volatile memory controller can format the non-volatile memory to ensure the memory is operating properly, map out bad non-volatile memory cells, and allocate spare cells to be substituted for future failed cells. Some part of the spare cells can be used to hold firmware to operate the non-volatile memory controller and implement other features. In operation, when a host needs to read data from or write data to the non-volatile memory, it can communicate with the non-volatile memory controller. If the host provides a logical address to which data is to be read/written, the non-volatile memory controller can convert the logical address received from the host to a physical address in the non-volatile memory. (Alternatively, the host can provide the physical address.) The non-volatile memory controller can also perform various memory management functions, such as, but not limited to,

wear leveling (distributing writes to avoid wearing out specific blocks of memory that would otherwise be repeatedly written to) and garbage collection (after a block is full, moving only the valid pages of data to a new block, so the full block can be erased and reused).

[0038] Non-volatile memory die 104 may include any suitable non-volatile storage medium, including resistive random-access memory (ReRAM), magnetoresistive random-access memory (MRAM), phase-change memory (PCM), NAND flash memory cells and/or NOR flash memory cells. The memory cells can take the form of solid-state (e.g., flash) memory cells and can be one-time programmable, few-time programmable, or many-time programmable. The memory cells can also be single-level cells (SLC), multiple-level cells (MLC) (e.g., dual-level cells, triple-level cells (TLC), quad-level cells (QLC), etc.) or use other memory cell level technologies, now known or later developed. Also, the memory cells can be fabricated in a two-dimensional or three-dimensional fashion.

[0039] The interface between controller 102 and non-volatile memory die 104 may be any suitable flash interface, such as Toggle Mode 200, 400, or 800. In one embodiment, the data storage device 100 may be a card based system, such as a secure digital (SD) or a micro secure digital (micro-SD) card. In an alternate embodiment, the data storage device 100 may be part of an embedded data storage device.

[0040] Although, in the example illustrated in Figure 1A, the data storage device 100 (sometimes referred to herein as a storage module) includes a single channel between controller 102 and non-volatile memory die 104, the subject matter described herein is not limited to having a single memory channel. For example, in some architectures (such as the ones shown in Figures 1B and 1C), two, four, eight or more memory channels may exist between the controller and the memory device, depending on controller capabilities. In any of the embodiments described herein, more than a single channel may exist between the controller and the memory die, even if a single channel is shown in the drawings.

[0041] Figure 1B illustrates a storage module 200 that includes plural non-volatile data storage devices 100. As such, storage module 200 may include a storage controller 202 that interfaces with a host and with data storage device 204, which includes a plurality of data storage devices 100. The interface between storage controller 202 and data storage devices 100 may be a bus interface, such as a serial advanced technology attachment (SATA), peripheral component interconnect express (PCIe) interface, or double-data-rate (DDR) interface. Storage module 200, in one embodiment, may be a solid state drive (SSD), or non-volatile dual in-line memory module (NVDIMM), such as found in server PC or portable computing devices, such as laptop computers, and tablet computers.

[0042] Figure 1C is a block diagram illustrating a hierarchical storage system. A hierarchical storage system 250 includes a plurality of storage controllers 202, each of which controls a respective data storage device 204. Host systems 252 may access memories within the storage system 250 via a bus interface. In one embodiment, the bus interface may be a Non-Volatile Memory Express (NVMe) or Fibre Channel over Ethernet (FCoE) interface. In one embodiment, the system illustrated in Figure 1C may be a rack mountable mass storage system that is accessible by multiple host computers, such as would be found in a data center or other location where mass storage is needed.

[0043] Figure 2A is a block diagram illustrating components of controller 102 in more detail. Controller 102 includes a front-end module 108 that interfaces with a host, a back-end module 110 that interfaces with the one or more non-volatile memory die 104, and various other modules that perform functions which will now be described in detail. A module may take the form of a packaged functional hardware unit designed for use with other components, a portion of a program code (e.g., software or firmware) executable by a (micro)processor or processing circuitry that usually performs a particular function of related functions, or a self-contained hardware or software component that interfaces with a larger system, for example. Also, "means" for performing a function can be implemented with at least any of the structure noted herein for the controller and can be pure hardware or a combination of hardware and computer-readable program code.

[0044] Referring again to modules of the controller 102, a buffer manager/bus controller 114 manages buffers in random access memory (RAM) 116 and controls the internal bus arbitration of controller 102. A read only memory (ROM) 118 stores system boot code. Although illustrated in Figure 2A as located separately from the controller 102, in other embodiments one or both of the RAM 116 and ROM 118 may be located within the controller. In yet other embodiments, portions of RAM and ROM may be located both within the controller 102 and outside the controller.

[0045] Front-end module 108 includes a host interface 120 and a physical layer interface (PHY) 122 that provide the electrical interface with the host or next level storage controller. The choice of the type of host interface 120 can depend on the type of memory being used. Examples of host interfaces 120 include, but are not limited to, SATA, SATA Express, serially attached small computer system interface (SAS), Fibre Channel, universal serial bus (USB), PCIe, and NVMe. The host interface 120 typically facilitates transfer for data, control signals, and timing signals.

[0046] Back-end module 110 includes an error correction code (ECC) engine 124 that encodes the data bytes received from the host, and decodes and error corrects the data bytes read from the

non-volatile memory. A command sequencer 126 generates command sequences, such as program and erase command sequences, to be transmitted to non-volatile memory die 104. A RAID (Redundant Array of Independent Drives) module 128 manages generation of RAID parity and recovery of failed data. The RAID parity may be used as an additional level of integrity protection for the data being written into the memory device 104. In some cases, the RAID module 128 may be a part of the ECC engine 124. A memory interface 130 provides the command sequences to non-volatile memory die 104 and receives status information from non-volatile memory die 104. In one embodiment, memory interface 130 may be a double data rate (DDR) interface, such as a Toggle Mode 200, 400, or 800 interface. A flash control layer 132 controls the overall operation of back-end module 110.

[0047] The data storage device 100 also includes other discrete components 140, such as external electrical interfaces, external RAM, resistors, capacitors, or other components that may interface with controller 102. In alternative embodiments, one or more of the physical layer interface 122, RAID module 128, media management layer 138 and buffer management/bus controller 114 are optional components that are not necessary in the controller 102.

[0048] Figure 2B is a block diagram illustrating components of non-volatile memory die 104 in more detail. Non-volatile memory die 104 includes peripheral circuitry 141 and non-volatile memory array 142. Non-volatile memory array 142 includes the non-volatile memory cells used to store data. The non-volatile memory cells may be any suitable non-volatile memory cells, including ReRAM, MRAM, PCM, NAND flash memory cells and/or NOR flash memory cells in a two-dimensional and/or three-dimensional configuration. Non-volatile memory die 104 further includes a data cache 156 that caches data. Peripheral circuitry 141 includes a state machine 152 that provides status information to the controller 102.

[0049] Returning again to Figure 2A, the flash control layer 132 (which will be referred to herein as the flash translation layer (FTL) or, more generally, the "media management layer," as the memory may not be flash) handles flash errors and interfaces with the host. In particular, the FTL, which may be an algorithm in firmware, is responsible for the internals of memory management and translates writes from the host into writes to the memory 104. The FTL may be needed because the memory 104 may have limited endurance, may be written in only multiples of pages, and/or may not be written unless it is erased as a block. The FTL understands these potential limitations of the memory 104, which may not be visible to the host. Accordingly, the FTL attempts to translate the writes from host into writes into the memory 104.

[0050] The FTL may include a logical-to-physical address (L2P) map (sometimes referred to herein as a table or data structure) and allotted cache memory. In this way, the FTL translates logical block addresses ("LBAs") from the host to physical addresses in the memory 104. The

FTL can include other features, such as, but not limited to, power-off recovery (so that the data structures of the FTL can be recovered in the event of a sudden power loss) and wear leveling (so that the wear across memory blocks is even to prevent certain blocks from excessive wear, which would result in a greater chance of failure).

[0051] Turning again to the drawings, Figure 3 is a block diagram of a host 300 and data storage device 100 of an embodiment. The host 300 can take any suitable form, including, but not limited to, a computer, a mobile phone, a tablet, a wearable device, a digital video recorder, a surveillance system, etc. The host 300 in this embodiment (here, a computing device) comprises a processor 330 and a memory 340. In one embodiment, computer-readable program code stored in the host memory 340 configures the host processor 330 to perform the acts described herein. So, actions performed by the host 300 are sometimes referred to herein as being performed by an application (computer-readable program code) run on the host 300. For example, the host 300 can be configured to send data (e.g., initially stored in the host's memory 340) to the data storage device 100 for storage in the data storage device's memory 104.

[0052] As mentioned above, in some data storage devices, such as NVMe SSD devices, when a host is doing a read command, many control operations are involved on the host side, and many unused resources are allocated ahead of time in both the host and the data storage device. Figure 4 is a read-command flow diagram that illustrates this issue. As shown in Figure 4, the host allocates space for pointers (act 1) and allocates space for data buffers (act 2). Then, the host prepares the list of pointers that point to the buffers (act 3) and also prepares a command that includes pointers to a physical region page (PRP) list (act 4). Those items are stored in a queue in the host's volatile memory (e.g., DRAM). The host informs the data storage device that a command is ready in the queue by ringing a doorbell (e.g., setting a bit in a doorbell register) (act 5), and, later, when the data storage device is free, the data storage device fetches the read command from the queue in the host (act 6). The data storage device parses the command and fetches the PRP pointers (act 7) and stores the PRP pointers in a temporary memory (e.g., its local RAM) (act 8). Later, when the non-volatile memory (e.g., NAND) is not busy and can service the read command, the controller of the data storage device reads the data from the non-volatile memory and stores it in its local RAM (act 9). After data is error corrected (if needed), the controller reads the locally-stored PRP and calculates a host offset address (act 10), so that it can write the data to the buffer location in the host that is pointed to by the pointers provided in the PRP list (act 11). The data storage device then informs the host that the command is complete and that the data is ready for the host to read it the host's buffer in DRAM (act 12). In response, the host re-reads the PRP list to determine which buffer location to read (act 13) and then reads and uses the data it requested (act 14). At that point, the host can release the data

buffers (act 15) and release the PRP list (act 16), so the buffer locations pointed to by the pointers can be released and repurposed for another read command.

[0053] In this process, a temporary/random allocated memory of the PRP list and the empty read buffers are reserved. Only at act 11 does the memory allocated for the data buffers become available. Also, on the data storage device side, the local copy of the PRP list in the randomly-allocated memory is prepared ahead of time in act 8 and is held until act 10. This approach requires ahead-of-time buffer allocation done by the host. It also can require allocating the buffers ahead of time before using them. In order to support full throughput, more buffers are required since the life cycle of each buffer is longer. Also, as indicated by the fast-pace adoption in the industry of the zone-append command, the host prefers if the data storage device offloads as much management as possible (e.g., in zone-append, the logical block address (LBA) of the write commands).

[0054] The following embodiments can be used to offload work from the host 300 and move to a just-in-time memory allocation scheme. In one embodiment, the host 300 performs a read and gets the location of the data in the response. This will offload the management of PRP lists from the host and reduces the amount of time that randomly-allocated memory is allocated and stays empty. In addition, these embodiments can improve the latency of read commands and reduce the complexity related to PRP fetching in out-of-order data transfer modes.

[0055] Turning again to the drawings, Figure 5 is a block diagram of a host 300 and a controller 102 and non-volatile memory (e.g., NAND) 104 of a data storage device 100 of an embodiment. Additional or different components, including those mentioned above, can be used. As shown in Figure 5, in this embodiment, the controller 102 of the data storage device 100 comprises a PCIe host interface 130, a control path 110, and a data path 120 between the host 300 and the non-volatile memory 104. The controller path 110 comprises a processor 116 and receives commands from the host 300 and triggers the modules in the data path 120. The data path 120 comprises a flash interface module 128 to read data from the non-volatile memory 104, an error correction code module 126 to perform any required error correction, an encryption/decryption (XTS) module 124 to decrypt read data, a direct memory access (DMA) module 122 to write data to the host 300, and a resource server 700.

[0056] In this embodiment, the resource server 700 is configured (e.g., via computer readable-program code executed by the controller 102 and/or hardware) to manages the memory allocation in a host memory buffer (HMB) in the volatile memory 340 in the host 300. When the DMA module 122 wants to transfer data (and, later, a completion) to the host 300, the DMA module 122 requests a free memory range from the resource server 700. In this way, the

resource server 700 provides HMB memory management. This process is illustrated in HMB read flow diagram in Figure 6.

[0057] As shown in Figure 6, the host 300 fills-in a read command in a queue in the host's volatile memory (e.g. DRAM) (act 601) and informs the data storage device 100 that a command is ready by ringing a doorbell (act 602). Note, that the host 300 is not allocating space for a data buffer or pointer list because the host 300 is not managing its buffers in this embodiment. Later, when the data storage device 100 is free, the controller 102 fetches the read command from the queue in the host 300 (act 603). Later, when the non-volatile memory 104 is not busy and can service the read command, the controller 102 reads the data from the non-volatile memory 104 and stores the data in local volatile memory (e.g., RAM) in the data storage device 100 (act 604). Later, after correcting for any errors, the controller 102 selects free space in the host memory buffer (HMB) to store the data (e.g., the DMA module 122 and request the free space from the resource server 700) (act 605). The controller 102 then writes the data to the selected location in the HMB (act 606) and informs the host 300 that the command is completed and that the data is ready to be read from the HMB (act 607). The host 300 can then read and use the data it requested (act 608). Later, after data is no longer required, the host 300 can suggest to the data storage device 100 that the buffers can be released (act 609). In response, the controller 102 can mark the range as free (act 610).

[0058] Following the zoned namespace (ZNS) concept, the host 300 can be informed of the data location in the completion. For example, the completion queue entry layout shown in Figure 7 can be used, with command-specific fields (DW0, DW1) holding a pointer to the address of the data (e.g., the PRP list) in the host's DRAM. When the host 300 finishes reading the command, it can inform the data storage device 100 by providing the PRP list that was generated by the data storage device 100. The control path of the controller 102 can interact with the resource server 700 to release the buffers, so they can be reused for next commands.

[0059] There are several advantages associated with these embodiments. For example, with these embodiments, the host 300 can reduce a lot of overhead in buffer management. These embodiments also allow resources to be allocated "just in time" and (due to removing the PRP fetching phase) can reduce latency because fewer read transactions go over the PCIe interface. These embodiments also simplify out-of-order data transfer on read operations.

[0060] Finally, as mentioned above, any suitable type of memory can be used. Semiconductor memory devices include volatile memory devices, such as dynamic random access memory ("DRAM") or static random access memory ("SRAM") devices, non-volatile memory devices, such as resistive random access memory ("ReRAM"), electrically erasable programmable read only memory ("EEPROM"), flash memory (which can also be considered a subset of EEPROM),

ferroelectric random access memory ("FRAM"), and magnetoresistive random access memory ("MRAM"), and other semiconductor elements capable of storing information. Each type of memory device may have different configurations. For example, flash memory devices may be configured in a NAND or a NOR configuration.

[0061] The memory devices can be formed from passive and/or active elements, in any combinations. By way of non-limiting example, passive semiconductor memory elements include ReRAM device elements, which in some embodiments include a resistivity switching storage element, such as an anti-fuse, phase change material, etc., and optionally a steering element, such as a diode, etc. Further by way of non-limiting example, active semiconductor memory elements include EEPROM and flash memory device elements, which in some embodiments include elements containing a charge storage region, such as a floating gate, conductive nanoparticles, or a charge storage dielectric material.

[0062] Multiple memory elements may be configured so that they are connected in series or so that each element is individually accessible. By way of non-limiting example, flash memory devices in a NAND configuration (NAND memory) typically contain memory elements connected in series. A NAND memory array may be configured so that the array is composed of multiple strings of memory in which a string is composed of multiple memory elements sharing a single bit line and accessed as a group. Alternatively, memory elements may be configured so that each element is individually accessible, e.g., a NOR memory array. NAND and NOR memory configurations are examples, and memory elements may be otherwise configured.

[0063] The semiconductor memory elements located within and/or over a substrate may be arranged in two or three dimensions, such as a two-dimensional memory structure or a three dimensional memory structure.

[0064] In a two-dimensional memory structure, the semiconductor memory elements are arranged in a single plane or a single memory device level. Typically, in a two-dimensional memory structure, memory elements are arranged in a plane (e.g., in an x-z direction plane) which extends substantially parallel to a major surface of a substrate that supports the memory elements. The substrate may be a wafer over or in which the layer of the memory elements are formed or it may be a carrier substrate which is attached to the memory elements after they are formed. As a non-limiting example, the substrate may include a semiconductor such as silicon.

[0065] The memory elements may be arranged in the single memory device level in an ordered array, such as in a plurality of rows and/or columns. However, the memory elements may be arrayed in non-regular or non-orthogonal configurations. The memory elements may each have two or more electrodes or contact lines, such as bit lines and wordlines.

[0066] A three-dimensional memory array is arranged so that memory elements occupy multiple planes or multiple memory device levels, thereby forming a structure in three dimensions (i.e., in the x, y and z directions, where the y direction is substantially perpendicular and the x and z directions are substantially parallel to the major surface of the substrate).

[0067] As a non-limiting example, a three-dimensional memory structure may be vertically arranged as a stack of multiple two dimensional memory device levels. As another non-limiting example, a three dimensional memory array may be arranged as multiple vertical columns (e.g., columns extending substantially perpendicular to the major surface of the substrate, i.e., in the y direction) with each column having multiple memory elements in each column. The columns may be arranged in a two dimensional configuration, e.g., in an x-z plane, resulting in a three dimensional arrangement of memory elements with elements on multiple vertically stacked memory planes. Other configurations of memory elements in three dimensions can also constitute a three dimensional memory array.

[0068] By way of non-limiting example, in a three dimensional NAND memory array, the memory elements may be coupled together to form a NAND string within a single horizontal (e.g., x-z) memory device levels. Alternatively, the memory elements may be coupled together to form a vertical NAND string that traverses across multiple horizontal memory device levels. Other three dimensional configurations can be envisioned wherein some NAND strings contain memory elements in a single memory level while other strings contain memory elements which span through multiple memory levels. Three dimensional memory arrays may also be designed in a NOR configuration and in a ReRAM configuration.

[0069] Typically, in a monolithic three dimensional memory array, one or more memory device levels are formed above a single substrate. Optionally, the monolithic three dimensional memory array may also have one or more memory layers at least partially within the single substrate. As a non-limiting example, the substrate may include a semiconductor such as silicon. In a monolithic three dimensional array, the layers constituting each memory device level of the array are typically formed on the layers of the underlying memory device levels of the array. However, layers of adjacent memory device levels of a monolithic three dimensional memory array may be shared or have intervening layers between memory device levels.

[0070] Then again, two dimensional arrays may be formed separately and then packaged together to form a non-monolithic memory device having multiple layers of memory. For example, non-monolithic stacked memories can be constructed by forming memory levels on separate substrates and then stacking the memory levels atop each other. The substrates may be thinned or removed from the memory device levels before stacking, but as the memory device levels are initially formed over separate substrates, the resulting memory arrays are not

monolithic three dimensional memory arrays. Further, multiple two dimensional memory arrays or three dimensional memory arrays (monolithic or non-monolithic) may be formed on separate chips and then packaged together to form a stacked-chip memory device.

[0071] Associated circuitry is typically required for operation of the memory elements and for communication with the memory elements. As non-limiting examples, memory devices may have circuitry used for controlling and driving memory elements to accomplish functions such as programming and reading. This associated circuitry may be on the same substrate as the memory elements and/or on a separate substrate. For example, a controller for memory read-write operations may be located on a separate controller chip and/or on the same substrate as the memory elements.

[0072] One of skill in the art will recognize that this invention is not limited to the two dimensional and three-dimensional structures described but cover all relevant memory structures within the spirit and scope of the invention as described herein and as understood by one of skill in the art.

[0073] It is intended that the foregoing detailed description be understood as an illustration of selected forms that the invention can take and not as a definition of the invention. It is only the following claims, including all equivalents, that are intended to define the scope of the claimed invention. Finally, it should be noted that any aspect of any of the embodiments described herein can be used alone or in combination with one another.

What is claimed is:

1. A data storage device comprising:

a non-volatile memory; and

a controller configured to:

receive a read command from a host;

read data from the non-volatile memory;

identify a location in a host memory buffer (HMB) in the host that is available to store the data;

write the data to the location in the HMB; and

inform the host of the location in the HMB that stores the data.

2. The data storage device of Claim 1, wherein the controller, not the host, is responsible for managing where the data is written in the host.

3. The data storage device of Claim 1, wherein the controller comprises:

a resource server configured to manage memory allocation in the HMB; and

a direct memory access module configured to send a request to the resource server to identify the location in the HMB that is available to store the data.

4. The data storage device of Claim 1, wherein the controller is further configured to inform the host of the location in the HMB by storing a completion queue entry in a completion queue, wherein the completion queue entry comprises a pointer to the location.

5. The data storage device of Claim 1, wherein the controller is further configured to receive, from the host, an indication that the location in the HMB can be released.

6. The data storage device of Claim 5, wherein the controller is further configured to release the location in response to receiving the indication.

7. The data storage device of Claim 5, wherein:

the controller is further configured to inform the host of the location in the HMB by sending a pointer to the host; and

receiving the indication from the host comprises receiving the pointer back from the host.

8. The data storage device of Claim 1, wherein the controller is further configured to receive the read command by fetching the read command from a submission queue in the host in response to determining a doorbell register indicates that the read command is available to be fetched from the submission queue.

9. The data storage device of Claim 1, further comprising a volatile memory, wherein the controller is further configured to store the data in the volatile memory after the data is read from the non-volatile memory but before the data is written to the location in the HMB.

10. The data storage device of Claim 1, wherein the controller is further configured to error correct the data prior to writing the data to the location in the HMB.

11. The data storage device of Claim 1, wherein the data storage device comprises a solid-state drive (SSD).

12. The data storage device of Claim 1, wherein the non-volatile memory comprises a three-dimensional memory.

13. A method comprising:

performing the following in a host in communication with a data storage device:

generating a request for data stored in the data storage device;

receiving, from the data storage device, a pointer to a location in a memory in the host where the data storage device stored the data after executing the request;

reading the data from the location in the memory; and

informing the data storage device that the location in the memory can be released.

14. The method of Claim 13, wherein the data storage device, not the host, is responsible for managing where the data is written in the host.

15. The method of Claim 14, wherein a resource server in the data storage device is responsible for managing where the data is written in the host.

16. The method of Claim 13, wherein the memory in the host comprises a host memory buffer (HMB).

17. The method of Claim 13, wherein the pointer is received in a completion queue entry written in a completion queue by the data storage device.

18. The method of Claim 13, wherein informing the data storage device that the location in the memory can be released comprises sending the pointer back to the data storage device.

19. The method of Claim 13, further comprising:

placing the read command in a submission queue; and

informing the data storage device that the read command is ready to be fetched from the submission queue.

20. A data storage device comprising:

a memory; and

means for responding to a read command from a host by:

reading data from the memory;

selecting a location in host memory that is available to store the data;

writing the data to the location in the host memory; and

informing the host of the location in the host memory that stores the data.

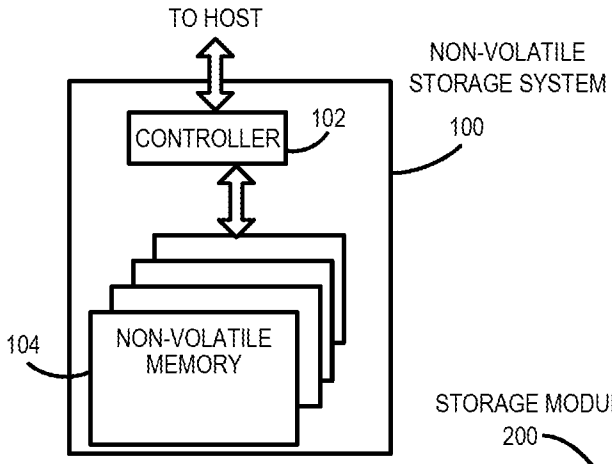


FIG. 1A

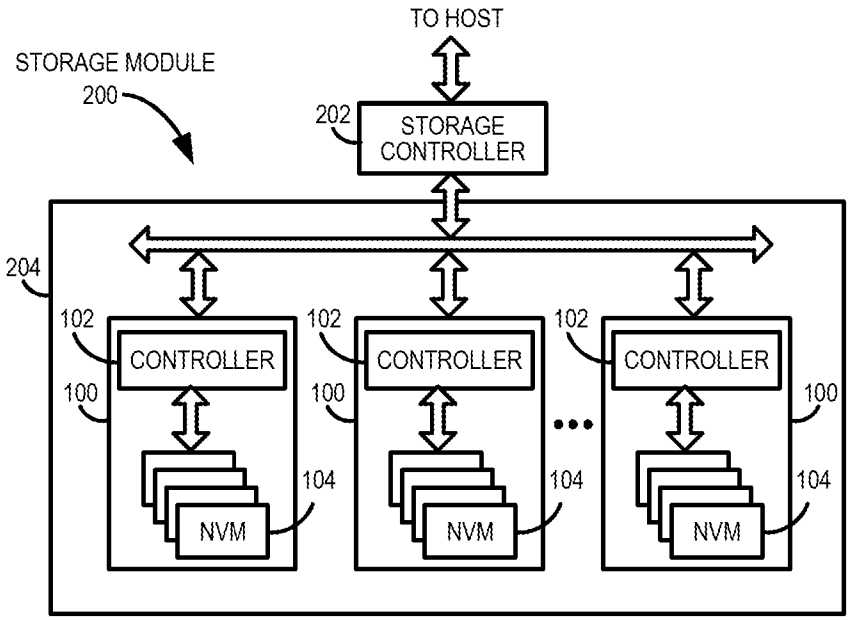


FIG. 1B

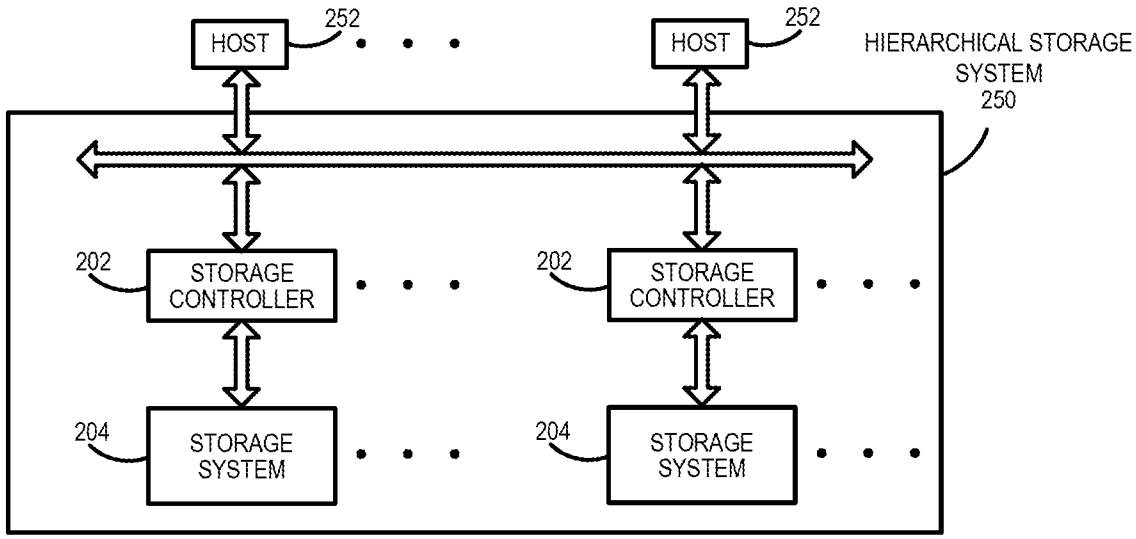


FIG. 1C

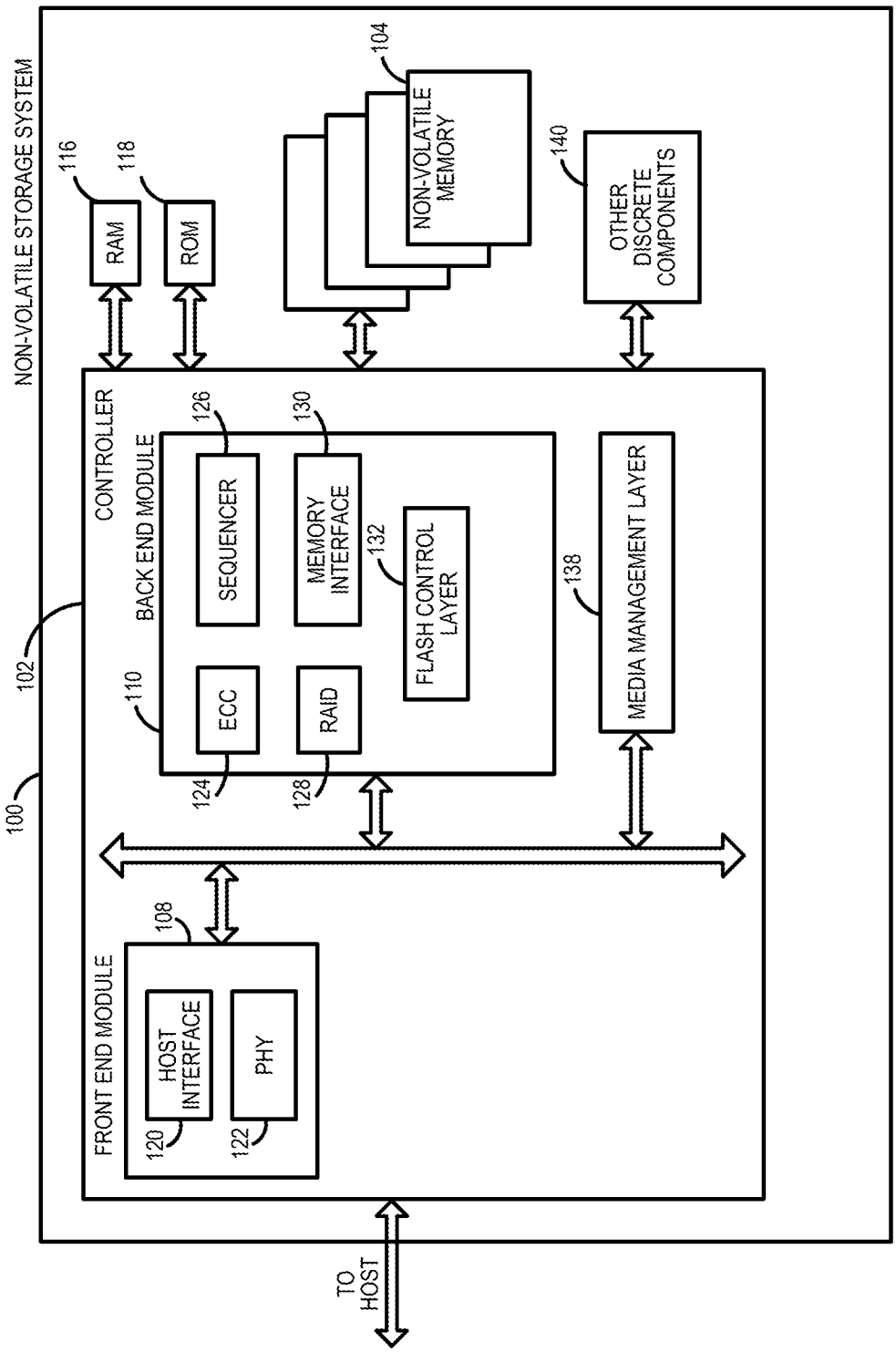

NON-VOLATILE STORAGE SYSTEM
100
102
104
NON-VOLATILE MEMORY
116
RAM
118
ROM
140
OTHER DISCRETE COMPONENTS
CONTROLLER
BACK END MODULE
110
126
SEQUENCER
130
MEMORY INTERFACE
132
FLASH CONTROL LAYER
138
MEDIA MANAGEMENT LAYER
ECC
124
RAID
128
FRONT END MODULE
108
HOST INTERFACE
120
PHY
122
TO HOST

FIG. 2A

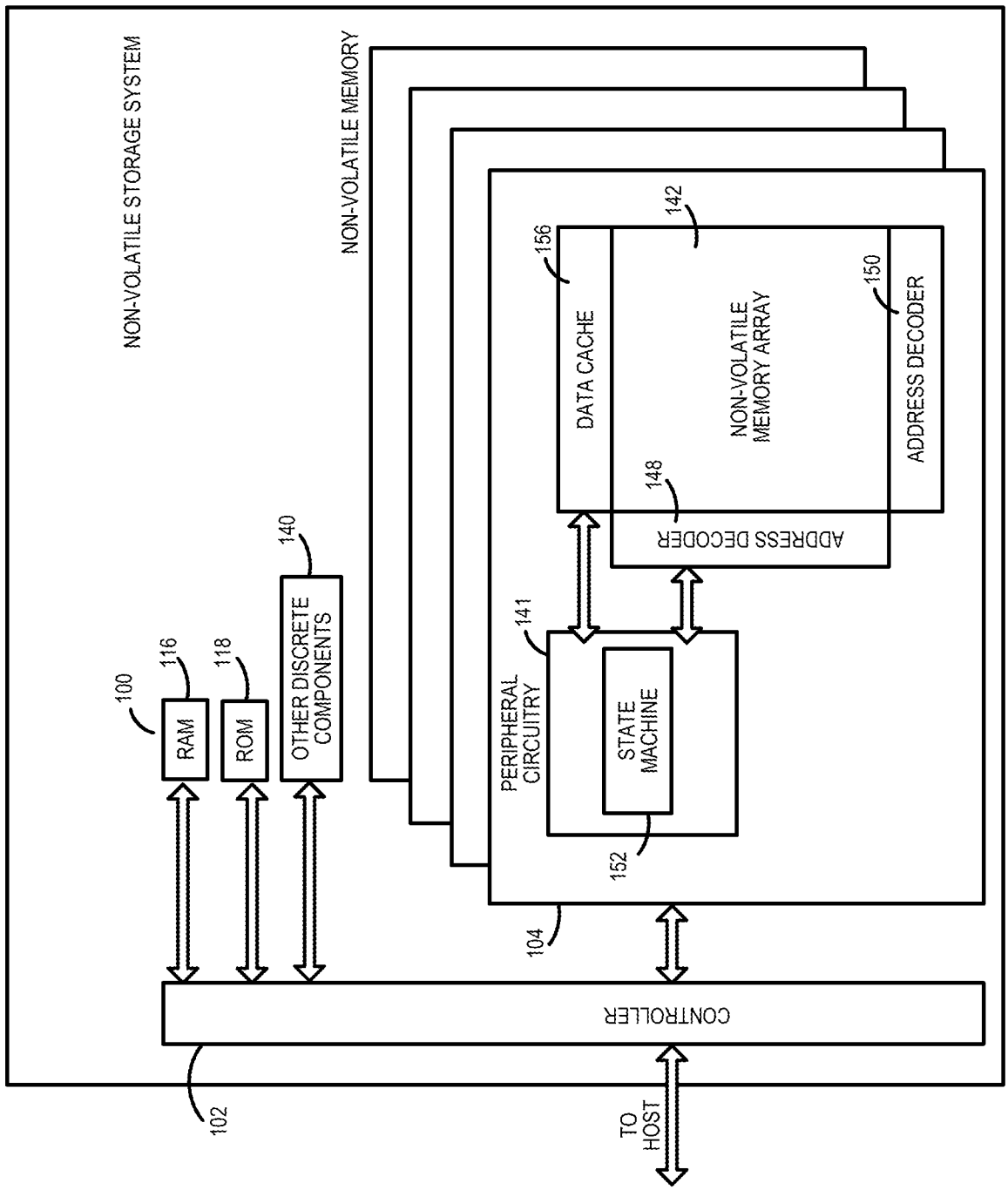

NON-VOLATILE STORAGE SYSTEM
NON-VOLATILE MEMORY
142
156
DATA CACHE
NON-VOLATILE MEMORY ARRAY
150
ADDRESS DECODER
148
ADDRESS DECODER
140
OTHER DISCRETE COMPONENTS
116
RAM
118
ROM
100
141
PERIPHERAL CIRCUITRY
STATE MACHINE
152
104
CONTROLLER
102
TO HOST

FIG. 2B

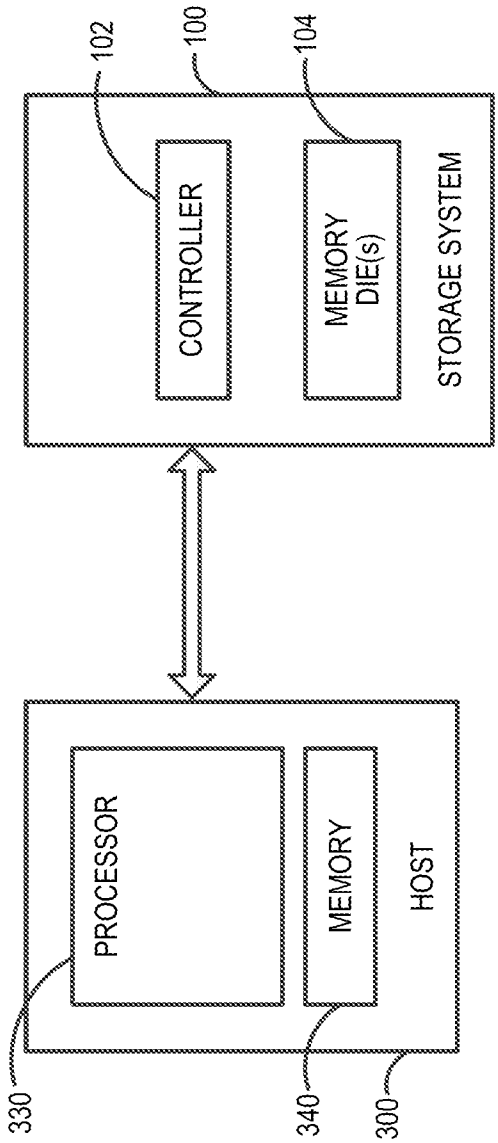

102
100
104
CONTROLLER
MEMORY DIE(s)
STORAGE SYSTEM
PROCESSOR
MEMORY
HOST
330
340
300

FIG. 3

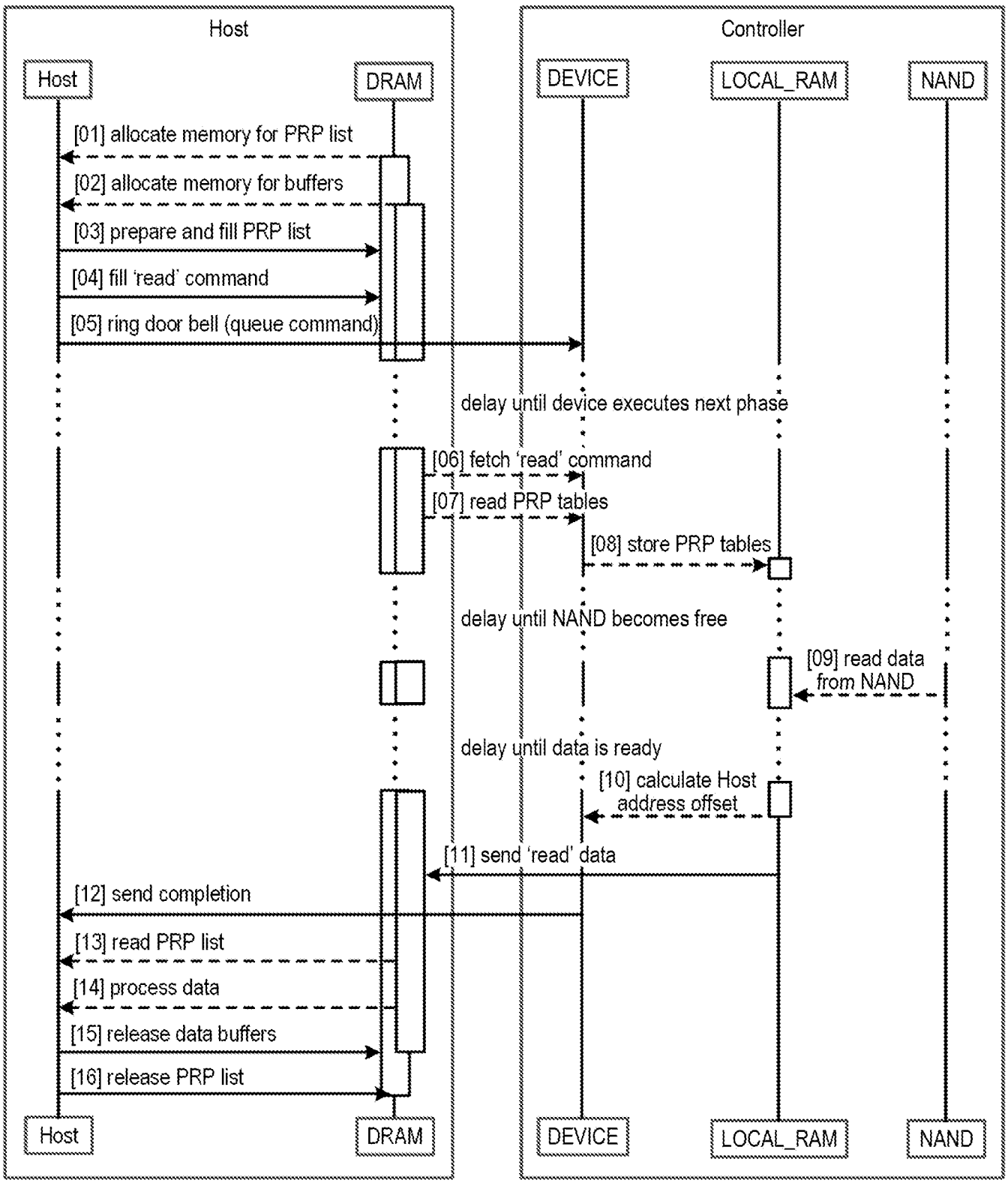

Host
Controller
Host
DRAM
DEVICE
LOCAL_RAM
NAND
[01] allocate memory for PRP list
[02] allocate memory for buffers
[03] prepare and fill PRP list
[04] fill 'read' command
[05] ring door bell (queue command)
delay until device executes next phase
[06] fetch 'read' command
[07] read PRP tables
[08] store PRP tables
delay until NAND becomes free
[09] read data from NAND
delay until data is ready
[10] calculate Host address offset
[11] send 'read' data
[12] send completion
[13] read PRP list
[14] process data
[15] release data buffers
[16] release PRP list
Host
DRAM
DEVICE
LOCAL_RAM
NAND

FIG. 4

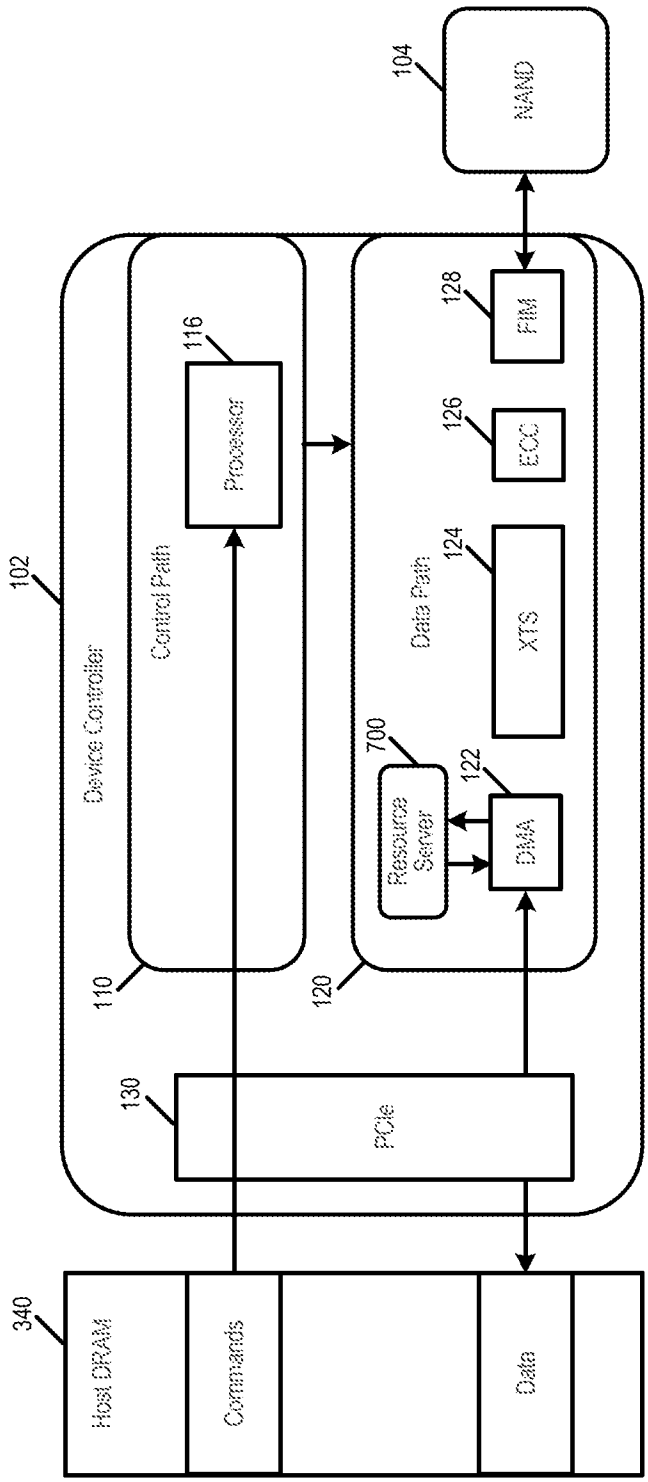

104
NAND
102
Device Controller
Control Path
116
Processor
110
Data Path
128
FIM
126
ECC
124
XTS
700
Resource Server
122
DMA
120
130
PCIe
340
Host DRAM
Commands
Data

FIG. 5

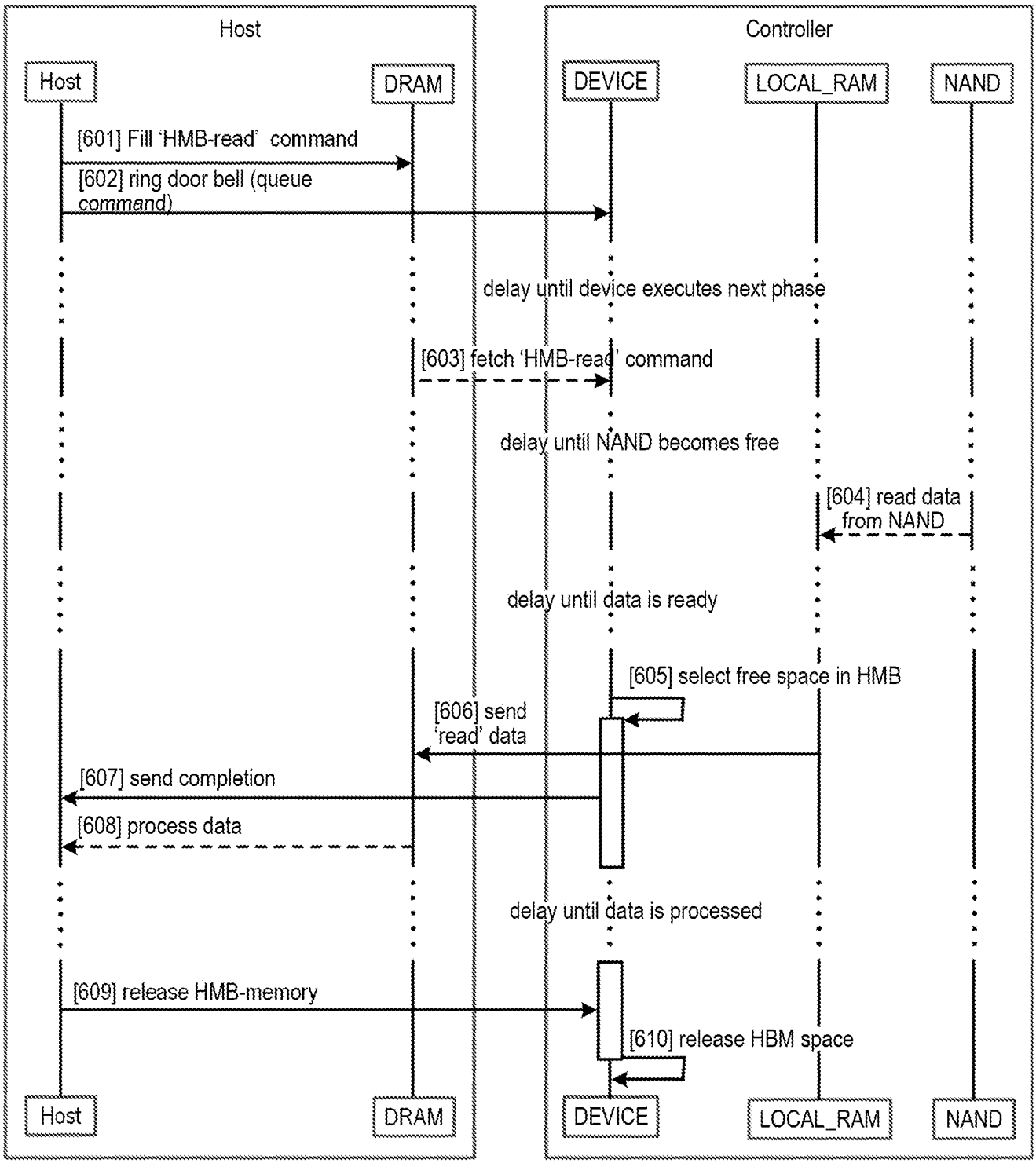

Host
Controller
Host
DRAM
DEVICE
LOCAL_RAM
NAND
[601] Fill 'HMB-read' command
[602] ring door bell (queue command)
delay until device executes next phase
[603] fetch 'HMB-read' command
delay until NAND becomes free
[604] read data from NAND
delay until data is ready
[605] select free space in HMB
[606] send 'read' data
[607] send completion
[608] process data
delay until data is processed
[609] release HMB-memory
[610] release HBM space
Host
DRAM
DEVICE
LOCAL_RAM
NAND

FIG. 6

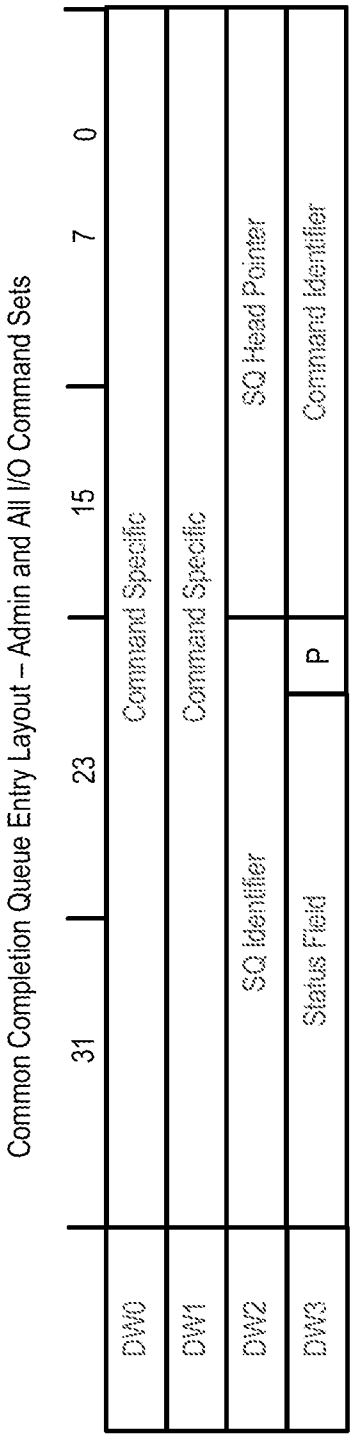

Common Completion Queue Entry Layout – Admin and All I/O Command Sets
31
23
15
7
0
DW0
Command Specific
DW1
Command Specific
DW2
SQ Identifier
SQ Head Pointer
DW3
Status Field
P
Command Identifier

FIG. 7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2023/021373

A. CLASSIFICATION OF SUBJECT MATTER

G06F 3/06(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F 3/06(2006.01); G06F 11/07(2006.01); G06F 12/02(2006.01); G06F 13/28(2006.01); G11C 11/56(2006.01)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

eKOMPASS(KIPO internal) & Keywords: host memory buffer (HMB), read command, controller, pointer, doorbell register, queue, location

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2016-0011964 A1 (SANDISK TECHNOLOGIES INC.) 14 January 2016 (2016-01-14) paragraphs [0012]-[0016], [0022]-[0024], [0043], [0051]-[0058]; claims 48, 52; and figures 1-2	1-4,9-12,20
Y		5-8,13-19
Y	US 2019-0294350 A1 (WESTERN DIGITAL TECHNOLOGIES, INC.) 26 September 2019 (2019-09-26) paragraph [0072]; claim 17; and figure 2A	5-8,13-19
A	US 2020-0151040 A1 (SAMSUNG ELECTRONICS CO., LTD.) 14 May 2020 (2020-05-14) paragraphs [0020]-[0072]; and figures 1-8	1-20
A	US 2018-0260347 A1 (WESTERN DIGITAL TECHNOLOGIES, INC.) 13 September 2018 (2018-09-13) paragraphs [0021]-[0120]; and figures 1A-6	1-20

☑ Further documents are listed in the continuation of Box C. ☑ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"D" document cited by the applicant in the international application
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
23 August 2023	**24 August 2023**

Name and mailing address of the ISA/KR	Authorized officer
Korean Intellectual Property Office **189 Cheongsa-ro, Seo-gu, Daejeon** **35208, Republic of Korea**	**BYUN, SUNG CHEAL**
Facsimile No. **+82-42-481-8578**	Telephone No. **+82-42-481-8262**

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CN 107665091 B (SHENZHEN DAXIN ELECTRONIC TECHNOLOGY CO., LTD.) 02 March 2021 (2021-03-02) paragraphs [0045]-[0134]; and figures 1-4	1-20

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2023/021373

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
US 2016-0011964 A1	14 January 2016	US 10824335 B2	03 November 2020
US 2019-0294350 A1	26 September 2019	US 10613778 B2	07 April 2020
		WO 2019-182654 A1	26 September 2019
US 2020-0151040 A1	14 May 2020	CN 111190841 A	22 May 2020
		KR 10-2020-0056538 A	25 May 2020
		US 11011243 B2	18 May 2021
		US 11501843 B2	15 November 2022
		US 2021-0257034 A1	19 August 2021
		US 2023-0020537 A1	19 January 2023
US 2018-0260347 A1	13 September 2018	CN 108572933 A	25 September 2018
		CN 108572933 B	19 February 2021
		US 10275378 B2	30 April 2019
		WO 2018-164741 A1	13 September 2018
CN 107665091 B	02 March 2021	CN 107665091 A	06 February 2018