



159159



G00002

SYSTEM AND METHOD FOR IMPLEMENTING A MEMORY DEFECT MAP

TECHNICAL FIELD

5 The present disclosure relates generally to computing systems and information handling systems, and, more particularly, to a system and method for creating a map of memory which governs its use by the computing system or information handling system.

BACKGROUND

10 As the value and use of information continues to increase, individuals and businesses seek additional ways to process and store information. One option available to these users is an information handling system. An information handling system generally processes, compiles, stores, and/or communicates information or data for business, personal, or other purposes thereby allowing users to take advantage of the value of the information. Because technology and information handling needs and requirements vary between different users or applications, information handling systems may vary with respect to the type of information handled; the methods for handling the information; the methods for processing, storing or communicating the information; the amount of information processed, stored, or communicated; and the speed and efficiency with which the information is processed, stored, or communicated. The variations in information handling systems allow for information handling systems to be general or configured for a specific user or specific use such as financial transaction processing, airline reservations, enterprise data storage, or global communications. In addition, information handling systems may include or comprise a variety of hardware and software components that may be configured to process, store, and communicate information and may include one or more computer systems, data storage systems, and networking systems.

25 An information handling system will typically include some type of temporary information storage medium, such as random access memory. The amount of memory included in an information handling system may be on the order of gigabytes. As memory size increases,

the likelihood that part of the memory will either be manufactured defective or become defective over time increases. If left unmanaged, the presence of defective memory cells, regardless of their size, can cause the information handling system to fail. Such failure can initiate an abrupt end to the current operation of the information handling system, resulting in the loss of critical data. A memory failure could also prevent the information handling system from starting up altogether.

As information handling systems continue to evolve and computer technology advances, the operational relationship between the CPU and memory becomes more significant and complex. Many attributes of modern systems (specifically, the introduction of multi-core processors and virtualization) are contributing to an ever-larger memory footprint within a typical information handling system. Consequently, not only is system memory becoming a much more substantial percentage of the overall cost of the information handling solution, the impact of erroneous behavior in the memory can have a much more adverse effect on the life cycle expense associated with the information handling system.

SUMMARY

In accordance with the present disclosure, a system and method are herein disclosed for managing memory defects in an information handling system. In an information handling system, a first quantity of memory, such as RAM, may contain defective memory
5 elements. A second quantity of memory, such as EEPROM, is physically coupled to the first quantity of memory and is used to store a memory defect map containing information regarding the location of defective memory elements in the first quantity of memory. The memory defect map may then be referenced by the BIOS or the operating system to preclude use of regions of memory containing defective memory elements.

10 The system and method disclosed herein are technically advantageous because a mechanism is provided for reducing the functional and financial impact of defective memory elements. In particular, the present disclosure provides a way for an information handling system to operate normally, despite the presence of one or more defective memory elements, with minimal impact on system performance. By effectively hiding defective memory blocks from the
15 operating system, crashes of the information handling system are better avoided. In addition, by sequestering defective memory elements, the present disclosure extends the life of memory. Overall system costs for the information handling system are correspondingly reduced as memory need not be replaced even after multiple memory elements are found to be defective. Other technical advantages will be apparent to those of ordinary skill in the art in view of the
20 following specification, claims, and drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

A more complete understanding of the present embodiments and advantages thereof may be acquired by referring to the following description taken in conjunction with the accompanying drawings, in which like reference numbers indicate like features, and wherein:

5 Figure 1 is a block diagram of a typical information handling system;

 Figure 2A shows the addressable memory space for a typical information handling system;

 Figure 2B shows the memory space included in a usable memory map which excludes defect blocks;

10 Figures 3A and 3B depict the results of reducing fragmentation in a usable memory map;

 Figure 4 is a block diagram of two dual-inline memory modules; and

 Figure 5 shows a flow diagram from system initialization to the creation of a usable memory map.

DETAILED DESCRIPTION

For purposes of this disclosure, an information handling system may include any instrumentality or aggregate of instrumentalities operable to compute, classify, process, transmit, receive, retrieve, originate, switch, store, display, manifest, detect, record, reproduce, handle, or
5 utilize any form of information, intelligence, or data for business, scientific, control, or other purposes. For example, an information handling system may be a personal computer, a network storage device, or any other suitable device and may vary in size, shape, performance, functionality, and price. The information handling system may include random access memory (RAM), one or more processing resources such as a central processing unit (CPU) or hardware or
10 software control logic, ROM, and/or other types of nonvolatile memory. Additional components of the information handling system may include one or more disk drives, one or more network ports for communication with external devices as well as various input and output (I/O) devices, such as a keyboard, a mouse, and a video display. The information handling system may also include one or more buses operable to transmit communications between the various hardware
15 components.

Shown in Figure 1 is a block diagram of a typical information handling system. A processor or CPU 12 of the typical information handling system 10 is communicatively coupled to a memory controller hub or north bridge 30. Memory controller hub 30 is coupled to RAM 20 and a graphics processing unit 40. Memory controller hub 30 is also coupled to an I/O controller
20 hub or south bridge 60. I/O hub 60 is coupled to storage elements of the computer system, including a storage element 50, which may comprise a flash ROM that includes the BIOS of the computer system. I/O hub 60 is also coupled to the hard drive 70 of the computer system. I/O hub 60 may also be coupled to a Super I/O chip 80, which is itself coupled to several of the I/O ports of the computer system, including keyboard 90, mouse 100, and one or more parallel ports.

25 The system and method described herein involve the management of the memory resources of an information handling system. The system and method involve the use of storage included within memory devices to identify which, if any, of the physical memory of the memory

device is unusable by the system. If physical memory is identified as being unusable by the system, the physical memory is logically mapped out of a resulting memory map of the usable computer system, resulting in a memory map of usable memory that does not include the defective physical memory of the memory device.

5 At initialization, the information handling system will create a map of the memory or RAM of the system. Figure 2A is a diagram of the memory space that is reviewed by a memory mapping call, such as an E820 call, within the information handling system. The information handling system 10 may be comprised of numerous distinct components capable of being addressed as memory, including dual-inline memory modules (DIMMs) 120 and PCI
10 devices 130, such as discrete graphics cards. Devices that include addressable memory are detected during the BIOS initialization period, and the memory within these devices is mapped and identified as system-accessible memory during the memory mapping call. A memory mapping call results in the iterative interrogation of possible memory locations, beginning with memory having the lowest possible address within the system 110. The memory mapping call
15 reviews all addressable memory locations, including memory blocks 124, 125 and 126, even though memory block 125 contains a memory defect. As system-accessible memory is located, including DIMMs 120, a map is created of usable, non-defective memory.

 As the usable memory map is created, portions of physical memory that are determined to be unavailable or unusable are excluded from the map, resulting in the creation of
20 a logical map that only includes usable and non-defective memory locations. Figure 2B is a diagram of a usable memory map in which the unusable locations in the memory of the computer system have been logically mapped out of the computer system. For example, memory blocks 124 and 126 are included in the usable memory map, but memory block 125, which contains a defective memory element, is not.

25 The usable memory map is acquired by the operating system, either as a result of a set of iterative memory mapping calls by the operating system or as a result of a memory mapping function in which the BIOS transmits the completed map to the operating system.

Because the map excludes from its scope those physical memory locations that are determined to be unavailable or defective, the operating system is only able to access the memory that is mapped as being usable. Physical memory that is mapped as being unusable cannot be used by the operating system and can therefore be described as being not visible by the operating system.

5 During the creation of the usable memory map, information regarding the location of usable memory regions is sent to the operating system. This information may take the form of a starting address, the size of the memory block in question, and the type of memory at that location. As defective memory blocks are carved or mapped out of the memory map, the memory map is divided into more pieces and the number of discrete entries that are required to
10 describe the memory available (or unavailable) to the system increases, requiring additional storage and resulting in the fragmentation of memory. A number of strategies can be employed to reduce the administrative burden of identifying an excessive number of defective entries and usable entries.

 One strategy for reducing the administrative burden of identifying an excessive
15 number of defective entries and usable entries involves grouping proximate defective memory cells into a common defective entry. Figures 3A and 3B illustrate DIMMs 300 divided into memory blocks 310. A memory block is a defined section of contiguous storage within a memory device. Included within DIMMs 300 are defect blocks 320 and 340. A defect block is a memory block that includes at least some defective memory within the block. A defect block
20 may include a mixture of defective memory and non-defective or usable memory. If defect blocks 320 and 340 were individually mapped out of the usable memory map, each block would be identified by a starting address and an ending address. Such fragmentation may be ameliorated by detecting the presence of small blocks of usable memory between defective memory blocks and grouping the small usable memory block together with the adjacent defective
25 memory blocks and designating the entire group as unusable. This excludes additional non-defective memory from use by the information handling system, but it reduces the number of addresses that must be stored as part of the usable memory map.

As shown in Figure 3B, defect blocks 320 and 340 can be grouped into one larger defect block which includes non-defective memory block 330. Since only a single defect block is identified to the operating system, a single usable memory map entry is sufficient to map out both memory defects. To accomplish this reduction in fragmentation, the memory defect map is searched for defect blocks within some specified number of memory blocks from each other. When two such blocks are found, any intervening blocks are marked as defective. This information is subsequently stored in the memory defect map. When the information handling system is next initialized, the usable memory map will include only one entry for the contiguous memory blocks marked as defective, including the intervening non-defective memory block.

Figure 4 presents a block diagram of memory elements within DIMMs 400. A DIMM may include both RAM 410 and a quantity of electrically-erasable programmable read-only memory (EEPROM) 420, which comprises non-volatile storage that is included on DIMM 400. EEPROM 420 can store serial presence detect (SPD) information, which is data that describes the size, speed, electrical requirements, and other information about the RAM 410. Although the standards of the JEDEC Solid State Technology Association specify that the lower 128 bytes 430 of the EEPROM are reserved for SPD information, DIMMs typically include an additional instance of nonvolatile memory 440 that is not dedicated to SPD information and is referred to herein as memory defect storage 440.

Memory defect storage 440 of EEPROM 420 is used to store information describing the location of defective memory in RAM 410. When a memory cell is identified as defective or if it is otherwise determined that the system should not be granted access to that cell, a block of memory containing the memory cell in question is identified and the location of that memory is stored in memory defect storage 440. The placement of the defect memory storage in a DIMM has advantages in that the memory location within the DIMM is widely implemented in information handling systems and is available for the storage of this information. Moreover, the placement of the memory defect map within the DIMM itself is advantageous, as the memory defect information remains with the DIMM and allows the defect information for the DIMM to

travel with the DIMM in the event that the DIMM moves from one computer system to another. Although memory defect storage 440 serves a dedicated location for the identification of defective memory locations, it is not necessary that the defect map be stored in a nonvolatile location within EEPROM 420. Rather, any form of non-volatile storage that is accessible by the computer system could be used to store the record of defective memory locations.

The content of the memory defect storage location is a memory defect map, with each entry in the map corresponding to a region of the physical address on the associated DIMM 400. The size of each designated region in the map is defined by the memory capacity of the DIMM amortized across the number of entries permitted in the map. The size of memory block designated as defective is thus equal to the DIMM capacity divided by the number of bits in the defect map. The larger the available space for the memory defect map, the smaller the size of the memory block that can be designated as defective.

In one embodiment, each bit in the memory defect map in memory defect storage corresponds to one memory block in the DIMM. The size of the memory defect map can be adjusted as appropriate either to minimize the size of the memory map or to maximize the resolution of the memory map, which results in more usable memory. If storage space for the memory map is at a premium, the size of the map can be reduced by simply associating each bit in the memory map with a larger memory block in the RAM of the DIMM. If the maximization of usable memory is a more important consideration, the resolution and size of the memory map may be increased, resulting in finer granularity in identifying the defective memory cells, thereby resulting in fewer usable memory cells being grouped with the defective cell in each memory block and designated unusable as a result. The grouping of memory cells that are designated as being unusable, including defective memory cells and usable memory cells grouped with the defective memory cells, is known as the defect block. As an example, in a system that includes 4GB of RAM and the defect memory map is constrained to be 80 bytes (640 bits) in size, each time a defective memory cell is detected, 6.4MB (4GB/80 bytes) of memory is blocked out as unusable, even if only one cell is identified as being defective. Thus, increasing the size of

the defect memory map may result in fewer usable memory cells being grouped with the defective memory cell and being designated as unusable.

In another embodiment, the defect map may be implemented with one byte per entry. This increases the size of the map considerably, but results in much less wasted memory for each defective cell. The defect map is initially indexed such that each entry in the defect map corresponds to one region of memory on the DIMM. Each entry in the defect map is eight bits in length. One bit of each eight-bit entry flags the region as one containing a defective bit. The remaining seven bits of each entry are used as an address for one or more sub-blocks within the designated memory region. Thus, each memory region corresponding to each eight-bit entry in the defect map is thus divided into 2^7 (128) sub-blocks, and seven bits of each entry are used as an address to point to the specific sub-block within the larger region that contains the defective memory cell. This type of map has the same trade-off between granularity and map size, but it provides an effective way to manage defect maps with much higher granularity.

Another embodiment of a defect map within a DIMM can be implemented when both high granularity and smaller map size are desired, if the number of defective regions is capped at a specific number. In this type of defect map, the size of each entry in the memory defect map is maximized based on the space available for the map and the number of defective regions chosen. For instance, if the SPD has sixty-four bytes (512 bits) available for the defect map and a limit of forty defective regions is chosen, each entry would be 512 bits divided by the forty available entries, resulting in 16 bits per entry. Unlike other examples, no bits are used to flag a region as defective or usable. Instead, all the bits are used as an offset pointer from the lowest memory address on the DIMM. Entries are populated serially as memory defects are found, and each populated entry indicates that the region beginning at the pointer contains a defective memory cell. This type of map has the advantages of increased granularity (resulting in less wasted memory per defect) and small map size, but the memory cannot support more than the chosen number of defective regions. Any additional defects would not be incorporated into the map, and could either be left unmanaged or could be managed by some other method.

An information handling system may include multiple DIMMs. During operation of the information handling system, the memory defect map that is included in the memory storage locations of each of the DIMMs may be combined in the memory of the information handling system into a global memory defect map that identifies the defective memory of the entire information handling system. Thus, during operation of the information handling system, locations of defective memory are known and defective memory locations can be excluded during computing operations. If additional defective memory locations are discovered during operation of the information handling system, these locations are added to the global memory defect map on a real-time basis. If a defective memory location is discovered during operation of the information handling system, a defective memory pointer to the block of defective memory is established in system memory at memory address that is above or higher than the memory space of the DIMMs of the information handling system. Depending on the number of DIMMs in the information handling system, the first one, two, or three bits of the entry are a DIMM index that identifies the DIMM that includes the defective memory block. The remaining bits of the defective memory pointer are a block index that identifies the location of the defective memory block within the affected DIMM. The block index is created by dividing the physical address of the defective memory cell on the DIMM by the defined block size of the defect block that has been established for the DIMMs of the information handling system. In this manner, an index is created, comprising an entry in memory that includes (a) an index to the DIMM that includes a defect and (b) an index to the memory block within the DIMM. After this pointer is created, the pointer is added to the global memory defect map, resulting in the defective memory block being excluded from operations in the information handling system.

Figure 5 is a flow diagram of the creation of a usable memory map. After the information handling system is initialized at step 500, the power-on self-test (POST) phase begins at step 510. During POST, memory devices are identified and the defect maps for memory are retrieved from each respective device (step 520). A defect map is a memory map of defective memory locations at step 530. At step 540, a usable memory map is created which

excludes the memory locations that are identified in the memory defect map. Thus, even though a memory device includes defective memory, the locations of the defective memory are mapped out of the usable memory map that is created by the operating system.

5 Although this disclosure has been described with respect to the operation of DIMMs within an information handling system, it should be recognized that the memory system and method described herein may be implemented with any physical storage device that is potentially subject to defects. Consistent with this disclosure, for example, a magnetic storage device may be coupled to other non-volatile storage which contains a defect map for the magnetic storage device.

10 Although the present disclosure has been described in detail, it should be understood that various changes, substitutions, and alterations can be made hereto without departing from the spirit and the scope of the invention as defined by the appended claims.

WHAT IS CLAIMED IS:

1. A system for managing memory defects, comprising:
a first quantity of memory;
5 a second quantity of memory;
a memory defect map, wherein said memory defect map contains information
regarding the first quantity of memory and is stored in the second quantity of memory.
2. The system of claim 1, wherein the second quantity of memory is nonvolatile.
10
3. The system of claim 1, wherein the first and second quantities of memory are
physically coupled.
4. The system of claim 1, wherein the first quantity of memory is random access
15 memory.
5. The system of claim 1, wherein the second quantity of memory is read-only
memory.
- 20 6. The system of claim 1, wherein a portion of the read-only memory is reserved for
serial presence detect information.
7. The system of claim 1, wherein the first and second quantities of memory are
comprised in a dual inline memory module.

8. A method for managing memory defects, comprising:
detecting defective portions of a first quantity of memory;
creating a memory defect map;
storing the memory defect map; and
5 restricting access to the first quantity of memory by an information handling
system according to the information stored in the memory defect map
9. The system of claim 8, wherein the memory defect map is stored in a second
quantity of memory.
10. The system of claim 8, wherein the memory defect map and the first quantity of
memory are physically coupled.
11. The system of claim 8, wherein the memory defect map is used to prevent access
15 to defective memory by an operating system running on the information handling system.

12. A method for creating a memory defect map, comprising:
 - determining the quantity of storage available for a memory defect map;
 - determining a quantity of memory to be mapped by the memory defect map;
 - dividing the quantity of memory to be mapped into a number of regions
 - 5 corresponding to the quantity of storage available for the memory defect map; and
 - storing, in entries of the memory defect map corresponding to the regions of memory to be mapped, information regarding the presence of defective memory in each respective region of memory.
- 10 13. The method of claim 12, wherein each entry comprises one bit.
14. The method of claim 12, wherein each entry comprises one byte.
- 15 15. The method of claim 12, wherein a single bit of each entry indicates the presence of defective memory within the corresponding region of memory.
16. The method of claim 12 further comprising the step of:
 - storing, in the available bits of each entry in the memory defect map, address
 - information specifying more particularly the location of the defective memory within the
 - 20 associated memory region.

17. A method for creating a memory defect map, comprising the steps of:
determining the quantity of memory to be addressed by the memory defect map;
determining a maximum number of defective regions of memory to be addressed
by the memory defect map; and
5 storing a reference address to the location of each defective memory region until
the maximum number of defective regions of memory has been reached.
18. The method of claim 17, further comprising the steps of:
determining the quantity of storage available for a memory defect map;
10 determining the maximum storage space available for each reference address
based on the quantity of storage available for the memory defect map and the maximum number
of defective regions of memory to be addressed; and
addressing the smallest memory regions possible based on the quantity of storage
used for each reference address and the quantity of memory to be addressed.
15

19. A method for reducing the fragmentation of a memory defect map, comprising the steps of:
 - searching a memory defect map for defect blocks within a specified number of memory blocks of each other; and
 - 5 modifying the memory defect map to designate intervening memory blocks as defective.
20. The method of claim 19, further comprising the step of:
 - creating a usable memory map based on the modified memory defect map.