In one example in accordance with the present disclosure, a method includes mapping, using post-package repair, an address associated with a first memory row of a computing device to a spare memory row of the computing device, wherein the spare memory row has a memory failure pattern, and reading data from the spare memory row.

**FIG. 1**

**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))

**Published:**

— with international search report (Art. 21(3))
POST PACKAGE REPAIR FOR MAPPING TO A MEMORY FAILURE PATTERN

BACKGROUND
[0001] Newer memories, such as double data ram version 4 (DDR4) include so-called post-package repair (PPR) capabilities. PPR capabilities enable a compatible memory controller to remap accesses from a faulty row of a memory module to a spare row of the memory module that is not faulty.

BRIEF DESCRIPTION OF THE DRAWINGS
[0002] The following detailed description references the drawings, wherein:
[0003] FIG. 1 is a block diagram of an example computing system that includes post-package repair technology.
[0004] FIG. 2 is a block diagram of an example system memory that includes fault injection capability using post-package repair technology.
[0005] FIG. 3 is a block diagram of an example system memory that includes fault injection capability using post-package repair technology.
[0006] FIG. 4 is a block diagram of an example of changing a memory failure pattern in a spare row of system memory using post-package repair technology.
[0007] FIG. 5 is a flowchart of an example method for performing fault injection using post-package repair technology.
[0008] FIG. 6 is a flowchart of an example method for performing fault injection using post-package repair technology.
[0009] FIG. 7 is a block diagram of an example system for performing fault injection using post-package repair technology.
[0010] FIG. 8 is a block diagram of an example system for performing fault injection using post-package repair technology.
DETAILED DESCRIPTION

[0011] Computing devices include various types of synchronous dynamic random access memory (SDRAM), such as double data rate (DDR) SDRAM. As the capacity of DDR SDRAM included in computing devices has grown, so too has the failure rate of DDR SDRAM modules. DDR SDRAM modules are referred to as DIMMs (Dynamic Inline Memory Modules).

[0012] Each DIMM comprises at least one memory chip. Each memory chip comprises rows and columns of dynamic RAM cells. A memory controller, which is coupled to a processor, such as a CPU, and/or GPU, issues memory transfer requests to read or write data to or from a particular address of a DIMM. The memory controller may also issue other non-data transfer commands to a DIMM, such as PPR commands.

[0013] When processing a memory transfer request associated with a particular address, the memory controller determines a DIMM that is associated with the address. The Memory controller also determines as well as a column address, and a row address of the DIMM that are associated with the address. To read or write data at the designated row and column, the memory controller applies an address signal to indicate a designated row and column of the DIMM to access.

[0014] DIMMs may have faulty rows, columns, chips, banks, or even for entire DIMMs to be faulty. Some of the faults may be transient (temporary) while other faults may be permanent. To address transient and non-transient memory failures, DDR4 DIMMs include so-called "post-package repair" (PPR) capability. Based on ECC data within a word, a memory controller may detect permanent or transient errors. PPR allows a compatible memory controller to remap the faulty row to a spare row.
[001 5] The DDR4 standard defines two types of PPR remapping capabilities, referred to as "hard-PPR" and "soft-PPR." Hard-PPR permanently remaps accesses from a designated faulty row to a designated spare row. A Hard-PPR row remapping survives power cycles. Soft-PPR remapping temporarily maps accesses from a faulty row to a designated spare row. A Soft-PPR row remapping will survive a "warm" reboot, but does not survive a power cycle.

[001 6] Some techniques of this disclosure enable a memory controller to perform fault injection and testing of memory modules using hard- and/or soft-PPR capabilities of DRAM modules. A memory controller configured according to this disclosure may use PPR to remap memory transfer requests from a first row of a memory module to a designated spare row that is configured to have a particular memory failure pattern.

[001 7] To ensure that the memory controller appropriately detects, and responds to failures, the memory controller may read from the spare row having the failure pattern. Based on how the memory controller reacts to the failure pattern, a processor or other testing hardware may validate that memory controller's error correction, and PPR capabilities work as intended. For example, a test system that includes a processor may compare a memory controller's execution in response to at least one failure pattern to previously-generated model test execution patterns for the same memory failure patterns. The test system may validate the memory controller's fault handling capability based on whether or not the memory controller's execution matches the previously-determined execution for the at least one failure pattern.

[001 8] By using PPR-based fault injection as described herein, the techniques of this disclosure achieve several improvements. The techniques of this disclosure use existing PPR technology, which is already built-in to some DRAM modules. The techniques of this disclosure thus requiring minimal modification of memory modules. Some previous techniques for detecting memory errors modified the memory controller specifically for performing testing. Such modifications introduced greater potential for memory controller malfunction. Additionally, such modifications had to be designed specifically for each hardware vendor's chipset. The techniques of this
disclosure work with any PPR-compatible memory controller regardless of the underlying memory controller architecture, thereby reducing complexity.

[0019] Some techniques for performing fault injection and memory controller validation used known-failing memory modules. Such failing memory modules had a single failed address, so a test system could not validate a memory controller's response to failures across a large address range. Additionally, such failing memory modules could act non-deterministically and unreliably. The techniques of this disclosure enable a test system to validate a memory controller's response to memory failures over a broad (or even the entire) memory address range.

[0020] Additionally, because techniques did not allow a memory controller to inject a particular failure pattern into a row, these techniques could not provide that errors will be correctable or uncorrectable. The techniques of this disclosure also enable a computing device to inject a particular failure pattern. By enabling a memory controller to inject a particular failure pattern into a spare row, a computing system may test a memory controller's response to both correctable and uncorrectable errors while also maintaining deterministic memory module behavior.

[0021] FIG. 1 is a block diagram of an example computing system 100. Computing system 100 comprise a system for validating memory controller 104, and may include a central processing unit (CPU) 102. CPU 102 is illustrated as a single CPU for the purpose of example. In various examples, CPU 102 may comprise a plurality of CPUs. CPU 102 may further include memory controller 104. Memory controller 104 is illustrated as being integrated in CPU 102 in FIG. 1. However, memory controller 104 may be discrete from CPU 102 in various examples.

[0022] Computing system 100 also includes system memory 106. Memory controller 104 and system memory 106 may be coupled with a memory bus 108. Memory bus 108 may comprise a data and/or memory bus. System memory 106 may comprise one or more memory modules, such as memory module 112A-1 12N (“memory modules 112”). Memory modules 112 may comprise DDR4 modules or any other type of memory module that includes PPR capability.
Each of memory modules 112 includes a plurality of rows. For example, memory module 112A includes rows 114A-1 14N ("rows 114"). In addition to rows 114, memory module 112A includes a spare row 116. Spare row 116 is illustrated as a single row for the purpose of example, and may comprise a plurality of spare rows in various examples.

Spare row 116 may include a memory failure pattern 118 in various examples. A memory failure pattern may comprise one or more "stuck-at faults." Stuck-at faults as described herein are bits that have a fixed value, and do not change if written-to. For example, memory failure pattern may comprise a data word that includes one or more stuck-at 1's or stuck-at 0's. A stuck-at 1 value does not change from a one to a zero even if a zero is written to that bit. A stuck-at 0 value does not change from a zero to a one even if a one is written to that bit.

According to the techniques of this disclosure, memory controller 104 may be configured to use either soft- or hard-post PPR to remap a first row of one of rows 114 to spare row 116, having a particular memory failure pattern 118. By remapping one of rows 114 to row 116 having memory failure pattern 118, computing system 100 may test and/or validate the behavior of memory controller 104 in response to memory failure pattern 118.

FIG. 2 is a block diagram of an example system memory that includes fault injection capability using post-package repair technology. FIG. 2 includes memory controller 104, and system memory 106. Memory controller 104 and system memory 106 are coupled via memory bus 108. System memory 106 comprises one or more memory modules 112 that include hard- and/or soft-PPR capability. Each of memory modules 112 includes a plurality of rows. For example, memory module 112A includes rows 114A-1 14N ("rows 114"). In addition to rows 114, memory module 112A includes a spare row 116.

In the example of FIG. 2, spare row 116 includes a failure pattern 202. Failure pattern 202 comprises an 8-bit failure pattern comprising two stuck-at one bits, and one stuck-at zero bit. The "x's" within failure pattern 202 indicate bits that
are not stuck at a particular value. Failure pattern 202 may comprise any number of stuck bits and/or bits fixed to a particular value within spare row 116.

[0028] In the example of FIG. 2, memory controller 104 receives a transfer request for a particular memory address x, and decodes the request. As part of decoding the request for address x, memory controller 104 determines that the memory address is associated with a row and a column of memory module 112A.

[0029] In the example of FIG. 2, memory controller 104 determines the request for address x maps to row 114A of memory module 112A. In this example, memory control module 112A holds spare row 116 in reserve, and does not use spare row 116. To perform the transfer request, memory controller 104 sends a row address, opens a page with a given row address, and accesses a column within that row of memory module 112A. Memory controller 104 then reads and/or writes any data to the data word located indicated by the row 114A and the designated column of memory module 112A.

[0030] FIG. 3 is a block diagram of an example system memory that includes fault injection capability using post-package repair technology. In FIG. 3, memory controller 104 receives a transfer request for the same memory address x as described above with respect to FIG. 2. However, in FIG. 3, memory controller 104 has remapped the row associated with address x from row 114A, to to spare row 116, which includes failure pattern 202.

[0031] To remap row 114A to spare row 116, memory controller 104 issues at least one command to PPR control register 204. PPR control register 204 is a register within memory module 112A. PPR control register 204 controls the PPR remapping functionality of memory module 112A. PPR module 204 receives a command indicating a row that is to be remapped (e.g., row 114A), and/or data indicating the spare replacement row (spare row 116 in this example) that replaces the row to be remapped. Responsive to PPR control register 204 receiving a PPR remapping command, memory module 112A remaps accesses to row 114A to spare row 116.
Responsive to remapping row 114A to spare row 116, a processor, such as CPU 102, issues transfer requests to memory controller 104 to determine how memory controller 104 responds to various failure patterns. As an example, CPU 102 may issue transfer requests for memory address x, which has been remapped to be associated with spare row 116. The transfer request may cause memory controller 104 to write a particular value to spare row 116. However, as indicated above, spare row 116 includes failure pattern 202, which includes a number of stuck-at bits. Due to the stuck-at bits of failure pattern 202, during a write request, memory controller 104 is unable to change the values of the stuck-at bits of failure pattern 202.

Memory controller 104 and memory module 112A may include error correcting code (ECC) capability in some examples. For each word of data stored in memory module 112A, memory module 112A also includes additional bits used to store ECC data for that word. Memory controller 104 calculates the values of the ECC bits for each word during a write operation. Memory controller 104 uses the ECC bits to verify correctness of the data word during a read operation.

To validate memory controller 104, CPU 102 issues a memory transfer request at address x, which causes memory controller 104 to read from spare row 116. During a read, memory controller 104 may determine whether a correctable or an uncorrectable error is present based on the data read from spare row 116, as well as the additional ECC bits. Memory controller 104 records the occurrence of single-bit (correctable) or multiple-bit (uncorrectable) errors. If memory controller 104 detects a correctable error, memory controller 104 corrects the error.

CPU 102 may generate memory reads and writes patterns that are known to differ relative to memory failure pattern 118. As an example, CPU 102 may generate memory reads and writes that differ from memory failure pattern 118 by a single bit to test whether the ECC functionality of memory controller 104 properly corrects these one-bit errors. Similarly, memory controller 104 may generate errors that are known not to be correctable to ensure that memory
controller 104 does not attempt to correct such errors, and to test whether memory controller 104 detects such errors as being uncorrectable.

[0036] CPU 102 or another processor may also issue transfer requests to spare row 116 to test the PPR functionality of memory controller 104. As an example, CPU 102 or another processor may issue a transfer request at address x corresponding to failure row 116. CPU 102 may issue the transfer requests in such a way so as to cause memory controller 104 to determine that spare row 116 has failed.

[0037] For example, CPU 102 may configure failure pattern 202 to be uncorrectable by setting a failure pattern and issuing a write request that varies by more than one bit relative to the failure pattern. CPU 102 may then issue multiple read requests to spare row 116 so as to cause memory controller 104 to identify repeated uncorrectable read failures. Memory controller 104 may determine that spare row 116 has failed once a threshold number of uncorrectable transfer operations have occurred in some examples.

[0038] Based on the determination that spare row 116 has failed, memory controller 104 may issue at least one command to PPR control register 204 to remap spare row 116, e.g. to another spare row using either hard- or soft-PPR functionality. In some examples, memory controller 104 may decide whether to use hard-PPR or soft-PPR row remapping to correct memory failure patterns based on the number of failed memory transfers, and/or the type (e.g., read or write) transfers, and whether the memory transfer errors were correctable.

[0039] FIG. 4 is a block diagram of an example of changing a memory failure pattern in a spare row of system memory using post-package repair technology. In the example of FIG. 4, memory controller 104 transmits data to PPR control register 204. Responsive to receiving the data, PPR control register 204 may change failure pattern 202 of spare row 116.
To change the failure pattern, memory controller 104 may transmit a command to PPR control register 204. The command may indicate that memory controller 204 is supplying a new fault pattern to PPR control register 204. In the example of FIG. 4, the new failure pattern is "10xx1011." Responsive to receiving the failure pattern, PPR control register 204 updates failure pattern 402 with the value "10xx1011," which includes three stuck-at one bits, and two stuck-at zero bits. Although failure pattern 402 includes 8 bits, it should be understood that failure pattern 402 may comprise any number of bits (e.g., 32 or 64 bits). In some examples, responsive to transmitting a command to PPR control register 204, memory controller 104 may transmit new failure 402 pattern to spare row 116 via memory bus 108.

By signaling PPR control register 204, the techniques of this disclosure enable a processor to test and validate the response of memory controller 104 to a particular failure pattern against a predetermined response. For example, system 100 may set failure pattern 118 to a particular failure pattern for which system 100 has previously determined response data for that failure pattern. CPU 102 may compare the response of memory controller 104 to the failure pattern with the predetermined response data to determine whether or not memory controller 104 is functioning properly.

In FIG. 4, memory controller 104 receives a request for the same memory address x as described above with respect to FIG. 2. However, in FIG. 3, memory controller 104 has remapped the row associated with address x from row 114A, to spare row 116, which includes failure pattern 202.

In some examples, CPU 102 may cause memory controller 104 to change the address associated with spare row 116 from a first address to a second, different address. In some examples, CPU 102 may cause memory controller to associate spare row 116 associated with an address range (e.g. a starting and ending address range) or a plurality of address ranges. The address range(s) may be indicated by a command to PPR register 204.
[0044] FIG. 5 is a flowchart of an example method for performing fault injection using post-package repair technology. Method 500 may be described below as being executed or performed by a system, for example, system 100 of FIG. 1. Other suitable systems and/or computing devices may be used as well. Method 500 may be implemented in the form of executable instructions stored on at least one machine-readable storage medium of the system and executed by at least one processor of the system. Alternatively or in addition, method 500 may be implemented in the form of electronic circuitry (e.g., hardware). In alternate examples of the present disclosure, one or more blocks of method 500 may be executed substantially concurrently or in a different order than shown in FIG. 5. In alternate examples of the present disclosure, method 500 may include more or fewer blocks than are shown in FIG. 5. In some examples, one or more of the blocks of method 500 may, at certain times, be ongoing and/or may repeat.

[0045] Method 500 may start at block 502 and continue to block 504, where the system may map an address associated with a first memory row of a computing device to a spare memory row of the computing device. The spare memory row may have a memory failure pattern. At block 506, the system may read (e.g., via memory controller 104) data from the spare memory row. Method 500 may eventually continue to block 508, where method 500 may stop.

[0046] FIG. 6 is a flowchart of an example method for performing fault injection using post-package repair technology. Method 600 may be described below as being executed or performed by a system, for example, system 100 of FIG. 1. Other suitable systems and/or computing devices may be used as well. Method 600 may be implemented in the form of executable instructions stored on at least one machine-readable storage medium of the system and executed by at least one processor of the system. Alternatively or in addition, method 600 may be implemented in the form of electronic circuitry (e.g., hardware). In alternate examples of the present disclosure, one or more blocks of method 600 may be executed substantially concurrently or in a different order than shown in FIG. 6. In alternate
examples of the present disclosure, method 600 may include more or less blocks than are shown in FIG. 6. In some examples, one or more of the blocks of method 600 may, at certain times, be ongoing and/or may repeat.

[0047] Method 600 may start at block 602 and continue to block 604, where the system may map an address associated with a first memory row of a computing device to a spare memory row of the computing device. The spare memory row may have a memory failure pattern. At block 606, the system may read (e.g., via memory controller 104) data from the spare memory row.

[0048] At block 608, in some examples, memory controller 104 may generate a write request to an address associated with the row having the memory failure pattern before reading the data from the spare memory. At block 610, memory controller 104 may change at least one bit of the memory failure pattern.

[0049] At block 612, in some examples, CPU 102 may determine a response of memory controller 104 to memory failure pattern 118, and at block 614, may compare the response of memory controller 104 to predetermined response to memory failure pattern 118.

[0050] At block 616, memory controller 104 may change at least one bit of memory failure pattern 118. At block 618, memory controller 104 may map spare row 116 to a second address. Method 600 may eventually continue to block 618, where method 600 may stop.

[0051] FIG. 7 is a block diagram of an example system 700 for performing fault injection using post-package repair. System 700 may be similar to system 100 of FIG. 1 or of FIG. 2, for example. In the example of FIG. 7, system 700 includes a processor 710 and a machine-readable storage medium 720. Although the following descriptions refer to a single processor and a single machine-readable storage medium, the descriptions may also apply to a system with multiple processors and multiple machine-readable storage mediums. In such examples, the instructions may
be distributed (e.g., stored) across multiple machine-readable storage mediums and
the instructions may be distributed (e.g., executed by) across multiple processors.

[0052] Processor 710 may be one or more central processing units (CPUs),
microprocessors, and/or other hardware devices suitable for retrieval and execution
of instructions stored in machine-readable storage medium 720. In the particular
example shown in FIG. 7, processor 710 may fetch, decode, and execute instructions
722, 724, 726 to perform fault injection using post-package repair. As an alternative
or in addition to retrieving and executing instructions, processor 710 may include one
or more electronic circuits comprising a number of electronic components for
performing the functionality of one or more of the instructions in machine-readable
storage medium 720. With respect to the executable instruction representations (e.g.,
boxes) described and shown herein, it should be understood that part or all of the
executable instructions and/or electronic circuits included within one box may, in
alternate examples, be included in a different box shown in the figures or in a different
box not shown.

[0053] Machine-readable storage medium 720 may be any electronic, magnetic,
optical, or other physical storage device that stores executable instructions. Thus,
machine-readable storage medium 720 may be, for example, Random Access
Memory (RAM), an Electrically-Erasable Programmable Read-Only Memory
(EEPROM), a storage drive, an optical disc, and the like. Machine-readable storage
medium 720 may be disposed within system 700, as shown in FIG. 7. In this
situation, the executable instructions may be "installed" on the system 700.
Alternatively, machine-readable storage medium 720 may be a portable, external or
remote storage medium, for example, that allows system 700 to download the
instructions from the portable/external/remote storage medium. As described herein,
machine-readable storage medium 720 may be encoded with executable instructions
for injecting a memory failure pattern using post-package repair.

[0054] Referring to FIG. 7, memory request instructions 722, when executed by a
processor (e.g., 710), may cause system 700 to issue a post-package repair request
to a memory controller (e.g., memory controller 104) to remap a row of a memory to a spare row of the memory. The request disassociates the row from an address and associates the spare row (e.g. spare row 116) with the address. Failure pattern instructions 724, when executed by a processor (e.g., 710), may cause system 710 to write a failure pattern to a spare row of a memory using a post-package repair capability of the memory (e.g. system memory 106). Spare row read instructions 726, when executed by a processor (e.g., 710), may cause system 710 issue a read request to the memory controller for the address associated with the spare row (e.g., the address associated with spare row 116).

[0055] FIG. 8 is a block diagram of an example system 800 for using post package repair for mapping to a memory failure pattern. System 800 may be similar to system 100 of FIG. 1 or of FIG. 2, for example. In the example of FIG. 8, system 800 includes a memory controller 802, which is connected to a memory module 804 via memory bus 806.

[0056] Memory module 804 includes a row 806 and a spare row 810. Spare row 810 includes a memory failure pattern 812. In the example of FIG. 8, memory controller 802 may read, using post-package repair remapping, data from spare row 810 having memory failure pattern 812.
CLAIMS

1. A method comprising:
   mapping, using post-package repair, an address associated with a first memory row of a computing device to a spare memory row of the computing device,
   wherein the spare memory row has a memory failure pattern; and
   reading data from the spare memory row.

2. The method of claim 1 further comprising:
   generating a write request to an address associated with the spare row having the memory failure pattern before reading the data from the spare row.

3. The method of claim 1,
   changing at least one bit of the memory failure pattern.

4. The method of claim 1,
   determining a response of a memory controller to the memory failure pattern; and
   comparing the response of the memory controller to a predetermined response to the memory failure pattern.

5. The method of claim 1, further comprising:
   mapping the spare row of the computing device to a second address of the computing device.
6. A system comprising:
   a memory comprising a spare row that further includes a failure pattern,
   a memory controller to:
       read, using post-package repair remapping, data from the spare row
       of the memory having the failure pattern.

7. The system of claim 6, wherein the memory controller is to determine
   whether the failure pattern is correctable or uncorrectable.

8. The system of claim 7, further comprising at least one processor,
   wherein responsive to determining the failure pattern is correctable, the at
   least one processor is further to determine whether the memory controller
   corrected an error associated with the correctable failure pattern.

9. The system of claim 6, wherein the memory controller is to:
   use at least one of hard post-package repair or soft post-package repair
   responsive to determining that the failure pattern is uncorrectable.

10. The system of claim 6, further comprising at least one processor to:
    generate a write request at a particular memory address associated with the
    spare row; and
    responsive to generating the write request, generate at least one read
    request at the address associated with the spare row.

11. The system of claim 6, wherein the memory includes a post-package repair
    register, wherein the post-package repair register is used to change the failure
    pattern.

12. The system of claim 6, wherein the failure pattern comprises at least one of:
    a stuck-at one bit, or a stuck-at zero bit.
13. The system of claim 6, wherein the memory comprises a synchronous dynamic random access memory (SDRAM) module.

14. A non-transitory machine-readable storage medium including instructions stored thereon that, when executed, cause at least one processor to:
   - issue a post-package repair request to a memory controller to remap a row of a memory to a spare row of the memory, wherein the request disassociates the row from an address and associates the spare row with the address;
   - write a failure pattern to a spare row of a memory using a post-package repair capability of the memory; and
   - issue a read request to the memory controller for the address associated with the spare row.

15. The non-transitory machine-readable storage medium of claim 14, wherein the instructions that cause the at least one processor to issue a post-package repair request to the memory controller further cause the memory controller to issue a post-package repair request to a post-package repair control register of the memory.
500

502
START

504
MAP AN ADDRESS ASSOCIATED WITH A FIRST MEMORY ROW TO A SPARE MEMORY ROW, THE SPARE ROW HAVING A FAILURE PATTERN

506
READ DATA FROM THE SPARE MEMORY ROW

508
STOP

FIG. 5
602 START

604 MAP AN ADDRESS ASSOCIATED WITH A FIRST MEMORY ROW TO A SPARE MEMORY ROW, THE SPARE ROW HAVING A FAILURE PATTERN

606 READ DATA FROM THE SPARE MEMORY ROW

608 GENERATE A WRITE REQUEST TO AN ADDRESS ASSOCIATED WITH THE SPARE ROW BEFORE READING THE DATA FROM THE SPARE ROW

610 CHANGE AT LEAST ONE BIT OF THE MEMORY FAILURE PATTERN

612 DETERMINE A RESPONSE OF THE MEMORY CONTROLLER TO THE MEMORY FAILURE PATTERN

614 COMPARE THE RESPONSE OF THE MEMORY CONTROLLER TO A PREDETERMINED RESPONSE

616 CHANGE AT LEAST ONE BIT OF THE MEMORY FAILURE PATTERN

618 MAP THE SPARE ROW TO A SECOND ADDRESS

618 STOP

FIG. 6
FIG. 7
FIG. 8
INTERNATIONAL SEARCH REPORT

International application No. PCT/US2015/045683

A. CLASSIFICATION OF SUBJECT MATTER
G11C 29/08; 2006.01.01

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)
G11C 29808; G01R 11/28; G11C 11/15; G06F 11/00; G11C 29/04; G06F 11/20

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: post-package repair, address, spare memory, failure pattern, mapping, write request, read request, and similar terms.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>US 2006-0036921 AI (BRIAN P. CALLAWAY et a1.) 16 February 2006 See paragraphs [0025]-[0027] ; and figure 1.</td>
<td>1, 6</td>
</tr>
<tr>
<td>A</td>
<td>US 2004-0246774 AI (ANDREW L. VAN BROCKLIN et a1.) 09 December 2004 See paragraphs [0022]-[0040] ; [0043][0067] ; and figures 1, 4-6.</td>
<td>2-5, 7-15</td>
</tr>
<tr>
<td>A</td>
<td>US 2006-0046418 AI (CHENG-WEN FU et a1.) 23 March 2006 See paragraphs [0040]-[0056] ; claim 1; and figures 3-6.</td>
<td>1-15</td>
</tr>
<tr>
<td>A</td>
<td>US 2013-0021859 AI (VOLODYMYR SHVYDUN et a1.) 24 January 2013 See paragraphs [0017]-[0028] ; claim 1; and figures 1A-2C.</td>
<td>1-15</td>
</tr>
<tr>
<td>A</td>
<td>US 2012-0266016 AI (TSUNG-CHU HUANG) 18 October 2012 See paragraph [0042] ; claim 1; and figure 4.</td>
<td>1-15</td>
</tr>
</tbody>
</table>

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

Date of the actual completion of the international search
23 June 2016 (23.06.2016)

Date of mailing of the international search report
28 June 2016 (28.06.2016)

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea

Facsimile No. +82-42-481-8578

Authorized officer
B YUN, Sung Cheal

Telephone No. +82-42-481-8262

Form PCT/ISA/210 (second sheet) (January 2015)
<table>
<thead>
<tr>
<th>Patent document cited in search report</th>
<th>Publication date</th>
<th>Patent family member(s)</th>
<th>Publication date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>US 2005-0083739 Al</td>
<td>21/04/2005</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 6829176 B2</td>
<td>07/12/2004</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 6879530 B2</td>
<td>12/04/2005</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 6977855 B2</td>
<td>20/12/2005</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 7212456 B2</td>
<td>01/05/2007</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 7468922 B2</td>
<td>23/12/2008</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 6839275 B2</td>
<td>04/01/2005</td>
</tr>
<tr>
<td>US 2006-0064618 Al</td>
<td>23/03/2006</td>
<td>TW 1252397 B</td>
<td>01/04/2006</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 7228468 B2</td>
<td>05/06/2007</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 7644323 B2</td>
<td>05/01/2010</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 2013-0001861 Al</td>
<td>24/01/2013</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 2013-0301369 Al</td>
<td>14/11/2013</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 8509014 B2</td>
<td>13/08/2013</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 8605527 B2</td>
<td>10/12/2013</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 8760949 B2</td>
<td>24/06/2014</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 8873318 B2</td>
<td>28/10/2014</td>
</tr>
<tr>
<td>US 2012-0266016 Al</td>
<td>18/10/2012</td>
<td>TW 201222248 A</td>
<td>01/06/2012</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TW 1439857 B</td>
<td>01/06/2014</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 8522072 B2</td>
<td>27/08/2013</td>
</tr>
</tbody>
</table>