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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
4 August 2016 (04.08.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/122463 A1

(51) International Patent Classification:
G06F 11/10 (2006.01) *G06F 12/00* (2006.01)

(21) International Application Number:
PCT/US2015/013124

(22) International Filing Date:
27 January 2015 (27.01.2015)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **HEWLETT PACKARD ENTERPRISE DEVELOPMENT LP** [US/US]; 11445 Compaq Center Drive West, Houston, TX 77070 (US).

(72) Inventors: **WARNES, Lidia**; 8000 Foothills Blvd., Roseville, California 95747 (US). **BENEDICT, Melvin K.**; 11445 Compaq Center Dr. W., Houston, Texas 77070 (US).

(74) Agents: **SHOOKMAN, Jeb A.** et al.; Hewlett Packard Enterprise, 3404 E. Harmony Road, Mail Stop 79, Fort Collins, CO 80528 (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, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, 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, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, 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, 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).

Declarations under Rule 4.17:

— *as to the identity of the inventor (Rule 4.17(i))*

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

[Continued on next page]

(54) **Title**: CORRECTING ERRORS OF A VARIABLE DATA PAYLOAD

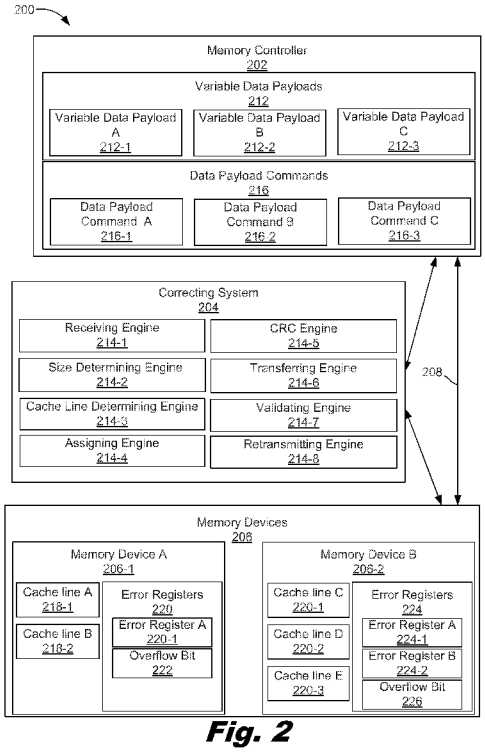

Fig. 2

(57) **Abstract**: Correcting errors of a variable data payload includes receiving a data payload command to transfer a variable data payload from a memory controller to a memory device, determining, based on a size of the variable data payload, a number of cache lines for the variable data payload, assigning the variable data payload to the number of cache lines, transferring each of the cache lines with a cyclic redundancy check (CRC) character from the memory controller to the memory device to create a transfer, and validating, based on the CRC character, each of the cache lines of the transfer.

WO 2016/122463 A1

Published:

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

CORRECTING ERRORS OF A VARIABLE DATA PAYLOAD

BACKGROUND

[0001] A memory controller is a digital circuit which manages the flow of data payloads to and from memory devices of a computing system. The flow may be executed by a memory command cycle. The memory command cycle may include a number of commands such as a data payload command, an activate command, and a read command or a write command. The read command may read data payloads from the memory devices. The write command may write the data payloads to the memory devices. The memory devices may be dynamic random-access memory (DRAM), synchronous dynamic random-access memory (SDRAM), non-volatile memory such as flash memory, other memory devices, or combinations thereof.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] The accompanying drawings illustrate various examples of the principles described herein and are a part of the specification. The examples do not limit the scope of the claims.

[0003] Fig. 1 is a diagram of a system for correcting errors of a variable data payload, according to one example of principles described herein.

[0004] Fig. 2 is a diagram of a system for correcting errors of a variable data payload, according to one example of principles described herein.

[0005] Fig. 3 is a flowchart of a method for correcting errors of a variable data payload, according to one example of principles described herein.

[0006] Fig. 4 is a flowchart of a method for correcting errors of a variable data payload, according to one example of principles described herein.

[0007] Fig. 5 is a diagram of a correcting system, according to one example of principles described herein.

[0008] Fig. 6 is a diagram of a correcting system, according to one example of principles described herein.

[0009] Throughout the drawings, identical reference numbers designate similar, but not necessarily identical, elements.

DETAILED DESCRIPTION

[0010] As mentioned above, a memory controller is a digital circuit which manages the flow of data payloads to and from memory devices. Further, the data payloads may be associated with a size. The size may be in terms of bytes. If the size of a data payload is sixty-four bytes, the memory controller may read or write the sixty-four bytes to or from the memory devices.

[0011] Often, the size of the data payloads transferred to and from the memory devices may vary in size. Further, memory controllers limit the number of bytes transferred for each data payload based on a read command or a write command to a fixed number of bytes, such as sixty-four bytes. Transferring larger amounts of data payloads increases the demand on the memory controller and the memory devices. Further, the memory controller may be optimized for transferring a burst length of the data payload or a block of the data payload. If the memory controller transfers a burst length of the data payload to a memory device optimized for blocks of the data payload, the memory controller may experience delays. As a result, the data payload is not transferred to and from the memory devices at an optimal rate.

[0012] Further, errors may be introduced into the data payload while transferring the data payload to and from memory devices. When an error is detected in the data payload, the entire data payload is retransmitted to correct the error. If the data payload is large in size, this can negatively impact the performance of the memory controller and the memory devices.

[0013] The principles described herein include a method for correcting errors of a variable data payload. Such a method includes receiving a data payload command to transfer a variable data payload from a memory controller to a memory device, determining, based on a size of the variable data payload, a number of cache lines for the variable data payload, assigning the variable data payload to the number of cache lines, transferring each of the cache lines with a cyclic redundancy check (CRC) character from the memory controller to the memory device to create a transfer, and validating, based on the CRC character, each of the cache lines of the transfer. Such a method allows the transfer of variable payloads between the memory controller and memory device to enable multiple memory technologies to be attached to the same memory controller as long as they are using the same protocol. As a result, the method minimizes the retransmission of the variable data payload by retransmitting cache lines associated with errors instead of the entire variable data payload.

[0014] In the present specification and in the appended claims, the term "data payload command" means a mechanism to control the transfer of a variable data payload between a memory controller and a memory device. The data payload command may include a start address for transferring each of the cache lines of the variable data payload to the memory device, a length of the variable data payload, a termination character associated with the variable data payload, or combinations thereof.

[0015] In the present specification and in the appended claims, the term "variable data payload" means information transferred between a memory controller and a memory device. The variable data payload may be associated with a size, the size varying depending on the amount of information associated with each variable data payload. The size of the variable data payload may be sent to the memory devices.

[0016] In the present specification and in the appended claims, the term "memory controller" means is a digital circuit which manages the flow of variable data payloads to and from at least one memory device. The flow may be executed by a memory command cycle. The memory command cycle may

include a number of commands such as a data payload command, an activate command, and a read command or a write command. The read command may read data payloads from the memory devices. The write command may write the data payloads to the memory devices.

[0017] In the present specification and in the appended claims, the term "memory device" means a mechanism to store a variable data payload as cache lines in memory. The memory devices may include dynamic random-access memory (DRAM), synchronous dynamic random-access memory (SDRAM), non-volatile memory such as flash memory, other memory devices or combinations thereof. Further, the memory devices may include a number of error registers and an overflow bit.

[0018] In the present specification and in the appended claims, the term "cache line" means a portion of the variable data payload that is transferred to a memory device. The cache line is fixed in size and ranges from 16 to 256 bytes.

[0019] In the present specification and in the appended claims, the term "CRC Character" means a mechanism used to detect accidental changes made to a cache line while transferring the cache line from a memory controller to a memory device. A CRC character may be a short, fixed-length binary sequence that is concatenated to a cache line before the cache line is transferred to the memory device.

[0020] In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present systems and methods. It will be apparent, however, to one skilled in the art that the present apparatus, systems, and methods may be practiced without these specific details. Reference in the specification to "an example" or similar language means that a particular feature, structure, or characteristic described in connection with that example is included as described, but may not be included in other examples.

[0021] Fig. 1 is a diagram of a system for correcting errors of a variable data payload, according to one example of principles described herein. As will be described below, a correcting system is in communication with a

memory controller and memory devices to receive a data payload command to transfer a variable data payload from a memory controller to a memory device. The correcting system determines, based on a size of the variable data payload, a number of cache lines for the variable data payload. Further, the correcting system assigns the variable data payload to the number of cache lines. The correcting system transfers each of the cache lines with a CRC character from the memory controller to the memory device to create a transfer. Further, the correcting system validates, based on the CRC character, each of the cache lines of the transfer.

[0022] As illustrated, the system (100) includes a memory controller (102). The memory controller (102) may be a digital circuit which manages the flow of variable data payloads to and from memory devices (106). The flow may be executed by a memory command cycle. Further, the variable payload data may be transferred via a bus (108). The bus (108) may include a command/address (C/A) bus and a data bus. More information about the memory controller (102) will be described in other parts of this specification.

[0023] The system (100) further includes the memory devices (106). The memory devices (106) may include a number of types of memory technologies as long as the memory devices (106) use the same protocol. Further, the memory devices (106) may be attached to the memory controller (102) via the bus (108). More information about the memory devices (106) will be described in other parts of this specification.

[0024] The system (100) further includes a correcting system (104). The correcting system (104) is in communication with the memory controller (102) and the memory devices (106) to receive a data payload command to transfer a variable data payload from the memory controller (102) to the memory devices (106).

[0025] Further, the correcting system (104) determines, based on a size of the variable data payload, a number of cache lines for the variable data payload. The size of the variable data payload may be sent to the memory devices (106) or a memory buffer depending on the architecture of the memory devices (106).

[0026] The correcting system (104) further assigns the variable data payload to the number of cache lines. If the correcting system (104) determines five cache lines are needed for the variable data payload, portions of the variable data payload are assigned to the five cache lines.

[0027] Further, the correcting system (104) transfers each of the cache lines with a CRC character from the memory controller (102) to the memory devices (106) to create a transfer. If the correcting system (104) determines five cache lines are needed for the variable data payload, five cache lines are transferred from the memory controller (102) to the memory devices (106) to create the transfer.

[0028] Further, the correcting system (104) validates, based on the CRC character, each of the cache lines of the transfer. The correcting system (104) validates, based on the CRC character, each of the cache lines of the transfer via a validating engine (114). Such a system (100) allows the transfer of variable payloads between the memory controller (102) and memory devices (106) to enable multiple memory technologies to be attached to the memory controller (102) as long as they are using the same protocol. As a result, the system (100) minimizes the retransmission of the variable data payload by retransmitting cache lines associated with errors instead of the entire variable data payload. More information about the correcting system (106) will be described later on in this specification.

[0029] While this example has been described with reference to the correcting system being located between the memory controller and the memory devices, the correcting system may be located in any appropriate location according to the principles described herein. For example, the correcting system may be located on the memory controller, the memory devices, other locations, or combinations thereof.

[0030] Fig. 2 is a diagram of a system for correcting errors of a variable data payload, according to one example of principles described herein. As mentioned above, a correcting system is in communication with a memory controller and memory devices to receive a data payload command to transfer a variable data payload from a memory controller to a memory device. The

correcting system determines, based on a size of the variable data payload, a number of cache lines for the variable data payload. Further, the correcting system assigns the variable data payload to the number of cache lines. The correcting system transfers each of the cache lines with a CRC character from the memory controller to the memory device to create a transfer. Further, the correcting system validates, based on the CRC character, each of the cache lines of the transfer.

[0031] As illustrated, the system (200) includes a memory controller (202). As mentioned above, the memory controller (202) may be a digital circuit which manages the flow of a variable data payload to and from memory devices (206) of the system (200). The flow may be executed by a memory command cycle. The memory command cycle may include a number of commands such as a data payload command, an activate command, and a read command or a write command. The read command may read the variable data payload from the memory devices (206). The variable data payload may be read from the memory devices (206) by the memory controller (202) via a bus (208). As mentioned above, the bus (208) may include a C/A bus and a data bus. The C/A bus may transfer commands between the memory controller (202) and the memory devices (206). The data bus may transfer variable data payloads between the memory controller (202) and the memory devices (206). Further, the write command may write the variable data payload to the memory devices (206). The variable data payload may be written to the memory devices (206) by the memory controller (202) via the bus (208). As a result, the variable payload data may be transferred based on the memory command cycle, via the bus (208).

[0032] As illustrated, the memory controller (202) may include variable data payloads (212). The variable data payloads (212) may be information that is to be transferred between the memory controller (202) and the memory devices (206). In Fig. 2, the variable data payloads (212) includes variable data payload A (212-1), variable data payload B (212-2), and variable data payload C (212-3). The variable data payload may be associated with a size. The size of the variable data payloads (212) may vary. For example, variable data payload

A (212-1) may be one-hundred twenty eight bytes. Variable data payload B (212-2) may be one-hundred ninety-two bytes. Variable data payload C (212-3) may be thirty-two bytes. Further, the size of the variable data payload may be sent to the memory devices (206).

[0033] As illustrated, the memory controller (202) may include a number of data payload commands (216). The data payload commands (216) may be a mechanism to control the transfer of the variable data payloads (212) between the memory controller (202) and the memory devices (206). As illustrated, the data payload commands (216) include data payload command A (216-1), data payload command B (216-2), and data payload command C (216-3). Data payload command A (216-1) may include a write command to write variable data payload A (212-1) to memory device A (206-1). Data payload command B (216-2) may include a write command to write variable data payload B (212-2) to memory device B (206-2). Data payload command C (216-3) may include a write command to write variable data payload C (212-3) to memory device B (206-2).

[0034] Further, the data payload commands (216) may include a start address for transferring each of the cache lines to the memory devices (206). For example, data payload command A (216-1) may indicate to transfer variable data payload A (212-1) starting at address of 0x002F of memory device A (206-1). As a result, a first cache line associated with variable data payload A (212-1) is transferred to the start address of 0x002F for memory device A (206-1).

[0035] Further, the data payload commands (216) may further include a length of the variable data payloads (212). The length of the variable data payloads (212) may be representative of the size of the variable data payloads (212). The length of the variable data payloads (212) may be represented in hexadecimal, binary, other representations, or combinations thereof.

[0036] Further, the data payload commands (216) may include a termination character associated with the variable data payloads (212). The termination character may be a special character that allows the correcting system (204) to determine the size of each of the variable data payloads (212).

[0037] The system (200) further includes the memory devices (206). The memory devices (206) may include a number of types of memory technologies as long as the memory devices (206) use the same protocol. The types of memory devices (206) may include dynamic random-access memory (DRAM), synchronous dynamic random-access memory (SDRAM), non-volatile memory such as flash memory, other memory devices, or combinations thereof. As a result, some of the memory devices (206) may be optimized for variable data payloads that are transferred in burst length while some of the memory devices (206) may be optimized for variable data payloads that are transferred in blocks.

[0038] As illustrated, the memory devices (206) include memory device A (206-1) and memory device B (206-2). As will be described below, memory device A (206-1) stores, in memory, cache lines (218) associated with variable data payload A (212-1). Further, memory device B (206-2) stores, in memory, cache lines (220) associated with variable data payload B (212-1).

[0039] Further, the memory devices (206) may include error registers (220, 224). As will be described in other parts of this specification, the error registers (220, 224) may store errors associated with cache lines transferred from the memory controller (202) to the memory devices (206). Further, the error registers may include overflow bits (222, 226). The overflow bits (222, 226) indicate when the error registers (220, 224) are full. More information about the error registers and the overflow bits will be described in other parts of this specification.

[0040] The system (200) further includes a correcting system (204). The correcting system (204). In one example, the correcting system (204) includes a processor and computer program code. The computer program code is communicatively coupled to the processor. The computer program code includes a number of engines (214). The engines (214) refer to program instructions to perform a designated function. The computer program code causes the processor to execute the designated function of the engines (214). In other examples, the engines (214) refer to a combination of hardware and program instructions to perform a designated function. Each of the engines

(214) may include a processor and memory. The program instructions are stored in the memory and cause the processor to execute the designated function of the engine. As illustrated, the correcting system (204) includes a receiving engine (214-1), a size determining engine (214-2), a cache line determining engine (214-3), an assigning engine (214-4), a CRC engine (214-5), a transferring engine (214-6), a validating engine (214-7), and a retransmitting engine (214-8).

[0041] The receiving engine (214-1) receives a data payload command to transfer a variable data payload from the memory controller (202) to the memory device (206). The receiving engine (214-1) may receive data payload command A (216-1) to transfer variable data payload A (212-1) from the memory controller (202) to memory device A (206-1). The receiving engine (214-1) may further receive data payload command B (216-1) to transfer variable data payload B (212-2) from the memory controller (202) to memory device B (206-2). Still further, the receiving engine (214-1) may receive data payload command C (216-3) to transfer variable data payload C (212-2) from the memory controller (202) to memory device B (206-2).

[0042] The size determining engine (214-2) determines a size of the variable data payload. As described above, variable data payload A (212-1) may be determined to be one-hundred twenty eight bytes. Variable data payload B (212-2) may be determined to be one-hundred ninety-two bytes. Variable data payload C (212-3) may be thirty-two bytes.

[0043] The cache line determining engine (214-3) determines, based on the size of the variable data payload, a number of cache lines for the variable data payload. If a cache line is sixty-four bytes, the cache line determining engine (214-3) determines two cache lines are needed to transfer variable data payload A (212-1) from the memory controller (202) to memory device A (206-1). Similarly, the cache line determining engine (214-3) determines three cache line is needed for variable data payload B (212-2).

[0044] As mentioned above, the correcting system (204) includes an assigning engine (214-4). The assigning engine (214-4) assigns the variable data payload to the number of cache lines. For example, the assigning engine

(214-4) assigns variable data payload A (212-1) to two cache lines. As will be described below, the two cache lines for variable data payload A (212-1) are transferred to memory device A (206-1) as cache line A (218-1) and cache line B (218-2). Further, variable data payload B's three cache lines are transferred to memory device B (206-2) as cache line C (220-1), cache line D (220-2), and cache line E (220-3).

[0045] The CRC engine (214-5) determines a CRC character for each of the cache lines (218, 220). A CRC character may be a mechanism used to detect accidental changes made to a cache line while transferring the cache line from the memory controller (202) and the memory devices (206). The CRC character may be a short, fixed-length binary sequence that is concatenated to a cache line. The CRC engine (214-5) may determine the CRC characters based on various methods and techniques.

[0046] The transferring engine (214-6) transfers each of the cache lines with a CRC character from the memory controller (202) to the memory devices (206). Variable data payload A (212-1) is transferred to memory device A (206-1) as cache line A (218-1) and cache line B (218-2). Further, cache line A (218-1) and cache line B (218-2) may include CRC characters.

[0047] Further, variable data payload B (212-2) is transferred to memory device B (206-2) as cache line C (220-1), cache line D (220-2), and cache line E (220-3). Cache line C (220-1), cache line D (220-2), and cache line E (220-3) may include CRC characters.

[0048] The validating engine (214-7) validates, based on the CRC character, each of the cache lines of the transfer. The validating engine (214-7) validates, based on the CRC character, each of the cache lines of the transfer by obtaining the CRC character associated with each of the cache lines before the transfer and obtaining the CRC character associated with each of the cache lines after the transfer. For example, the validating engine (214-7) obtains the CRC character associated with cache line A (218-1) before the transfer and obtains the CRC character associated with cache line A (218-1) after the transfer. Further, the validating engine (214-7) obtains the CRC character

associated with cache line B (218-2) before the transfer and obtains the CRC character associated with cache line B (218-2) after the transfer.

[0049] The validating engine (214-7) further validates, based on the CRC character, each of the cache lines of the transfer by comparing the CRC characters for each of the cache lines to determine if an error has occurred while transferring the cache lines from the memory controller (202) to the memory device (206). Further, the validating engine (214-7) stores, based on overflow bits (222, 226) of error registers (220, 224), an address of the cache line associated with the error in at least one of the error registers (220, 224) of the memory devices (206). For example, the validating engine (214-7) compares the CRC characters for cache line A (218-1) before and after the transfer. If the CRC characters for cache line A (218-1) match, an error did not occur during the transfer. However, if the CRC characters for cache line A (218-1) do not match, an error did occur during the transfer. As a result, the address of cache line A (218-1) is stored in error register A (220-1). Further, the validating engine (214-7) compares the CRC characters for cache line B (218-2). If the CRC characters for cache line B (218-2) match an error did not occur during the transfer. However, if the CRC characters for cache line B (218-2) do not match an error did occur during the transfer. As a result, since the error registers (220) are full, the overflow bit (222) is set.

[0050] The retransmitting engine (214-8) retransmits at least one of the cache lines to correct errors associated with the transfer. The retransmitting engine (214-8) retransmits at least one of the cache lines to correct the errors associated with the transfer by determining if the overflow bit (222, 226) of error registers (220, 224) is set. An overflow bit (222, 226) of 0 indicates the overflow bit (222, 226) is not set. An overflow bit (222, 226) of 1 indicates the overflow bit (222, 226) is set. Further, the overflow bit (222, 226) of the error registers (220, 224) is set when an error threshold is exceeded. The error threshold may correspond to the number of error registers of the memory device. For example, memory device A (206-1) includes one error register, error register A (220-1). As a result, the error threshold for memory device A (206-1) is one. If the variable data payload is received and the overflow bit (222, 226) has not

been set, then the memory controller (202) sets the data payload size to thirty two bytes or sixty-four bytes and schedules re-tries only for the transfers which have exhibited uncorrectable errors. If the overflow bit (222, 226) is set, the whole variable data payload will be retried. If the variable data payload continues to be uncorrectable after the retry it will be flagged as an uncorrectable error.

[0051] As a result, the retransmitting engine (214-8) retransmits, based on the overflow bit (222, 226), at least one of the cache lines associated with the error. Further, each of the cache lines is retransmitted when the overflow bit (222, 226) is set.

[0052] An overall example of the system (200) will now be described. The receiving engine (214-1) receives data payload command A (216-1) to transfer a variable data payload A (212-1) from the memory controller (202) to memory device A (206-1). The size determining engine (214-2) determines the size of variable data payload A (212-1) as one-hundred twenty-eight bytes. The cache line determining engine (214-3) determines that variable data payload A (212-1) is to be represented as two cache lines. The assigning engine (214-4) assigns variable data payload A (212-1) to two cache lines. The CRC engine (214-5) determines eight bit CRC characters for each of the two cache lines for variable data payload A (212-1). The transferring engine (214-6) transfers, the two cache lines with the CRC characters from the memory controller (202) to memory device A (206-1) as cache line A (218-1) to cache line B (218-2). The validating engine (214-7) further validates, based on the CRC character, each of the cache lines of the transfer as described above. In this example, cache line A (218-1) includes an error because the CRC characters do not match. As a result, the address of cache line A (218-1) is stored in error register A (220-1). Further, cache line B (218-2) does not include an error because the CRC characters match. Since the overflow bit (222) was not set, cache line A (218-1) is retransmitted by the retransmitting engine (214-8).

[0053] Fig. 3 is a flowchart of a method for correcting errors of a variable data payload, according to one example of principles described herein. The method (300) may be executed by the system (100) of Fig. 1. The method

(300) may be executed by other systems such as system 200, system 500, or system 600. In this example, the method (300) includes receiving (301) a data payload command to transfer a variable data payload from a memory controller to a memory device, determining (302), based on the size of the variable data payload, a number of cache lines for the variable data payload, assigning (303) the variable data payload to the number of cache lines, transferring (304) each of the cache lines with a CRC character from the memory controller to the memory device, and validating (305), based on the CRC character, each of the cache lines of the transfer.

[0054] As mentioned above, the method (300) includes receiving (301) a data payload command to transfer a variable data payload from a memory controller to a memory device. The data payload command may include a start address in the memory device, a length of the variable data payload, a termination character, or combinations thereof.

[0055] As mentioned above, the method (300) includes determining (302), based on the size of the variable data payload, a number of cache lines for the variable data payload. The method (300) determines, based on the size of the variable data payload, the number of cache lines for the variable data payload by determining a burst length, determining a bit width of the memory device, and determining a prefetch amount as described above.

[0056] As mentioned above, the method (300) includes assigning (303) the variable data payload to the number of cache lines. If the method (300) determines four cache lines are needed for the variable data payload, the variable data payload is divided into four equal sizes. Each of the four equal sizes of the variable data payload is assigned to a cache line.

[0057] As mentioned above, the method (300) includes transferring (304) each of the cache lines with a CRC character from the memory controller to the memory device. If the variable data payload is divided into four cache lines, each of the four cache lines is sent to the memory device with a CRC character.

[0058] As mentioned above, the method (300) includes validating (305), based on the CRC character, each of the cache lines of the transfer.

Validating, based on the CRC character, each of the cache lines of the transfer includes obtaining the CRC character associated with each of the cache lines before the transfer and obtaining the CRC character associated with each of the cache lines after the transfer.

[0059] Further, validating (305), based on the CRC character, each of the cache lines of the transfer includes comparing the CRC characters for each of the cache lines to determine if an error has occurred while transferring the cache lines from the memory controller to the memory device and storing, based on a flow bit of error registers, an address of the cache line associated with the error in at least one of the error registers of the memory device.

[0060] Further, the validating (305) may be performed in parallel with the transferring (304). This may reduce the time the method (300) takes to validate each of the cache lines. While this method (300) has been described with reference to transferring cache lines, the method (300) may transfer the entire variable data payload.

[0061] Fig. 4 is a flowchart of a method for correcting errors of a variable data payload, according to one example of principles described herein. The method (400) may be executed by the system (100) of Fig. 1. The method (400) may be executed by other systems such as system 200, system 500, or system 600. In this example, the method (400) includes receiving (401) a data payload command to transfer a variable data payload from a memory controller to a memory device, determining (402) the size of the variable data payload, determining (403), based on a size of the variable data payload, a number of cache lines for the variable data payload, assigning (404) the variable data payload to the number of cache lines, determining (405) a CRC character for each of the cache lines, transferring (406) each of the cache lines with the CRC character from the memory controller to the memory device to create a transfer, validating (407), based on the CRC character, each of the cache lines of the transfer, and retransmitting (408) at least one of the cache lines to correct errors associated with the transfer.

[0062] As mentioned above, the method (400) includes determining (402) a size of the variable data payload. The size of the variable data payload

may be sent on predefined address lanes of the memory controller and encoded in hex. For example, a hex value of 0x000F indicates that the variable data payload is one-kilobyte.

[0063] As mentioned above, the method (400) includes determining (405) the CRC character for each of the cache lines. Depending on the size, complexity, and operation of the memory controller and the memory devices, the CRC character may vary in length. For a simple memory controller and the memory devices, the CRC character may be three bits. For a complex memory controller and the memory devices, the CRC character may be eight bits. The CRC character for each of the cache lines may be distinct from one another. The CRC character for each of the cache lines may be the same value.

[0064] As mentioned above, the method (400) includes retransmitting (408) at least one of the cache lines to correct errors associated with the transfer. Retransmitting the at least one of the cache lines to correct the errors associated with the transfer includes determining if an overflow bit of error registers is set. If the overflow bit is not set, cache lines with errors are retransmitted by the method (400). If the overflow bit is set, all of the cache lines associated with the variable data payload is retransmitted by the method (400).

[0065] To avoid retransmitting large amounts of data and to improve the performance of the memory controller, the memory controller may keep track of the addresses of each of the cache lines during the validating (407). This may be accomplished via a counter. The counter increments the physical address of the cache line received after each validating (407) check occurs. Further, since the physical addresses of the cache lines that have exhibited errors have been stored and if the retransmitting (408) is successful, the method (400) repopulates the corrected cache line inside the variable data payload.

[0066] While this method (400) has been described with reference to using CRC characters for validation, other techniques may be used. Other techniques may include utilizing error correcting code.

[0067] Fig. 5 is a diagram of a correcting system, according to one example of principles described herein. The correcting system (500) includes a

receiving engine (514-1), a size determining engine (514-2), a cache line determining engine (514-3), an assigning engine (514-4), a CRC engine (514-5), a transferring engine (514-6), validating engine (514-7), and a retransmitting engine (514-8). The engines (514) refer to a combination of hardware and program instructions to perform a designated function. Each of the engines (514) may include a processor and memory. The program instructions are stored in the memory and cause the processor to execute the designated function of the engine.

[0068] The receiving engine (514-1) receives a data payload command to transfer a variable data payload from a memory controller to a memory device. The receiving engine (514-1) may receive one data payload command to transfer a variable data payload from a memory controller to a memory device. The receiving engine (514-1) may receive several data payload commands to transfer several variable data payload from a memory controller to a memory device.

[0069] The size determining engine (514-2) determines a size of the variable data payload. The size determining engine (514-2) may determine the size of the variable data payload in bits, bytes, kilobytes, gigabytes, terabytes, other sizes, or combinations thereof.

[0070] The cache line determining engine (514-3) determines, based on the size of the variable data payload, a number of cache lines for the variable data payload. The cache line determining engine (514-3) determines the number of cache lines based on the size of the cache lines. If a variable data payload is one kilobyte and a cache lines is sixty four bytes, the cache line determining engine (514-3) determines sixteen cache lines are needed for the variable data payload.

[0071] The assigning engine (514-4) assigns the variable data payload to the number of cache lines. If sixteen cache lines are needed for the variable data payload, the assigning engine (514-4) assigns the variable data payload to sixteen cache lines.

[0072] The CRC engine (514-5) determines the CRC character for each of the cache lines. The CRC character may be two bits, four bits, eight bits, or other lengths of bits depending on the size of the variable data payload.

[0073] The transferring engine (514-6) transfers each of the cache lines with a CRC character from the memory controller to the memory device. The transferring engine (514-6) may transfer each of the cache lines with a CRC character from the memory controller to the memory device in a sequential order.

[0074] The validating engine (514-7) validates, based on the CRC character, each of the cache lines of the transfer. The validating engine (514-7) may randomly validate at least one cache line of the transfer.

[0075] The retransmitting engine (514-8) retransmits at least one of the cache lines to correct errors associated with the transfer. The retransmitting engine (514-8) retransmits at least one of the cache lines to correct errors associated with the transfer based on an overflow bit.

[0076] Fig. 6 is a diagram of a correcting system, according to one example of principles described herein. In this example, correcting system (600) includes processing resources (602) that are in communication with a machine-readable storage medium (604). Processing resources (602) include at least one processor and other resources used to process instructions. The machine-readable storage medium (604) represents generally any memory capable of storing data such as instructions or data structures used by the correcting system (600). The instructions shown stored in the machine-readable storage medium (604) includes receiving instructions (606), size determining instructions (608), cache line determining instructions (610), assigning instructions (612), CRC determining instructions (614), transferring instructions (616), validating instructions (618), retransmitting instructions (620).

[0077] The machine-readable storage medium (604) contains computer readable program code to cause tasks to be executed by the processing resources (602). The machine-readable storage medium may be tangible and/or physical storage medium. The machine-readable storage medium may be any appropriate storage medium that is not a transmission

storage medium. A non-exhaustive list of machine-readable storage medium types includes non-volatile memory, volatile memory, random access memory, write only memory, flash memory, electrically erasable program read only memory, or types of memory, or combinations thereof.

[0078] The receiving instructions (606) represents instructions that, when executed, cause the processing resources (602) to receive a data payload command to transfer a variable data payload from a memory controller to a memory device. The size determining instructions (608) represents instructions that, when executed, cause the processing resources (602) to determine a size of the variable data payload.

[0079] The cache line determining instructions (610) represents instructions that, when executed, cause the processing resources (602) to determine, based on the size of the variable data payload, a number of cache lines for the variable data payload. The assigning instructions (612) represents instructions that, when executed, cause the processing resources (602) to assign the variable data payload to the number of cache lines.

[0080] The CRC determining instructions (614) represents instructions that, when executed, cause the processing resources (602) to determine the CRC character for each of the cache lines. The transferring instructions (616) represents instructions that, when executed, cause the processing resources (602) to transfer each of the cache lines with a CRC character from the memory controller to the memory device.

[0081] The validating instructions (618) represents instructions that, when executed, cause the processing resources (602) to validate, based on the CRC character, each of the cache lines of the transfer. The retransmitting instructions (620) represents instructions that, when executed, cause the processing resources (602) to retransmit at least one of the cache lines to correct errors associated with the transfer.

[0082] Further, the machine-readable storage medium (604) may be part of an installation package. In response to installing the installation package, the instructions of the machine-readable storage medium (604) may be downloaded from the installation package's source, such as a portable

medium, a server, a remote network location, another location, or combinations thereof. Portable memory media that are compatible with the principles described herein include DVDs, CDs, flash memory, portable disks, magnetic disks, optical disks, other forms of portable memory, or combinations thereof. In other examples, the program instructions are already installed. Here, the memory resources can include integrated memory such as a hard drive, a solid state hard drive, or the like.

[0083] In some examples, the processing resources (602) and the memory resources (602) are located within the same physical component, such as a server, or a network component. The machine-readable storage medium (604) may be part of the physical component's main memory, caches, registers, non-volatile memory, or elsewhere in the physical component's memory hierarchy. Alternatively, the machine-readable storage medium (604) may be in communication with the processing resources (602) over a network. Further, the data structures, such as the libraries, may be accessed from a remote location over a network connection while the programmed instructions are located locally. Thus, the correcting system (600) may be implemented on a user device, on a server, on a collection of servers, or combinations thereof.

[0084] The correcting system (600) of Fig. 6 may be part of a general purpose computer. However, in alternative examples, the correcting system (600) is part of an application specific integrated circuit.

[0085] The preceding description has been presented to illustrate and describe examples of the principles described. This description is not intended to be exhaustive or to limit these principles to any precise form disclosed. Many modifications and variations are possible in light of the above teaching.

CLAIMS

WHAT IS CLAIMED IS:

1. A method for correcting errors of a variable data payload, the method comprising:

receiving a data payload command to transfer a variable data payload from a memory controller to a memory device;

determining, based on a size of the variable data payload, a number of cache lines for the variable data payload;

assigning the variable data payload to the number of cache lines;

transferring each of the cache lines with a cyclic redundancy check (CRC) character from the memory controller to the memory device to create a transfer; and

validating, based on the CRC character, each of the cache lines of the transfer.

2. The method of claim 1, in which validating, based on the CRC character, each of the cache lines of the transfer comprises:

obtaining the CRC character associated with each of the cache lines before the transfer;

obtaining the CRC character associated with each of the cache lines after the transfer;

comparing the CRC characters for each of the cache lines to determine if an error has occurred while transferring the cache lines from the memory controller to the memory device; and

storing, based on an overflow bit, addresses of the cache lines associated with the error in at least one error register of the memory device.

3. The method of claim 1, further comprising retransmitting at least one of the cache lines to correct errors associated with the transfer.

4. The method of claim 3, in which retransmitting the at least one of the cache lines to correct the errors associated with the transfer comprises:

determining if an overflow bit is set; and

retransmitting, based on the overflow bit, at least one of the cache lines associated with the error.

5. The method of claim 4, in which the overflow bit is set when an error threshold is exceeded, the error threshold corresponding to a number of error registers of the memory device.

6. The method of claim 4, in which each of cache lines is retransmitted when the overflow bit is set.

7. The method of claim 1, further comprising:

determining the size of the variable data payload; and

determining the CRC character for each of the cache lines.

8. A system for correcting errors of a variable data payload, the system comprising:

an assigning engine to assign a variable data payload to a number of cache lines;

a CRC determining engine to determine a cyclic redundancy check (CRC) character for each of the cache lines:

a transferring engine to transfer each of the cache lines with the CRC character from a memory controller to a memory device to create a transfer;

a validating engine to validate, based on the CRC character, each of the cache lines of the transfer; and

a retransmitting engine to retransmit at least one of the cache lines to correct errors associated with the transfer.

9. The system of claim 8, in which validating engine validates, based on the CRC character, each of the cache lines of the transfer by:

obtaining the CRC character associated with each of the cache lines before the transfer;

obtaining the CRC character associated with each of the cache lines after the transfer;

comparing the CRC characters for each of the cache lines to determine if an error has occurred while transferring the cache lines from the memory controller to the memory device; and

storing, based on an overflow bit, addresses of the cache lines associated with the error in at least one error register of the memory device.

10. The system of claim 8, in which the retransmitting engine retransmits the at least one of the cache lines to correct the errors associated with the transfer by:

determining if an overflow bit is set; and

retransmitting, based on the overflow bit, at least one of the cache lines associated with the error;

in which the overflow bit is set when an error threshold is exceeded, the error threshold corresponding to a number of error registers of the memory device.

11. The system of claim 8, further comprising:

a receiving engine to receive a data payload command to transfer the variable data payload from the memory controller to the memory device;

a size determining engine to determine a size of the variable data payload; and

a cache line determining engine to determine, based on the size of the variable data payload, the number of cache lines for the variable data payload.

12. A machine-readable storage medium encoded with instructions for correcting errors of a variable data payload, the instructions executable by a processor of a system to cause the system to:

assign a variable data payload to a number of cache lines;

transfer each of the cache lines with a cyclic redundancy check (CRC) character from a memory controller to a memory device to create a transfer;

validate, based on the CRC character, each of the cache lines of the transfer; and

retransmit at least one of the cache lines to correct errors associated with the transfer.

13. The product of claim 12, further comprising instructions that, when executed, cause the processor to:

obtain the CRC character associated with each of the cache lines before the transfer;

obtain the CRC character associated with each of the cache lines after the transfer;

compare the CRC characters for each of the cache lines to determine if an error has occurred while transferring the cache lines from the memory controller to the memory device; and

store, based on an overflow bit, addresses of the cache lines associated with the error in at least one error register of the memory device.

14. The product of claim 12, further comprising instructions that, when executed, cause the processor to

determine if an overflow bit is set; and

retransmit, based on the overflow bit, at least one of the cache lines associated with the error.

15. The product of claim 12, further comprising instructions that, when executed, cause the processor to:

receive a data payload command to transfer the variable data payload from the memory controller to the memory device;

determine, based on a size of the variable data payload, the number of cache lines for the variable data payload;

determine the size of the variable data payload; and

determine the CRC character for each of the cache lines.

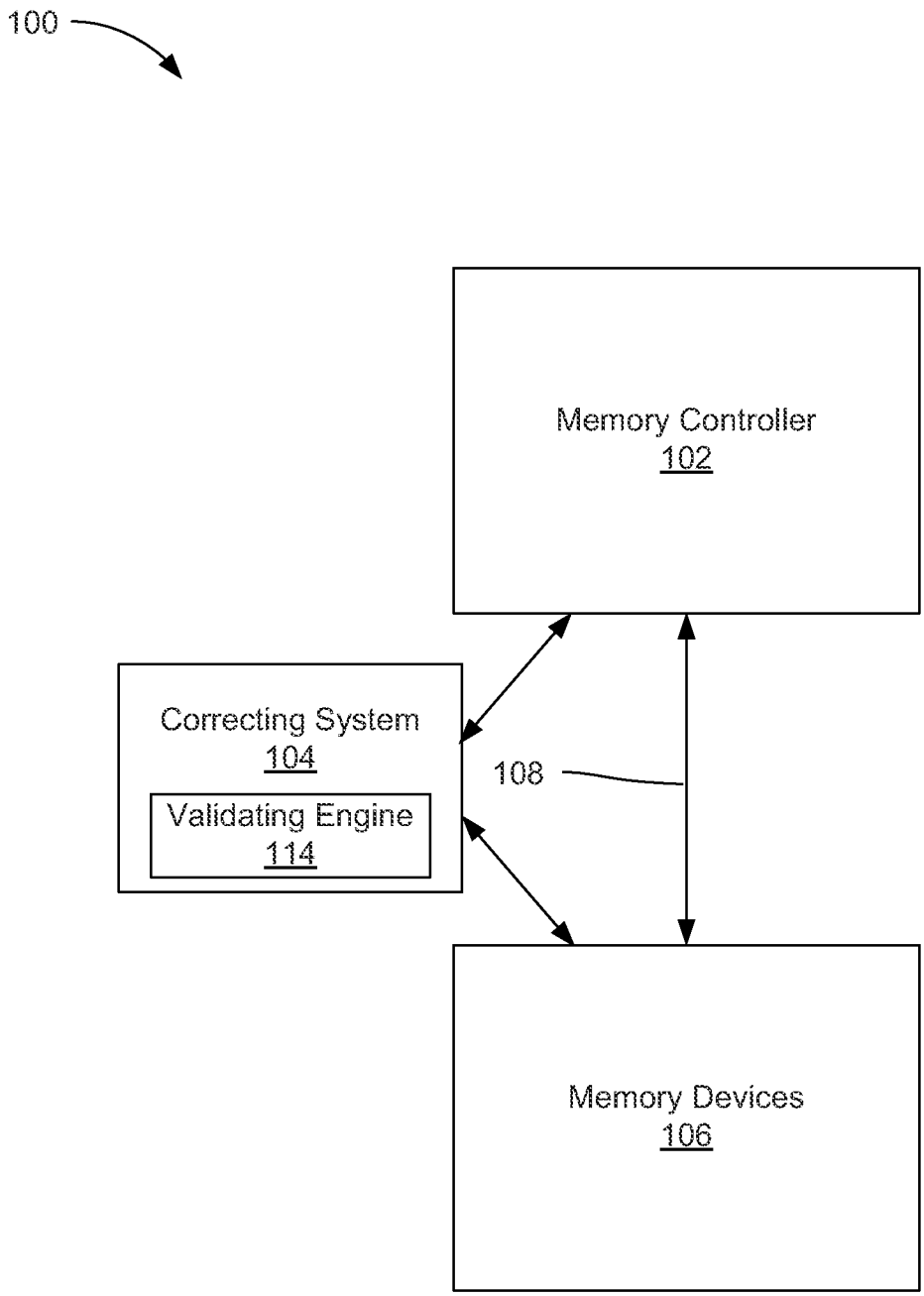
100
Memory Controller
102
Correcting System
104
Validating Engine
114
108
Memory Devices
106

Fig. 1

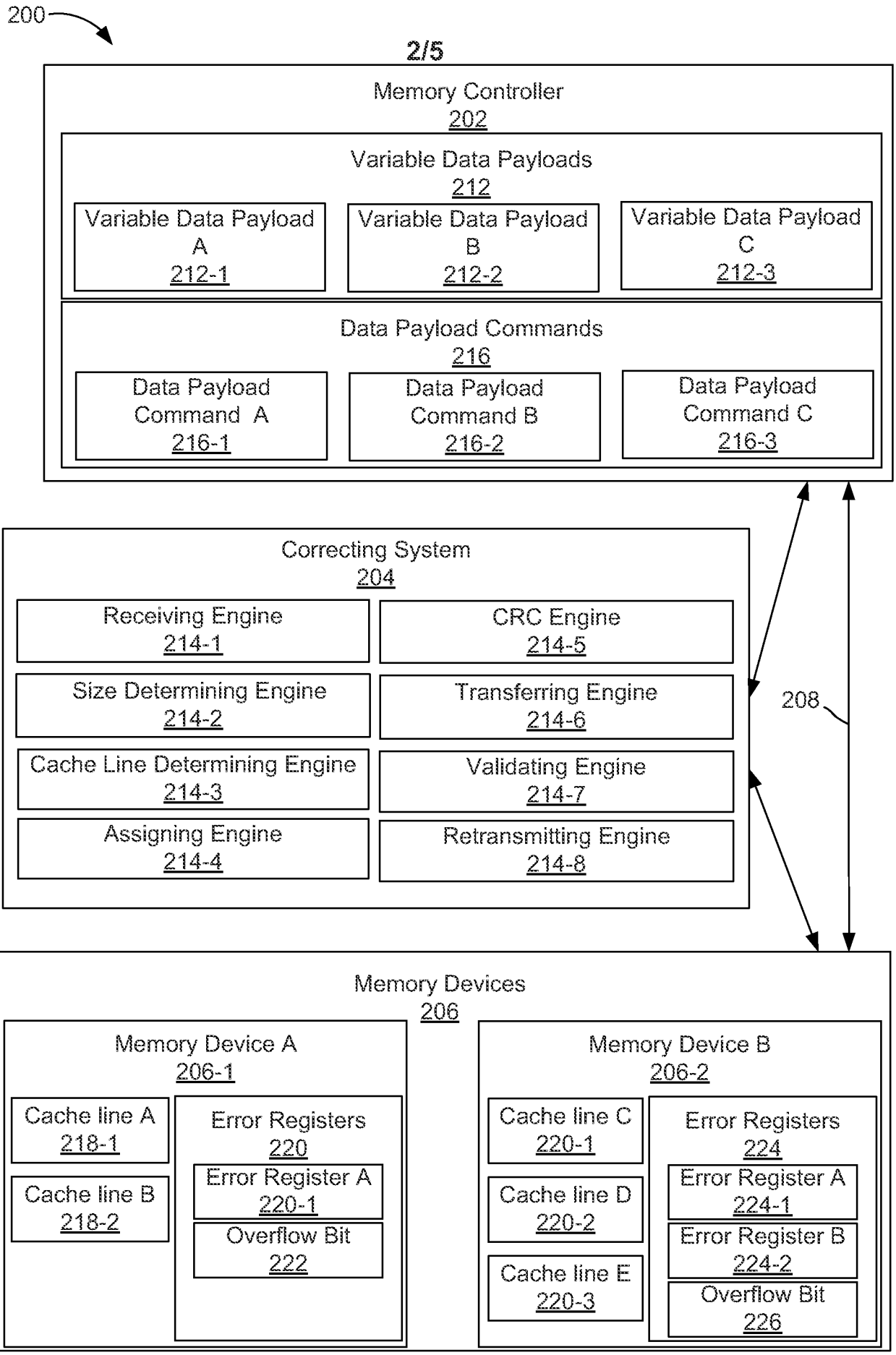
200
Memory Controller
202
Variable Data Payloads
212
Variable Data Payload A
212-1
Variable Data Payload B
212-2
Variable Data Payload C
212-3
Data Payload Commands
216
Data Payload Command A
216-1
Data Payload Command B
216-2
Data Payload Command C
216-3
Correcting System
204
Receiving Engine
214-1
CRC Engine
214-5
Size Determining Engine
214-2
Transferring Engine
214-6
Cache Line Determining Engine
214-3
Validating Engine
214-7
Assigning Engine
214-4
Retransmitting Engine
214-8
208
Memory Devices
206
Memory Device A
206-1
Cache line A
218-1
Cache line B
218-2
Error Registers
220
Error Register A
220-1
Overflow Bit
222
Memory Device B
206-2
Cache line C
220-1
Cache line D
220-2
Cache line E
220-3
Error Registers
224
Error Register A
224-1
Error Register B
224-2
Overflow Bit
226

Fig. 2

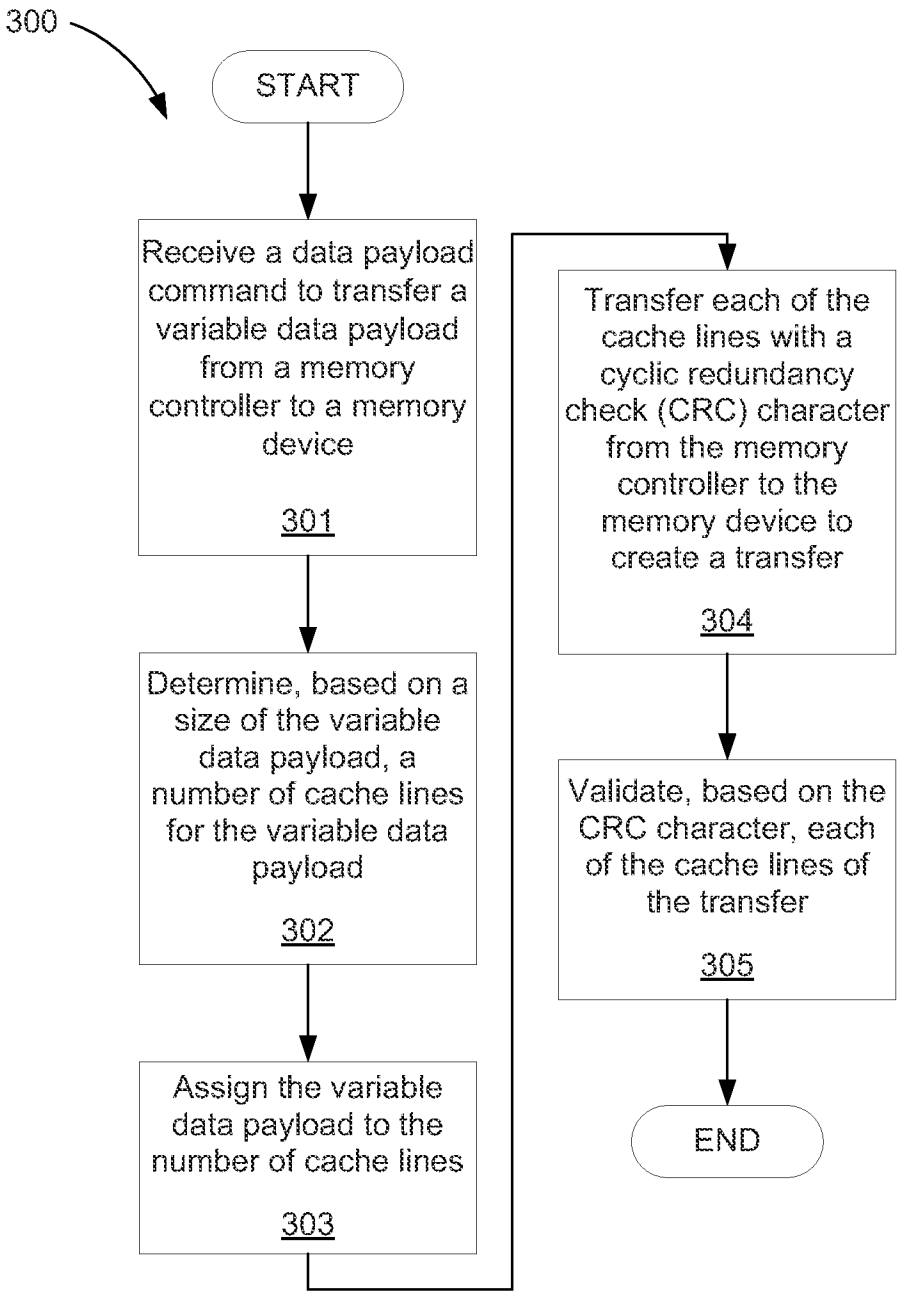
300
START
Receive a data payload command to transfer a variable data payload from a memory controller to a memory device
301
Determine, based on a size of the variable data payload, a number of cache lines for the variable data payload
302
Assign the variable data payload to the number of cache lines
303
Transfer each of the cache lines with a cyclic redundancy check (CRC) character from the memory controller to the memory device to create a transfer
304
Validate, based on the CRC character, each of the cache lines of the transfer
305
END

Fig. 3

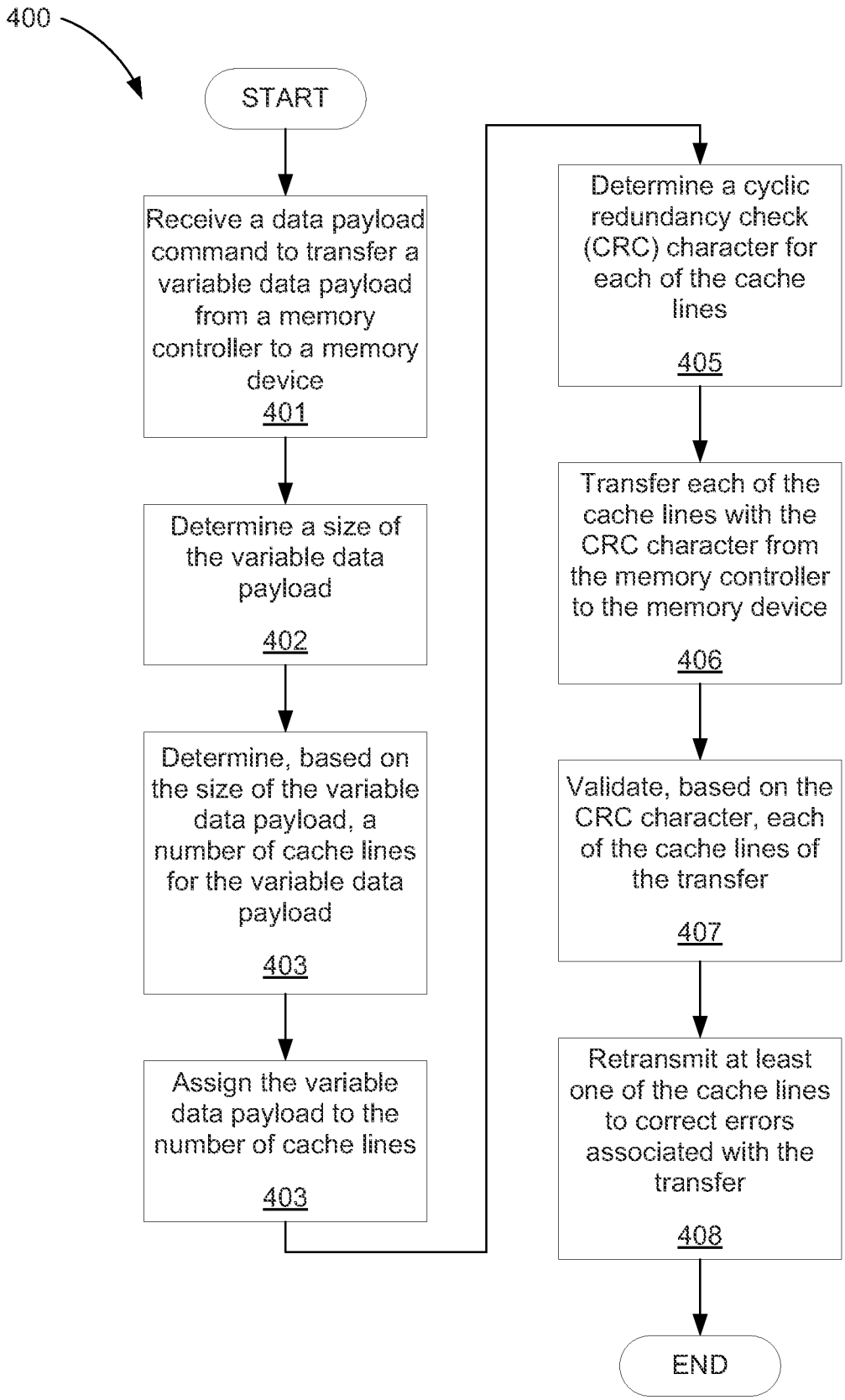
400
START
Receive a data payload command to transfer a variable data payload from a memory controller to a memory device
401
Determine a size of the variable data payload
402
Determine, based on the size of the variable data payload, a number of cache lines for the variable data payload
403
Assign the variable data payload to the number of cache lines
403
Determine a cyclic redundancy check (CRC) character for each of the cache lines
405
Transfer each of the cache lines with the CRC character from the memory controller to the memory device
406
Validate, based on the CRC character, each of the cache lines of the transfer
407
Retransmit at least one of the cache lines to correct errors associated with the transfer
408
END

Fig. 4

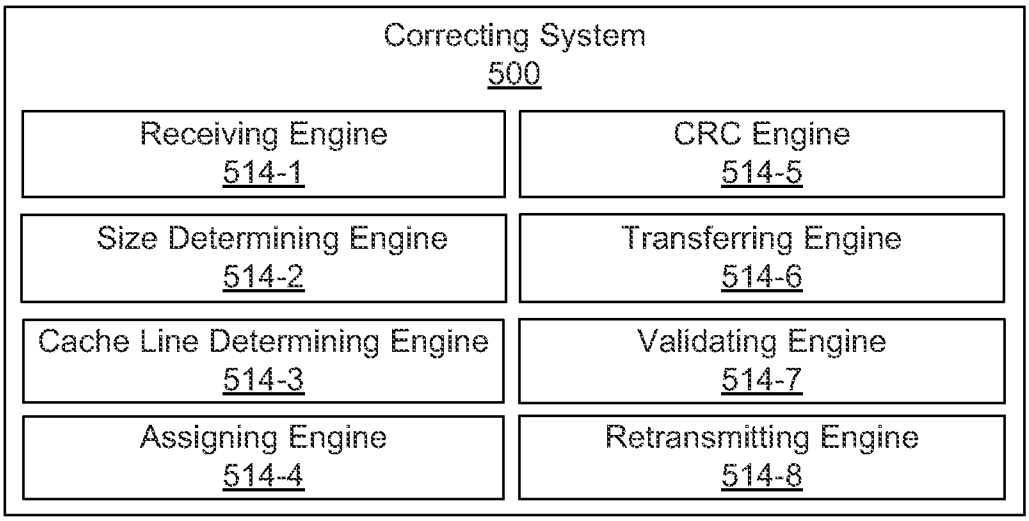
Correcting System
500
Receiving Engine
514-1
CRC Engine
514-5
Size Determining Engine
514-2
Transferring Engine
514-6
Cache Line Determining Engine
514-3
Validating Engine
514-7
Assigning Engine
514-4
Retransmitting Engine
514-8

Fig. 5

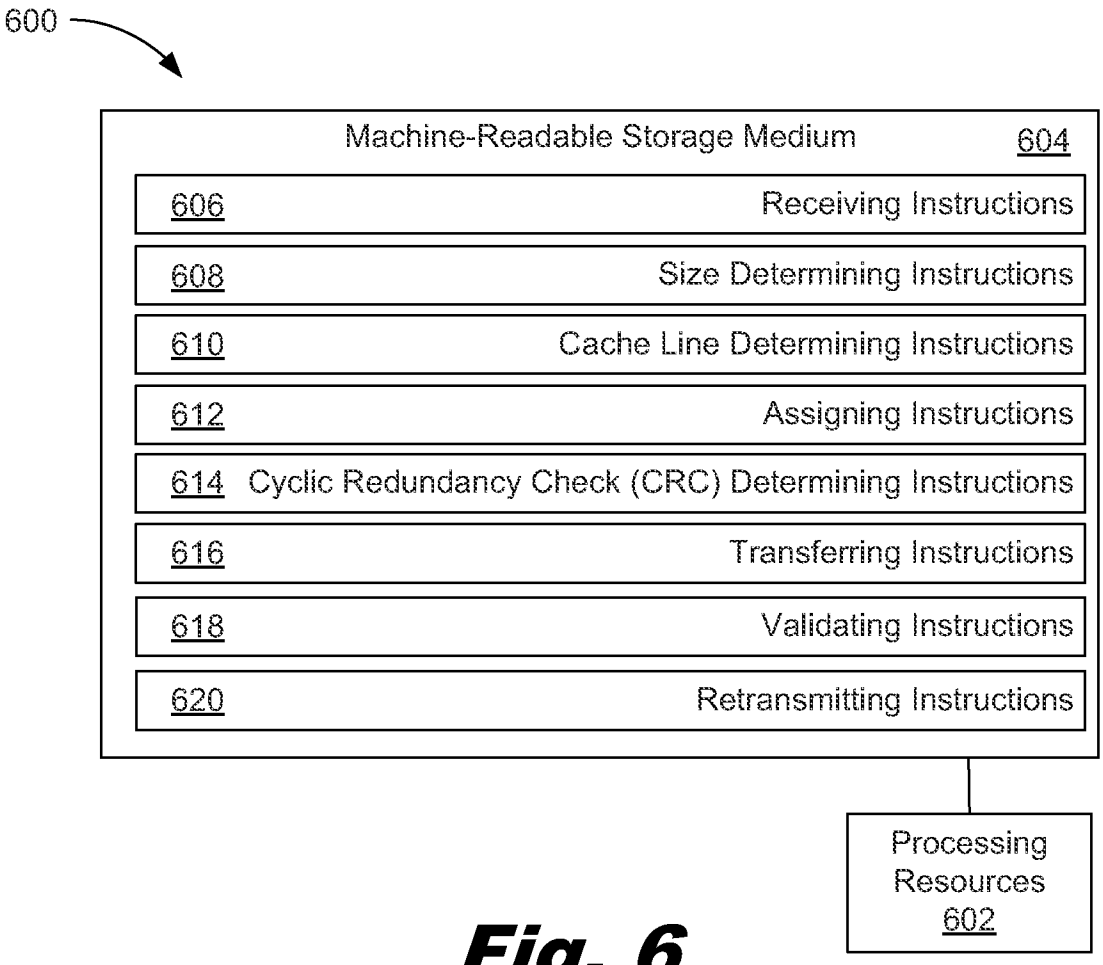
600
Machine-Readable Storage Medium
604
606
Receiving Instructions
608
Size Determining Instructions
610
Cache Line Determining Instructions
612
Assigning Instructions
614
Cyclic Redundancy Check (CRC) Determining Instructions
616
Transferring Instructions
618
Validating Instructions
620
Retransmitting Instructions
Processing Resources
602

Fig. 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2015/013124

A. CLASSIFICATION OF SUBJECT MATTER

G06F 11/10(2006.01)i, G06F 12/00(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 11/10; H03M 13/09; G06F 12/00; H04L 1/08; G06F 11/00

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: memory, CRC, cache line, error detection, variable data size, and similar terms.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2009-0187806 A1 (JOHN C. PESCATORE) 23 July 2009 See paragraphs [0002], [0005]-[0007], [0017], and [0022]-[0024]; and figures 1-6.	1,3,7-8,11-12,15
A		2,4-6,9-10,13-14
Y	US 2012-0254562 A1 (MICHAEL J. MORRISON et al.) 04 October 2012 See paragraphs [0035] and [0049]; and figures 1, 4B, and 6.	1,3,7-8,11-12,15
A	US 2014-0089755 A1 (SHVETA KANTAMSETTI et al.) 27 March 2014 See paragraphs [0015]-[0017] and [0036]-[0040]; and figures 3 and 5.	1-15
A	WO 2008-073494 A1 (ADVANCED MICRO DEVICES, INC.) 19 June 2008 See paragraphs [0023]-[0024], [0032]-[0037], and [0045]; and figures 1-2.	1-15
A	US 2014-0149833 A1 (DELL PRODUCTS L.P.) 29 May 2014 See paragraph [0045] and figure 9.	1-15

☐ 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
"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
22 September 2015 (22.09.2015)	**22 September 2015 (22.09.2015)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 35208, Republic of Korea
Facsimile No. +82-42-472-7140

Authorized officer
NHO, Ji Myong
Telephone No. +82-42-481-8528

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2015/013124

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2009-0187806 A1	23/07/2009	US 2006-0077750 A1	13/04/2006
		US 8341499 B2	25/12/2012
US 2012-0254562 A1	04/10/2012	CN 102968291 A	13/03/2013
		EP 2506149 A1	03/10/2012
		JP 2012-216210 A	08/11/2012
		KR 10-2012-0112265 A	11/10/2012
		US 2012-0254558 A1	04/10/2012
		US 8473695 B2	25/06/2013
		US 8635417 B2	21/01/2014
US 2014-0089755 A1	27/03/2014	None	
WO 2008-073494 A1	19/06/2008	CN 101681277 A	24/03/2010
		CN 101681277 B	20/02/2013
		DE 112007003086 T5	08/10/2009
		GB 2457618 A	26/08/2009
		JP 2010-514264 A	30/04/2010
		KR 10-2009-0099558 A	22/09/2009
		TW 200841168 A	16/10/2008
		TW I461899 B	21/11/2014
		US 2008-0148131 A1	19/06/2008
		US 7881303 B2	01/02/2011
		US RE44487 E1	10/09/2013
US 2014-0149833 A1	29/05/2014	US 2013-0111308 A1	02/05/2013
		US 8645811 B2	04/02/2014
		US 9009580 B2	14/04/2015