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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
12 September 2014 (12.09.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/134808 A1

(51) International Patent Classification:
G06F 21/00 (2013.01)

(21) International Application Number:
PCT/CN2013/072294

(22) International Filing Date:
7 March 2013 (07.03.2013)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **INTEL CORPORATION** [US/US]; 2200 Mission College Blvd., Santa Clara, California 95052 (US).

(72) Inventors; and
(71) Applicants *(for US only)*: **YAO, Jiewen** [CN/CN]; Room 404, Building 3, 818 Ming Sheng Road, Shanghai 200135 (CN). **ZIMMER, Vincent J.** [US/US]; 1937 S. 369th St. Federal Way, Washington 98003 (US).

(74) Agent: **NTD PATENT AND TRADEMARK AGENCY LIMITED**; 10th Floor, Block A, Investment Plaza, 27 Jinrongdajie, Xicheng District, Beijing 100033 (CN).

(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, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, 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, 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, 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, ML, MR, NE, SN, TD, TG).

Published:

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

(54) Title: MECHANISM TO SUPPORT RELIABILITY, AVAILABILITY, AND SERVICEABILITY (RAS) FLOWS IN A PEER MONITOR

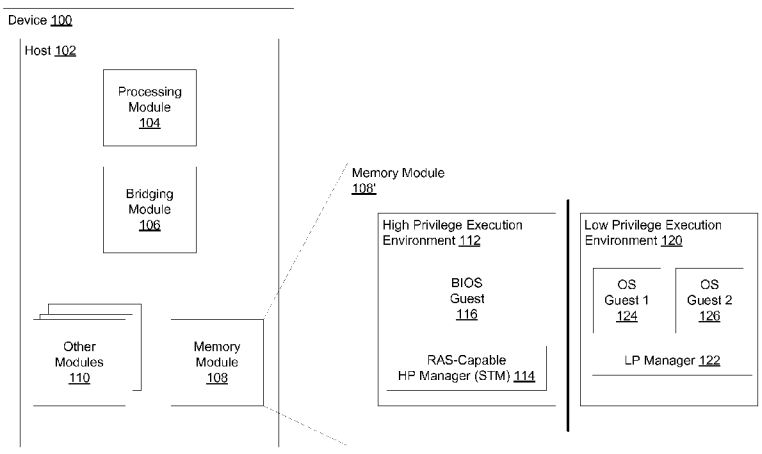

Figure 1

(57) Abstract: A mechanism to support reliability, availability, and serviceability (RAS) flows in a peer monitor is disclosed. A method of the disclosure includes receiving, by a processing device, a system management interrupt (SMI) event. The method further includes invoking, in response to the SMI event, a privilege manager to execute from a read-only memory (ROM) entry point to handle the SMI event, the privilege manager comprising a hot plug service module to provide support for memory hot plug functionality and processor hot plug functionality.

WO 2014/134808 A1

Mechanism to Support Reliability, Availability, and Serviceability (RAS) Flows in a Peer Monitor

TECHNICAL FIELD

[0001] The embodiments of the disclosure relate generally to computer security and, more specifically, relate to a mechanism to support reliability, availability, and serviceability (RAS) flows in a peer monitor.

BACKGROUND

[0002] SMM is a mode of operation where all normal execution (including the OS) is suspended, and special separate software (usually firmware or a hardware-assisted debugger) is executed in a high-privilege mode. SMM provides an isolated memory and execution environment, and SMM code is invisible to the OS while retaining full access to host physical memory and complete control over peripheral hardware. When SMM is initiated, the current state of the processor is saved and all other processes are stopped. High privileged operations may be performed in SMM mode, such as debugging, hardware management, security functions, emulation, etc., followed by the computing device resuming operation based on the save state of the processor.Upon occurrence of an SMI, the computing device may enter the SMM.

[0003] Vulnerabilities in SMM code implementations have led to the introduction of some current security schemes in computing devices, where software critical to device

operation is protected through segregation. For example, in a virtual machine (VM) environment, such as, for example, Virtualization Technology (VT) functionality incorporated on some processing devices, one or more machine managers may control VMs operating in different operational environments. For example, VT defines a primary monitor mode where virtual machine managers (VMM) (also known as hypervisors) are able to de-privilege guest operating systems (OS). Similarly, VT also provides a system management mode transfer monitor (STM) that can de-privilege a System Management Interrupt (SMI) handler, such that the SMI handler runs as a guest of the STM in system management mode (SMM).

[0004] However, current implementations of the STM lack support for reliability, availability, and serviceability (RAS). RAS is a set of related attributes used to describe a multitude of features that protect data integrity and enable a computer system to stay available for long periods of time without failure. RAS attributes may be considered when designing, manufacturing, purchasing, or using a computer product or component. Current STM implementations do not meet RAS requirements when supporting hot plug, read-only memory (ROM) SMI handler feature, and other RAS actions.

[0005] For example, in the case of hot plug support, current STM implementations do not provide for CPU or memory hot plug support. In the case of ROM SMI handler feature support, current STM implementations only support STM on dynamic random access memory (DRAM).

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] The disclosure will be understood more fully from the detailed description given below and from the accompanying drawings of various embodiments of the disclosure. The drawings, however, should not be taken to limit the disclosure to the specific embodiments, but are for explanation and understanding only.

[0007] **Figure 1** is a block diagram of a device supporting Reliability, Availability, Serviceability (RAS) flows in a peer monitor according to an embodiment of the disclosure;

[0008] **Figure 2** is a block diagram illustrating a memory module to implement ROM-based execution of a peer monitor according to an embodiment of the disclosure;

[0009] **Figure 3** is a flow diagrams of a method for Read-Only Memory (ROM)-based execution of a peer monitor to support RAS according to an embodiment of the disclosure;

[0010] **Figure 4** is a block diagram illustrating hot plug support for RAS by a peer monitor in accordance with embodiments of the disclosure;

[0011] **Figures 5A** and **5B** are flow diagrams illustrating methods for hot plug support by an STM when adding or removing memory according to an embodiment of the disclosure;

[0012] **Figures 6A** and **6B** are flow diagrams illustrating a method for hot plug support by an STM when adding or removing processors according to an embodiment of the disclosure; and

[0013] **Figure 7** illustrates a block diagram of one embodiment of a computer system.

DETAILED DESCRIPTION

[0014] Embodiments of the disclosure provide for a mechanism to support reliability, availability, and serviceability (RAS) flows in a peer monitor. In one embodiment, a RAS-capable high privilege (HP) manager acts as a peer monitor, such as a System Management Interrupt (SMI) Transfer Monitor (STM). The STM may de-privilege an SMI handler, so that the SMI handler runs as a guest of the STM in system management mode (SMM). Because current implementations of the STM lack support for RAS flows, embodiments of the disclosure implement a RAS-capable HP manager that is configured to support features of RAS. In particular, the RAS-capable HP manager, such as an STM, may implement ROM-based execution and support for hot plug functionality.

[0015] In one embodiment, a method of the disclosure includes receiving, by a processing device, a system management interrupt (SMI) event. The method further includes invoking, in response to the SMI event, a privilege manager to execute from a read-only memory (ROM) entry point to handle the SMI event, the privilege manager comprising a hot plug service module to provide support for memory hot plug functionality and processor hot plug functionality.

[0016] **Figure 1** is a block diagram of a device 100 supporting RAS flows in a peer monitor according to an embodiment of the disclosure. Some examples of device 100 may include, but are not limited to, a mobile communications device such as a cellular handset or smart phone, a mobile computing device such as a tablet computer, a netbook, a notebook computer, a laptop computer, a desktop computer, a server computer, and so on.

[0017] Device 100 may include, for example, host 102 to handle baseline operations for device 100. Host 102 may include, for example, a processing module 104, bridging module 106, memory module 108, and other modules 110. Processing module 102 may comprise one or more processors (also known as processing devices) situated in separate component, or alternatively, one or more processing cores embodied in a single integrated circuit (IC) arranged, for example, in a System-on-a-Chip (SOC) configuration.

[0018] Bridging module 106 may include circuitry configured to support processing module 104. Example circuitry may include interface/bridging circuitry (e.g., a group of integrated circuits (ICs)) that may be configured to handle communications using various buses in device 100. For example, bridging module 106 may handle signaling between the various modules by converting from one type/speed of communication to another, and may also be compatible with a variety of different devices to allow for different system implementations, upgrades, etc. Some of the functionality of bridging module 106 may also be incorporated into processing module 104, memory module 108, or other modules 110.

[0019] Processing module 104 may execute instructions. Instructions may include program code to cause processing module 104 to perform activities such as, but not limited to, reading data, writing data, processing data, formulating data, converting data, transforming data, etc. Information, including instructions, data, etc. may be stored in memory module 108.

[0020] Memory module 108 may include random access memory (RAM) or read-only memory (ROM) in a fixed or removable format. RAM may include memory to hold information during the operation of the device 100 such as, for example, static RAM

(SRAM) or dynamic RAM (DRAM). ROM may include memories such as computing device BIOS memory to provide instructions when device 100 activates, programmable memories such as electronic programmable ROMs (EPROMs), Flash, etc. Other fixed and/or removable memory may include magnetic memories such as floppy disks, hard drives, etc., electronic memories such as solid state Flash memory (e.g., eMMC, etc.), removable memory cards or sticks (E.g., USB, micro-SD, etc.), optical memories such as compact disc-based ROM (CD-ROM), holographic, etc.

[0021] Other modules 110 may include modules directed to supporting other functionality within device 100. Other modules 110 may include, for example, modules to supply power to device 100, modules to support wired and/or wireless communications in device 100, modules to provider user interface (UI) features in device 100, modules to support specialized functionality, and so on. The composition of other modules 100 may be variable depending upon, for example, form factor, the use for which device 100 has been configured, and so on.

[0022] An embodiment of memory module 108 according to an embodiment of the disclosure is shown in a blown-up view at 108'. Memory module 108' may include a high privilege execution environment 112 and a low privilege execution environment 120. Software running in high privilege execution environment 112 may be able to affect the operation of other software in device 100 (e.g., may be able to read, write, and/or execute software in low privilege execution environment 120), but software running in low privilege execution environment 120 cannot affect any software running in high privilege execution environment 112. High privilege execution environment 112 may include a Reliability,

Availability, Serviceability (RAS)-capable high privilege (HP) manager 114 to manage the operations of BIOS guest 116 and other guests 118. Low privilege execution environment 120 may include a low privilege (LP) manager 122 to manage the operations of OS guest 1 124 and OS guest 2 126. While two OS guests 124, 126 are shown, embodiments consistent with the disclosure are not limited to only two guests.

[0023] In at least one embodiment, certain activities in high privilege execution environment 112 may occur when device 100 enters a particular security mode. In this security mode, all other processing activity may be discontinued in processing module 104, the current context of processing module 104 may be saved, and then any operations related to high privilege execution environment 112 may be carried out prior to returning to normal operation of device 100. This security mode may be configured by RAS-capable HP manager 114.

[0024] In embodiments of the disclosure, the RAS-capable HP manager 114 may be a peer monitor, such as a System Management Interrupt (SMI) Transfer Monitor (STM). The STM 114 may de-privilege a System Management Interrupt (SMI) handler, so that the SMI handler runs as a guest of the STM 114 in system management mode (SMM). Embodiments of the disclosure provide an STM 114 that is considered RAS-capable because it supports features of RAS. In particular, the RAS-capable HP manager 114 may implement ROM-based execution of the STM 114 and STM 114 support for hot plug functionality. Hot plug describes the functions of replacing or adding computer system components without shutting down or rebooting the system. **Figures 2** and **4**provide further

detail of the RAS features of ROM-based execution and hot plug that are supported by STM 114 in embodiments of the disclosure.

[0025] **Figure 2** is a block diagram illustrating a memory module 200 to implement ROM-based execution of a peer monitor according to an embodiment of the disclosure. In one embodiment, the memory module 200 is the same as memory module 108 described with respect to **Figure 1**. In another embodiment, the peer monitor is an STM, such as STM 114 described with respect to **Figure 1**. Memory module 200 may include both a ROM 210 region and a RAM 220 region.

[0026] Embodiments of the disclosure place an STM image 211 in ROM 210. In addition, an STM page table and global description table (GDT) 212 are created and placed in ROM 210 with the STM image 211. The STM page table and the STM GDT 212 enable the STM image 211 to be run in long mode (e.g., x64 bit mode). In one embodiment, the access bit and dirty bit for the STM page table and GDT 212 are set by default. ROM STM 211 may also include an STM header 214 that stores the page table and GDT 212 for ROM STM 211.

[0027] In one embodiment, the ROM STM may be initialized during a boot-up process of the BIOS of the computer system. A BIOS SMM module (not shown) may place an address of the ROM STM image 211 in a special register of a processor running the BIOS and STM code. This ROM STM image 211 address is the entry point 215 for the ROM STM image 211, where a stack 213 for the STM code is set up for execution of the STM code. In one embodiment, the special register storing the ROM STM entry point 215 is an IA32_SMM_MONITOR MSR.

[0028] At runtime, when a SMI occurs, the processor may invoke the ROM STM entry point 215 first (using the address at the special register). The ROM STM 211 may then check whether a memory error occurs at the RAM 220 region of the memory module 200. If the memory check reveals that there are no errors in RAM 220, then operation of the STM code may transfer to RAM STM 221 by invoking a RAM STM entry point 225. In one embodiment, the RAM STM entry point 225 is stored in the ROM STM 211. In one embodiment, the RAM STM 221 resides in a region of RAM 220 that the STM code typically resides, such as in a monitor segment (MSEG) region of a top segment (TSEG) of RAM 220.

[0029] On the other hand, if the ROM STM 211 detects that there is a RAM error, then the ROM STM 211 cannot invoke the RAM STM 221. Instead, ROM STM 211 can call a BIOS ROM SMM handler directly to handle the corrupt RAM 220 or handle the error otherwise (according to BIOS SMM handler choice). In one embodiment, an entry point for the BIOS ROM SMM handler may be defined in another special register, such as a TXT DESC SMM SAVE STATE MSR.

[0030] In embodiments of the disclosure, an STM transition state data structure 222 is used to transition between the ROM STM 211 and the RAM STM 221. In one embodiment, the STM transition state data structure 222 is included at the beginning of a MSEG portion of RAM 220. During initialization of a system having the memory module 200, a BIOS SMM may determine a state of the ROM STM 211 and a state of the RAM STM 221. The transition data structure may include, but is not limited to, the size of the ROM STM, the size of the RAM STM, a CRC, check, hash or other cryptographic marker.

The transition data structure may be utilized by the ROM STM 211 to verify the integrity of the RAM STM prior to passing control.

[0031] The ROM-based STM execution described above makes it possible to handle memory errors during runtime, which is a RAS feature. It allows for the early flows of an STM to be pushed onto ROM, to address a scenario where the RAM STM, or any other portion of RAM, is corrupted.

[0032] **Figure 3** is a flow diagram of a method 300 for ROM-based execution of a peer monitor to support RAS according to an embodiment of the disclosure. Method 300 may be performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, programmable logic, microcode, etc.), software (such as instructions run on a processing device), firmware, or a combination thereof. In one embodiment, method 300 is performed by device 100 described with respect to **Figure 1**.

[0033] Method 300 begins at block 310, where STM code in a ROM region is invoked upon detecting an SMI event. In one embodiment, a processor detects the SMI event and invokes the ROM STM code based on an address for an entry point of the ROM STM stored in a register of the processor. Then, at block 320, the ROM STM code runs an error check on RAM memory. At decision block 330, it is determined whether there was an error at the RAM memory.

[0034] If there is a RAM error at decision block 330 and the memory in error does not fall within the region of the OEM SMM code, then method 300 proceeds to block 340 where BIOS SMM handler is invoked directly from the ROM STM code. In one embodiment, an entry point address for the BIOS SMM handler may be stored in a register

of the processor and used by the ROM STM to invoke the BIOS SMM handler. At block 350, execution control is passed from the ROM STM code to the BIOS SMM handler in order for the BIOS SMM handler to resolve the memory error in RAM.

[0035] If, at decision block 330, there is a RAM memory error and the memory overlaps the OEM SMRAM, then the STM can setup a machine check log with information of the memory location. The firmware may then return to the host environment by injecting a machine check abort and/or reset the machine. In either case, an OS agent can ascertain the error information from the machine check log.

[0036] If there are no RAM errors detected at decision block 330, then method 300 proceeds to block 360 where RAM STM code is invoked from the ROM STM code. In one embodiment, an STM transition data structure is used to transition from the ROM STM to the RAM STM. The ROM STM may use the transition structure to pass control and also to ascertain if the ROM STM overlaps the memory range where the memory error has been reported. Then, at block 370, execution control is passed from the ROM STM to the RAM STM in order for the RAM STM to take normal STM action to invoke an SMM guest, handle the exception from the SMI event, get Resume from System Management Mode (RSM), and return to original execution environment.

[0037] **Figure 4** is a block diagram illustrating hot plug support for RAS by a peer monitor in accordance with embodiments of the disclosure. Some of the embodiments discussed herein may be described using terminology associated with virtualization technology (VT) currently available in many microprocessors. VT is functionality allowing more than one virtual machine (VM) to simultaneously share access to physical processing

resources in a safe and efficient manner. However, use of these terms is for the sake of explanation herein, and implementations consistent with the disclosure are not limited to using this technology. For example, other hardware (e.g., microprocessors) and/or software offering similar features may also be employed in a manner consistent with the various embodiments as disclosed herein.

[0038] In the example implementation of **Figure 4**, high privilege execution environment memory 112, STM 114, BIOS guest 116, and low privilege execution environment memory 120 are the same as their identically-numbered counterparts from **Figure 1**. STM is configured to manage the operation of BIOS SMM guest 116. In one embodiment, LP manager 122 from **Figure 1** is illustrated as Measured Launch Environment (MLE) 122 of low privilege execution environment memory 120. STM 114 communicates with BIOS SMM 116 and MLE 122 using a variety of different VMCALL commands, and vice versa. A VMCALL command may include an instruction between a hypervisor and a guest, or between two different hypervisors.In some embodiments, a VMCall is a mechanism that a virtual machine guest, such as the MLE or the BIOS SMM, can use to communicate with to a hypervisor, such as the STM.

[0039] In one embodiment, a hot plug service module 420 is added to the STM 114 to define a set of extensions to Application Programming Interfaces (APIs) to support hot plug functionality for RAS-capable STM. The hot plug service module 420 may communicate with corresponding hot plug modules 410, 430 in the BIOS SMM 116 and the MLE 122.

[0040] In one embodiment, the hot plug service module 420 provides support for memory hot plug operations and processor hot plug operations. With respect to memory hot plug, two new BIOS-to-STM VMCALL commands are introduced, an add BIOS resource VMCALL and a remove BIOS resource VMCALL. Although the description herein specifically refers to an add BIOS resource VMCALL and a remove BIOS resource VMCALL, other identifications of such instructions may be used and embodiments of the disclosure are not limited to the specific names used herein. **Figures 5A and 5B** below describe example flows for adding and removing memory using hot plug support via STM according to embodiments of the disclosure.

[0041] With respect to processor hot plug, two new BIOS-to-STM VMCALL commands are introduced, an add processor VMCALL and a remove processor VMCALL. Although the description herein specifically refers to an add processor VMCALL and a remove processor VMCALL, other identifications of such instructions may be used and embodiments of the disclosure are not limited to the specific names used herein. **Figures 6A and 6B** below describe example flows for adding and removing processors using hot plug support via STM according to embodiments of the disclosure.

[0042] **Figure 5A** is a flow diagram of a method 500 for hot plug support by an STM when adding memory to a computer system according to an embodiment of the disclosure. Method 500 may be performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, programmable logic, microcode, etc.), software (such as instructions run on a processing device), firmware, or a combination thereof. In one embodiment, method 500 is performed by STM 114 described with respect to **Figure4**.

[0043] Method 500 begins at block 505, where an add BIOS resource VMCALL is received at an STM from BIOS. In one embodiment, the add BIOS resource VMCALL is received after the BIOS receives notification of new memory being added via an SMI event. At decision block 510, the STM determines whether there is any overlap between an MLE-protected region of memory and a region of memory that the new memory would be associated with. In one embodiment, the STM maintains a list of resources claimed by the BIOS and by the MLE, and can cross-check this list for the overlap determination at decision block 510. If there is an overlap detected at decision block 510, then the STM generates and exception and denies the addition of the new memory at block 545.

[0044] On the other hand, if no overlap is detected at decision block 510, the method 500 proceeds to block 515 where the STM sends a confirmation to BIOS to add the new memory. Then, at block 520 the STM receives a remove BIOS resource VMCALL from the BIOS. This may be in response to the BIOS temporarily accessing the new memory after the STM confirmed to BIOS that there was no overlap.

[0045] At block 525, in response to the remove BIOS resource VMCALL, the STM opens memory protection for the new memory to the MLE. In one embodiment, once the memory protection for the new memory is opened to the MLE, the MLE may then receive the new memory added via a System Control Interrupt (SCI) event. At block 530, the STM receives a protect resource VMCALL from the MLE. In response, the STM, at decision block 535, determines whether there is an overlap between a BIOS-declared resource region and a memory of region associated with the new memory.

[0046] If an overlap is detected, then method 500 proceeds to block 545 where an exception is generated by the STM and the addition of the new memory is denied. On the other hand, if no overlap is detected, then the new memory region is protected by the STM for the MLE at block 540. In one embodiment, the new memory resource is added to the list of MLE resource maintained by the STM. The MLE may then inject a virtual SCI to a guest adding the new memory, so that the guest may use the new memory.

[0047] **Figure 5B** is a flow diagram of a method 550 for hot plug support by an STM when removing memory to a computer system according to an embodiment of the disclosure. Method 550 may be performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, programmable logic, microcode, etc.), software (such as instructions run on a processing device), firmware, or a combination thereof. In one embodiment, method 550 is performed by STM 114 described with respect to **Figure4**.

[0048] Method 550 begins at block 560, where an STM receives an unprotect resource VMCALL from an MLE. In one embodiment, the unprotect resource VMCALL is sent by the MLE to the STM in response to the MLE removing memory associated with one of the MLE's guests. The guest may remove the memory, which generates a remove memory request to the BIOS via an SMI. In turn, the BIOS may trigger an SCI to notify the MLE, and the MLE may inject a virtual SCI to the guest so that the guest removes the memory. Once the memory is removed, the MLE may invoke the unprotect resource VMCALL to the STM at block 560.

[0049] Subsequently, at block 570, the STM removes MLE protection for the memory in a list (or lists) maintained by the STM. The BIOS may then remove the memory

and invoke a remove BIOS resource VMCALL, which is received by the STM at block 580. In response to the remove BIOS resource VMCALL, the STM removes the memory from a BIOS resource list maintained by the STM at block 590.

[0050] **Figure 6A** is a flow diagram of a method 600 for hot plug support by an STM when adding a processor to a computer system according to an embodiment of the disclosure. Method 600 may be performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, programmable logic, microcode, etc.), software (such as instructions run on a processing device), firmware, or a combination thereof. In one embodiment, method 600 is performed by STM 114 described with respect to **Figure4**.

[0051] Method 600 begins at block 610, where an add processor VMCALL is received from BIOS via an original processor already operating on the computing system. In one embodiment, the add processor VMCALL is received in response to the BIOS receiving an SMI event indicated a new processor is being added. At block 620, the STM adds the new processor to a list of BIOS resources maintained by the STM. In one embodiment, the list of BIOS resources includes a data structure, such as a linked list, detailing processors of the computing system.

[0052] At block 630, an initialize protection VMCALL is received by the STM from the MLE via the new processor. In one embodiment, the initialize protection VMCALL is received in response to the MLE receiving the new processor, which is added via an SCI event. At block 640, STM is enabled on the new processor by the STM. Once the STM is enabled for the new processor, the MLE may inject a virtual SCI to the guest adding the new processor so that the guest can use the processor.

[0053] **Figure 6B** is a flow diagram of a method 650 for hot plug support by an STM when removing a processor to a computer system according to an embodiment of the disclosure. Method 650 may be performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, programmable logic, microcode, etc.), software (such as instructions run on a processing device), firmware, or a combination thereof. In one embodiment, method 650 is performed by STM 114 described with respect to **Figure4**.

[0054] Method 650 begins at block 660, wherean STM receives a stop STM VMCALL from an MLE. In one embodiment, the stop STM VMCALL is sent by the MLE to the STM in response to the MLE removing a processor associated with one of the MLE's guests. The guest may remove the processor, which generates a remove processor request to the BIOS via an SMI. In turn, the BIOS may trigger an SCI to notify the MLE, and the MLE may inject a virtual SCI to the guest so that the guest removes the processor. Once the processor is removed, the MLE may invoke the stop STM VMCALL to the STM at block 660.

[0055] Subsequently, at block 670, the STM stops STM services for the processor being removed in response to the stop STM VMCALL. The BIOS may then remove the processor and invoke a remove processor VMCALL, which is received by the STM at block 680. In response to the remove processor VMCALL, the STM removes the processor by destructing the data structure associated with the processor that is maintained by the STM at block 690.

[0056] **Figure 7** illustrates a diagrammatic representation of a machine in the example form of a computer system 700 within which a set of instructions, for causing the

machine to perform any one or more of the methodologies discussed herein, may be executed. In alternative embodiments, the machine may be connected (e.g., networked) to other machines in a LAN, an intranet, an extranet, or the Internet. The machine may operate in the capacity of a server or a client device in a client-server network environment, or as a peer machine in a peer-to-peer (or distributed) network environment. The machine may be a personal computer (PC), a tablet PC, a set-top box (STB), a Personal Digital Assistant (PDA), a cellular telephone, a web appliance, a server, a network router, switch or bridge, or any machine capable of executing a set of instructions (sequential or otherwise) that specify actions to be taken by that machine. Further, while only a single machine is illustrated, the term "machine" shall also be taken to include any collection of machines that individually or jointly execute a set (or multiple sets) of instructions to perform any one or more of the methodologies discussed herein.

[0057] The computer system 700 includes a processing device 702, a main memory 704 (e.g., read-only memory (ROM), flash memory, dynamic random access memory (DRAM) (such as synchronous DRAM (SDRAM) or DRAM (RDRAM), etc.), a static memory 706 (e.g., flash memory, static random access memory (SRAM), etc.), and a data storage device 718, which communicate with each other via a bus 730.

[0058] Processing device 702 represents one or more general-purpose processing devices such as a microprocessor, central processing unit, or the like. More particularly, the processing device may be complex instruction set computing (CISC) microprocessor, reduced instruction set computer (RISC) microprocessor, very long instruction word (VLIW) microprocessor, or processor implementing other instruction sets, or processors

implementing a combination of instruction sets. Processing device 702 may also be one or more special-purpose processing devices such as an application specific integrated circuit (ASIC), a field programmable gate array (FPGA), a digital signal processor (DSP), network processor, or the like. In one embodiment, processing device 702 may include one or processing cores. The processing device 702 is configured to execute the processing logic 726 for performing the operations and steps discussed herein. In one embodiment, processing device 702 is the same as processing device 100 described with respect to **Figure 1** that implements a RAS-capable HP manager, such as an STM. For example, processing device 702 may include a RAS-capable HP manager, such as STM 114 of **Figure 1**.

[0059] The computer system 700 may further include a network interface device 708 communicably coupled to a network 720. The computer system 700 also may include a video display unit 710 (e.g., a liquid crystal display (LCD) or a cathode ray tube (CRT)), an alphanumeric input device 712 (e.g., a keyboard), a cursor control device 714 (e.g., a mouse), and a signal generation device 716 (e.g., a speaker). Furthermore, computer system 700 may include a graphics processing unit 722, a video processing unit 728, and an audio processing unit 732.

[0060] The data storage device 718 may include a machine-accessible storage medium 724 on which is stored software 726 implementing any one or more of the methodologies of functions described herein, such as implementing an RS with restricted entries as described above. The software 726 may also reside, completely or at least partially, within the main memory 704 as instructions 726 and/or within the processing device 702 as processing logic 726 during execution thereof by the computer system 700;

the main memory 704 and the processing device 702 also constituting machine-accessible storage media.

[0061] The machine-readable storage medium 724 may also be used to store instructions 726 implementing a RAS-capable HP manager, such as described with respect to device 100 in**Figure 1**, and/or a software library containing methods that call the above applications. While the machine-accessible storage medium 728 is shown in an example embodiment to be a single medium, the term "machine-accessible storage medium" should be taken to include a single medium or multiple media (e.g., a centralized or distributed database, and/or associated caches and servers) that store the one or more sets of instructions. The term "machine-accessible storage medium" shall also be taken to include any medium that is capable of storing, encoding or carrying a set of instruction for execution by the machine and that cause the machine to perform any one or more of the methodologies of the disclosure. The term "machine-accessible storage medium" shall accordingly be taken to include, but not be limited to, solid-state memories, and optical and magnetic media.

[0062] The following examples pertain to further embodiments. Example 1 is a method for supporting reliability, availability, and serviceability (RAS) flows in a peer monitor comprising receiving, by a processing device, a system management interrupt (SMI) event, and invoking, by the processing device in response to the SMI event, a privilege manager to execute from a read-only memory (ROM) entry point to handle the SMI event, the privilege manager comprising a hot plug service module to provide support for memory hot plug functionality and processor hot plug functionality. In Example 2, the subject matter of Example 1 can optionally include the privilege manager comprising a system

management interrupt (SMI) transfer monitor (STM). In Example 3, the subject matter of any one of Examples 1-2 can optionally include the privilege manager comprising a hypervisor executing system management mode (SMM) code of a basic input/output system (BIOS) as a guest of the privilege manager.

[0063] In Example 4, the subject matter of any one of Examples 1-3 can optionally include the privilege manager to execute an error check on read-access memory (RAM) subsequent to invocation. In Example 5, the subject matter of any one of Examples 1-4 can optionally include the privilege manager is to access a transition data structure to verify an integrity of the RAM for the error check. In Example 6, the subject matter of any one of Examples 1-5 can optionally include the transition data structure to maintain data comprising at least one of a size of the privilege manager in the ROM, a size of the privilege manager in the RAM, a cyclic redundancy check (CRC) value, a check value, a hash, or a cryptographic marker.

[0064] In Example 7, the subject matter of any one of Examples 1-6 can optionally include wherein when the error check indicates an error in the RAM, the privilege manager is to invoke a BIOS SMM handler to resolve the error in the RAM. In Example 8, the subject matter of any one of Example 1-6 can optionally include wherein when the error check indicates no error in the RAM, the privilege manager is to pass control to another version of the privilege manager executing in the RAM.

[0065] In Example 9, the subject matter of any one of Examples 1-8 can optionally include wherein the hot plug service module comprises a set of extensions to Application Programming Interfaces (APIs) to support hot plug functionality for memory, the extensions

to the APIs comprising an add BIOS resource VMCALL and a remove BIOS resource VMCALL that are communicated between a basicinput/output system (BIOS) and the privilege manager. In Example 10, the subject matter of any one of Examples 1-9 can optionally include wherein the hot plug service module comprises a set of extensions to Application Programming Interfaces (APIs) to support hot plug functionality for processor, the extensions to the APIs comprising an add processor VMCALL and a remove processor VMCALLthat are communicated between a basicinput/output system (BIOS) and the privilege manager.

[0066] Example 11 is an apparatus for supporting reliability, availability, and serviceability (RAS) flows in a peer monitor comprising a memory module comprising a privilege execution environment and a low privilege execution environment, and a processing device communicably coupled to the memory module. In Example 11, the processing device is toreceive a system management interrupt (SMI) event, invoke, in response to the SMI event, a SMI transfer monitor (STM) to execute from a read-only memory (ROM) entry point to handle the SMI event, and provide, by the STM, support for memory hot plug functionality and processor hot plug functionality via a hot plug service module of the STM. In Example 12, the subject matter of Example 11 can optionally include wherein the STM to execute as a hypervisor and to virtualize system management mode (SMM) code of a basic input/output system (BIOS) of the apparatus as a guest of the STM.

[0067] In Example 13, the subject matter of any one of Examples 11-12 can optionally include wherein the STM to execute an error check on read-access memory

(RAM) subsequent to invocation. In Example 14, the subject matter of any one of Examples 11-13 can optionally include wherein when the error check indicates an error in the RAM, the STM is to invoke a BIOS SMM handler to resolve the error in the RAM. In Example 15, the subject matter of any one of Examples 11-14 can optionally include wherein when the error check indicates no error in the RAM, the STM is to pass control to another version of the privilege manager executing in the RAM.

[0068] In Example 16, the subject matter of any one of Examples 11-15 can optionally include wherein the hot plug service module comprises a set of extensions to Application Programming Interfaces (APIs) to support the hot plug functionality for the memory, the extensions to the APIs comprising an add BIOS resource VMCALL and a remove BIOS resource VMCALL that are communicated between a basicinput/output system (BIOS) and the STM.

[0069] In Example 17, the subject matter of any one of Examples 11-16 can optionally include wherein the hot plug service module comprises a set of extensions to Application Programming Interfaces (APIs) to support the hot plug functionality for the processor, the extensions to the APIs comprising an add processor VMCALL and a remove processor VMCALLthat are communicated between a basicinput/output system (BIOS) and the STM. All optional features of the apparatus described above may also be implemented with respect to the method or process described herein.

[0070] Example 18 is a non-transitory machine-readable storage medium for supporting reliability, availability, and serviceability (RAS) flows in a peer monitor. In Example 18, the non-transitory machine-readable medium includes data that, when accessed

by a processing device, cause the processing device to perform operations comprising accessing, by a privilege manager executed by the processing device from a read-only memory (ROM) entry point, a transition data structure tracking an integrity of a read-access memory (RAM), the accessing in response to the processing device receiving a system management interrupt (SMI) event. In addition, in Example 18, the operations further comprise executing, by the privilege manager based on the transition data structure, an error check on the RAM, and when the error check indicates no error in the RAM, passing, by the privilege manager, control to another version of the privilege manager executing in the RAM to handle the SMI event, the privilege manager comprising a hot plug service module to provide support for memory hot plug functionality and processor hot plug functionality.

[0071] In Example 19, the subject matter of claim 18 can optionally include wherein the privilege manager is a system management interrupt (SMI) transfer monitor (STM) that executes as a hypervisor, and wherein the STM virtualizes system management mode (SMM) code of a basic input/output system (BIOS) as a guest of the STM. In Example 20, the subject matter of any one of Examples 18-19 can optionally include when the error check indicates an error in the RAM, invoking, by the privilege manager, a BIOS SMM handler to resolve the error in the RAM.

[0072] In Example 21, the subject matter of any one of Examples 18-20 can optionally include wherein the hot plug service module comprising a set of extensions to Application Programming Interfaces (APIs) to support hot plug functionality for memory, the extensions to the APIs comprising an add BIOS resource VMCALL and a remove BIOS resource VMCALL that are communicated between a basicinput/output system (BIOS) and

the privilege manager. In Example 22, the subject matter of any one of Examples 18-21 can optionally include wherein the hot plug service module comprising a set of extensions to Application Programming Interfaces (APIs) to support hot plug functionality for processor, the extensions to the APIs comprising an add processor VMCALL and a remove processor VMCALLthat are communicated between a basicinput/output system (BIOS) and the privilege manager.

[0073] Example 23 is an apparatus for supporting reliability, availability, and serviceability (RAS) flows in a peer monitor comprising means for accessing, via a read-only memory (ROM) entry point, a transition data structure tracking an integrity of a read-access memory (RAM), the accessing in response to receiving a system management interrupt (SMI) event. The apparatus of Example 23 further comprises means for executing, based on the transition data structure, an error check on the RAM, and when the error check indicates no error in the RAM, means for passing control to version of a privilege manager executing in the RAM to handle the SMI event, the means for accessing comprising a hot plug service module to provide support for memory hot plug functionality and processor hot plug functionality. In Example 24, the subject matter of Example 22 can optionally include the apparatus further configured to perform the method of any one of the claims 2 to 10.

[0074] Example 25 is at least one machine readable medium comprising a plurality of instructions that in response to being executed on a computing device, cause the computing device to carry out a method according to any one of Examples 1-10. Example 26 is an apparatus for supporting reliability, availability, and serviceability (RAS) flows in a peer monitor, configured to perform the method of any one of Examples 1-10. Example 27

is an apparatus comprising means for performing the method of any one of Examples 1-10. Specifics in the Examines may be used anywhere in one or more embodiments.

[0075] While the disclosure has been described with respect to a limited number of embodiments, those skilled in the art will appreciate numerous modifications and variations there from. It is intended that the appended claims cover all such modifications and variations as fall within the true spirit and scope of this disclosure.

[0076] A design may go through various stages, from creation to simulation to fabrication. Data representing a design may represent the design in a number of manners. First, as is useful in simulations, the hardware may be represented using a hardware description language or another functional description language. Additionally, a circuit level model with logic and/or transistor gates may be produced at some stages of the design process. Furthermore, most designs, at some stage, reach a level of data representing the physical placement of various devices in the hardware model. In the case where conventional semiconductor fabrication techniques are used, the data representing the hardware model may be the data specifying the presence or absence of various features on different mask layers for masks used to produce the integrated circuit. In any representation of the design, the data may be stored in any form of a machine readable medium. A memory or a magnetic or optical storage such as a disc may be the machine readable medium to store information transmitted via optical or electrical wave modulated or otherwise generated to transmit such information. When an electrical carrier wave indicating or carrying the code or design is transmitted, to the extent that copying, buffering, or re-transmission of the electrical signal is performed, a new copy is made. Thus, a

communication provider or a network provider may store on a tangible, machine-readable medium, at least temporarily, an article, such as information encoded into a carrier wave, embodying techniques of embodiments of the disclosure.

[0077] A module as used herein refers to any combination of hardware, software, and/or firmware. As an example, a module includes hardware, such as a micro-controller, associated with a non-transitory medium to store code adapted to be executed by the micro-controller. Therefore, reference to a module, in one embodiment, refers to the hardware, which is specifically configured to recognize and/or execute the code to be held on a non-transitory medium. Furthermore, in another embodiment, use of a module refers to the non-transitory medium including the code, which is specifically adapted to be executed by the microcontroller to perform predetermined operations. And as can be inferred, in yet another embodiment, the term module (in this example) may refer to the combination of the microcontroller and the non-transitory medium. Often module boundaries that are illustrated as separate commonly vary and potentially overlap. For example, a first and a second module may share hardware, software, firmware, or a combination thereof, while potentially retaining some independent hardware, software, or firmware. In one embodiment, use of the term logic includes hardware, such as transistors, registers, or other hardware, such as programmable logic devices.

[0078] Use of the phrase 'configured to,' in one embodiment, refers to arranging, putting together, manufacturing, offering to sell, importing and/or designing an apparatus, hardware, logic, or element to perform a designated or determined task. In this example, an apparatus or element thereof that is not operating is still 'configured to' perform a

designated task if it is designed, coupled, and/or interconnected to perform said designated task. As a purely illustrative example, a logic gate may provide a 0 or a 1 during operation. But a logic gate 'configured to' provide an enable signal to a clock does not include every potential logic gate that may provide a 1 or 0. Instead, the logic gate is one coupled in some manner that during operation the 1 or 0 output is to enable the clock. Note once again that use of the term 'configured to' does not require operation, but instead focus on the latent state of an apparatus, hardware, and/or element, where in the latent state the apparatus, hardware, and/or element is designed to perform a particular task when the apparatus, hardware, and/or element is operating.

[0079] Furthermore, use of the phrases 'to,' 'capable of/to,' and or 'operable to,' in one embodiment, refers to some apparatus, logic, hardware, and/or element designed in such a way to enable use of the apparatus, logic, hardware, and/or element in a specified manner. Note as above that use of to, capable to, or operable to, in one embodiment, refers to the latent state of an apparatus, logic, hardware, and/or element, where the apparatus, logic, hardware, and/or element is not operating but is designed in such a manner to enable use of an apparatus in a specified manner.

[0080] A value, as used herein, includes any known representation of a number, a state, a logical state, or a binary logical state. Often, the use of logic levels, logic values, or logical values is also referred to as 1's and 0's, which simply represents binary logic states. For example, a 1 refers to a high logic level and 0 refers to a low logic level. In one embodiment, a storage cell, such as a transistor or flash cell, may be capable of holding a single logical value or multiple logical values. However, other representations of values in

computer systems have been used. For example the decimal number ten may also be represented as a binary value of 1010 and a hexadecimal letter A. Therefore, a value includes any representation of information capable of being held in a computer system.

[0081] Moreover, states may be represented by values or portions of values. As an example, a first value, such as a logical one, may represent a default or initial state, while a second value, such as a logical zero, may represent a non-default state. In addition, the terms reset and set, in one embodiment, refer to a default and an updated value or state, respectively. For example, a default value potentially includes a high logical value, i.e. reset, while an updated value potentially includes a low logical value, i.e. set. Note that any combination of values may be utilized to represent any number of states.

[0082] The embodiments of methods, hardware, software, firmware or code set forth above may be implemented via instructions or code stored on a machine-accessible, machine readable, computer accessible, or computer readable medium which are executable by a processing element. A non-transitory machine-accessible/readable medium includes any mechanism that provides (i.e., stores and/or transmits) information in a form readable by a machine, such as a computer or electronic system. For example, a non-transitory machine-accessible medium includes random-access memory (RAM), such as static RAM (SRAM) or dynamic RAM (DRAM); ROM; magnetic or optical storage medium; flash memory devices; electrical storage devices; optical storage devices; acoustical storage devices; other form of storage devices for holding information received from transitory (propagated) signals (e.g., carrier waves, infrared signals, digital signals); etc, which are to be distinguished from the non-transitory mediums that may receive information there from.

[0083] Instructions used to program logic to perform embodiments of the disclosure may be stored within a memory in the system, such as DRAM, cache, flash memory, or other storage. Furthermore, the instructions can be distributed via a network or by way of other computer readable media. Thus a machine-readable medium may include any mechanism for storing or transmitting information in a form readable by a machine (e.g., a computer), but is not limited to, floppy diskettes, optical disks, Compact Disc, Read-Only Memory (CD-ROMs), and magneto-optical disks, Read-Only Memory (ROMs), Random Access Memory (RAM), Erasable Programmable Read-Only Memory (EPROM), Electrically Erasable Programmable Read-Only Memory (EEPROM), magnetic or optical cards, flash memory, or a tangible, machine-readable storage used in the transmission of information over the Internet via electrical, optical, acoustical or other forms of propagated signals (e.g., carrier waves, infrared signals, digital signals, etc.). Accordingly, the computer-readable medium includes any type of tangible machine-readable medium suitable for storing or transmitting electronic instructions or information in a form readable by a machine (e.g., a computer)

[0084] Reference throughout this specification to “one embodiment” or “an embodiment” means that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least one embodiment of the disclosure. Thus, the appearances of the phrases “in one embodiment” or “in an embodiment” in various places throughout this specification are not necessarily all referring to the same embodiment. Furthermore, the particular features, structures, or characteristics may be combined in any suitable manner in one or more embodiments.

[0085] In the foregoing specification, a detailed description has been given with reference to specific exemplary embodiments. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the disclosure as set forth in the appended claims. The specification and drawings are, accordingly, to be regarded in an illustrative sense rather than a restrictive sense. Furthermore, the foregoing use of embodiment and other exemplarily language does not necessarily refer to the same embodiment or the same example, but may refer to different and distinct embodiments, as well as potentially the same embodiment.

CLAIMS

What is claimed is:

1. A methodfor supporting reliability, availability, and serviceability (RAS) flows in a peer monitor, comprising:

receiving, by a processing device, a system management interrupt (SMI) event; and

invoking, by the processing device in response to the SMI event, a privilege manager to execute from a read-only memory (ROM) entry point to handle the SMI event, the privilege manager comprising a hot plug service module to provide support for memory hot plug functionality and processor hot plug functionality.

2. The method of claim 1, wherein the privilege manager is a system management interrupt (SMI) transfer monitor (STM).

3. The method of claim 1, wherein the privilege manager comprises a hypervisor executing system management mode (SMM) code of a basic input/output system (BIOS) as a guest of the privilege manager.

4. The method of claim 1, wherein the privilege manager is to execute an error check on read-access memory (RAM) subsequent to invocation.

5. The method of claim 4, wherein the privilege manager is to access a transition data structure to verify an integrity of the RAM for the error check.

6. The method of claim 5, wherein the transition data structure maintains data comprising at least one of a size of the privilege manager in the ROM, a size of the privilege manager in the RAM, a cyclic redundancy check (CRC) value, a check value, a hash, or a cryptographic marker.

7. The method of claim 4, wherein when the error check indicates an error in the RAM, the privilege manager is to invoke a BIOS SMM handler to resolve the error in the RAM.

8. The method of claim 4, wherein when the error check indicates no error in the RAM, the privilege manager is to pass control to another version of the privilege manager executing in the RAM.

9. The method of claim 1, wherein the hot plug service module comprises a set of extensions to Application Programming Interfaces (APIs) to support hot plug functionality for memory, the extensions to the APIs comprising an add BIOS resource VMCALL and a remove BIOS resource VMCALL that are communicated between a basicinput/output system (BIOS) and the privilege manager.

10. The method of claim 1, wherein the hot plug service module comprises a set of extensions to Application Programming Interfaces (APIs) to support hot plug functionality for processor, the extensions to the APIs comprising an add processor VMCALL and a remove processor VMCALLthat are communicated between a basicinput/output system (BIOS) and the privilege manager.

11. An apparatusfor supporting reliability, availability, and serviceability (RAS) flows in a peer monitor, comprising:

a memory module comprising a privilege execution environment and a low privilege execution environment; and

a processing device communicably coupled to the memory module, the processing device to:

receive a system management interrupt (SMI) event;

invoke, in response to the SMI event, a SMI transfer monitor (STM) to execute from a read-only memory (ROM) entry point to handle the SMI event; and

provide, by the STM, support for memory hot plug functionality and processor hot plug functionality via a hot plug service module of the STM.

12. The apparatus of claim 11,wherein the STM to execute as a hypervisor and to virtualize system management mode (SMM) code of a basic input/output system (BIOS) of the apparatus as a guest of the STM.

13. The apparatus of claim 11, wherein the STM to execute an error check on read-access memory (RAM) subsequent to invocation.

14. The apparatusof claim 13, wherein when the error check indicates an error in the RAM, the STM is to invoke a BIOS SMM handler to resolve the error in the RAM.

15. The apparatusof claim 13, wherein when the error check indicates no error in the RAM, the STM is to pass control to another version of the privilege manager executing in the RAM.

16. The apparatus of claim 11, wherein the hot plug service module comprises a set of extensions to Application Programming Interfaces (APIs) to support the hot plug functionality for the memory, the extensions to the APIs comprising an add BIOS resource VMCALL and a remove BIOS resource VMCALL that are communicated between a basicinput/output system (BIOS) and the STM.

17. The apparatusof claim 11, wherein the hot plug service module comprises a set of extensions to Application Programming Interfaces (APIs) to support the hot plug functionality for the processor, the extensions to the APIs comprising an add processor VMCALL and a remove processor VMCALLthat are communicated between a basicinput/output system (BIOS) and the STM.

18. A non-transitory machine-readable storage medium including data that, when accessed by a processing device, cause the processing device to perform operations comprising:

accessing, by a privilege manager executed by the processing device from a read-only memory (ROM) entry point, a transition data structure tracking an integrity of a read-access memory (RAM), the accessing in response to the processing device receiving a system management interrupt (SMI) event;

executing, by the privilege manager based on the transition data structure, an error check on the RAM; and

when the error check indicates no error in the RAM, passing, by the privilege manager, control to another version of the privilege manager executing in the RAM to

handle the SMI event, the privilege manager comprising a hot plug service module to provide support for memory hot plug functionality and processor hot plug functionality.

19. The non-transitory machine -readable storage medium of claim 18, wherein the privilege manager is a system management interrupt (SMI) transfer monitor (STM) that executes as a hypervisor, and wherein the STM virtualizes system management mode (SMM) code of a basic input/output system (BIOS) as a guest of the STM.

20. The non-transitory machine -readable storage medium of claim 18, the operations further comprise, when the error check indicates an error in the RAM, invoking, by the privilege manager, a BIOS SMM handler to resolve the error in the RAM.

21. The non-transitory machine -readable storage medium of claim 18, wherein the hot plug service module comprising a set of extensions to Application Programming Interfaces (APIs) to support hot plug functionality for memory, the extensions to the APIs comprising an add BIOS resource VMCALL and a remove BIOS resource VMCALL that are communicated between a basicinput/output system (BIOS) and the privilege manager.

22. The non-transitory machine -readable storage medium of claim 18, wherein the hot plug service module comprising a set of extensions to Application Programming Interfaces (APIs) to support hot plug functionality for processor, the extensions to the APIs comprising an add processor VMCALL and a remove processor VMCALLthat are communicated between a basicinput/output system (BIOS) and the privilege manager.

23. An apparatus comprising:

means for accessing, via a read-only memory (ROM) entry point, a transition data structure tracking an integrity of a read-access memory (RAM), the accessing in response to receiving a system management interrupt (SMI) event;

means for executing, based on the transition data structure, an error check on the RAM; and

when the error check indicates no error in the RAM, means for passing control to version of a privilege manager executing in the RAM to handle the SMI event, the means for accessing comprising a hot plug service module to provide support for memory hot plug functionality and processor hot plug functionality.

24. The apparatus of claim 23, further configured to perform the method of any one of the claims 2 to 10.

25. At least one machine readable medium comprising a plurality of instructions that in response to being executed on a computing device, cause the computing device to carry out a method according to any one of claims 1 to 10.

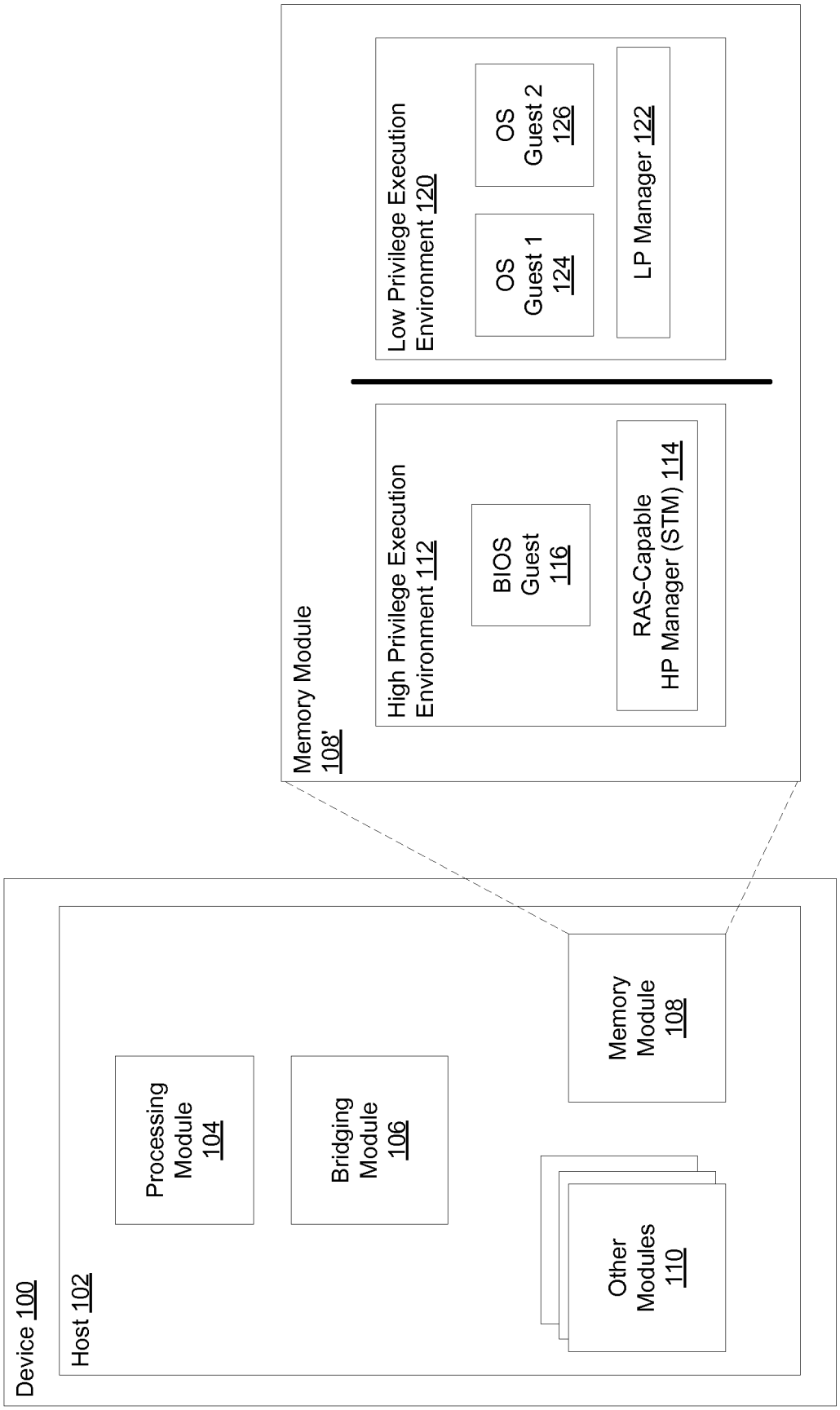
Device 100
Host 102
Processing Module 104
Bridging Module 106
Memory Module 108
Other Modules 110
Memory Module 108'
High Privilege Execution Environment 112
BIOS Guest 116
RAS-Capable HP Manager (STM) 114
Low Privilege Execution Environment 120
OS Guest 1 124
OS Guest 2 126
LP Manager 122

Figure 1

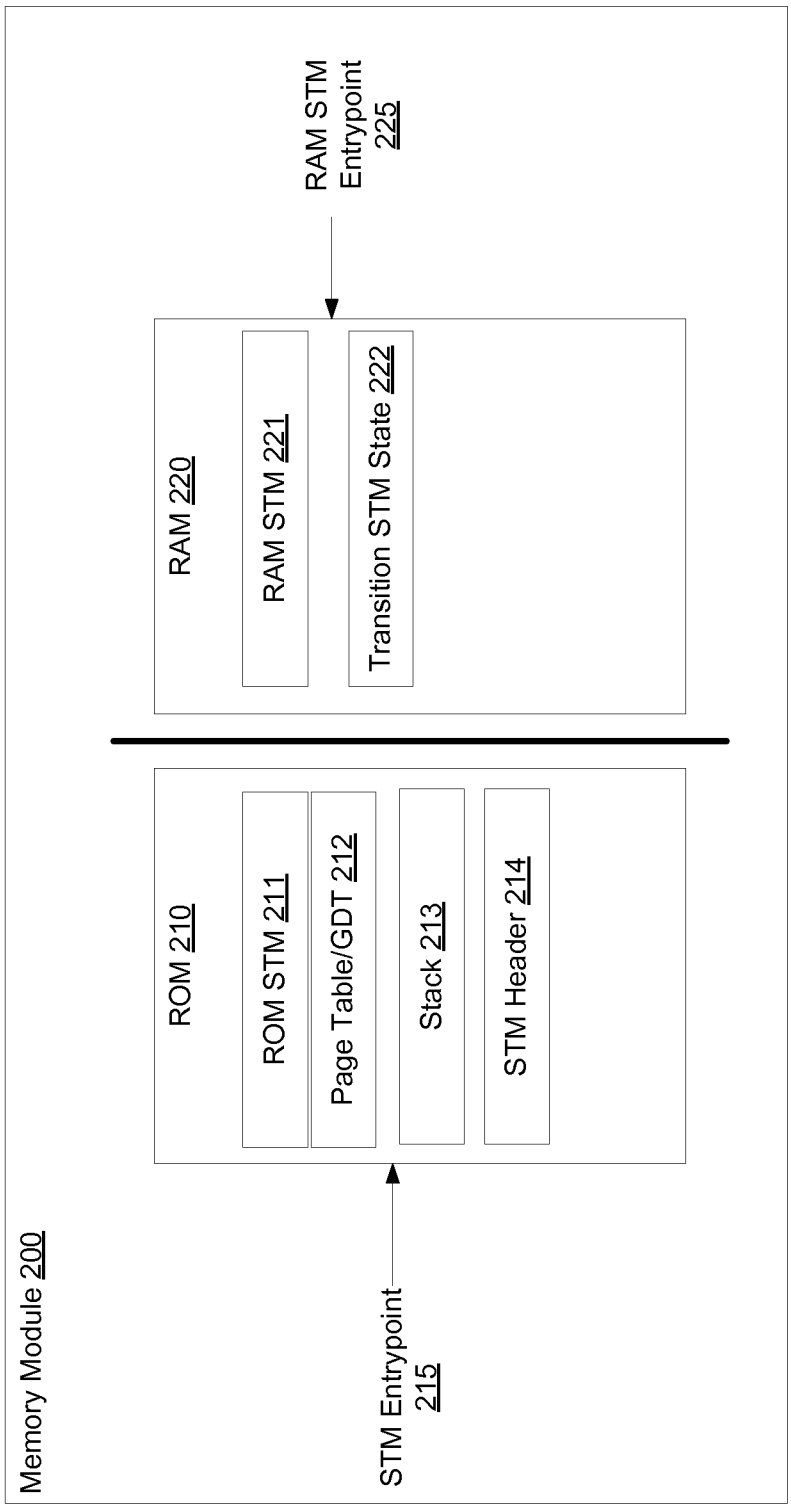
Memory Module 200
ROM 210
ROM STM 211
Page Table/GDT 212
Stack 213
STM Header 214
STM Entrypoint 215
RAM 220
RAM STM 221
Transition STM State 222
RAM STM Entrypoint 225

Figure 2

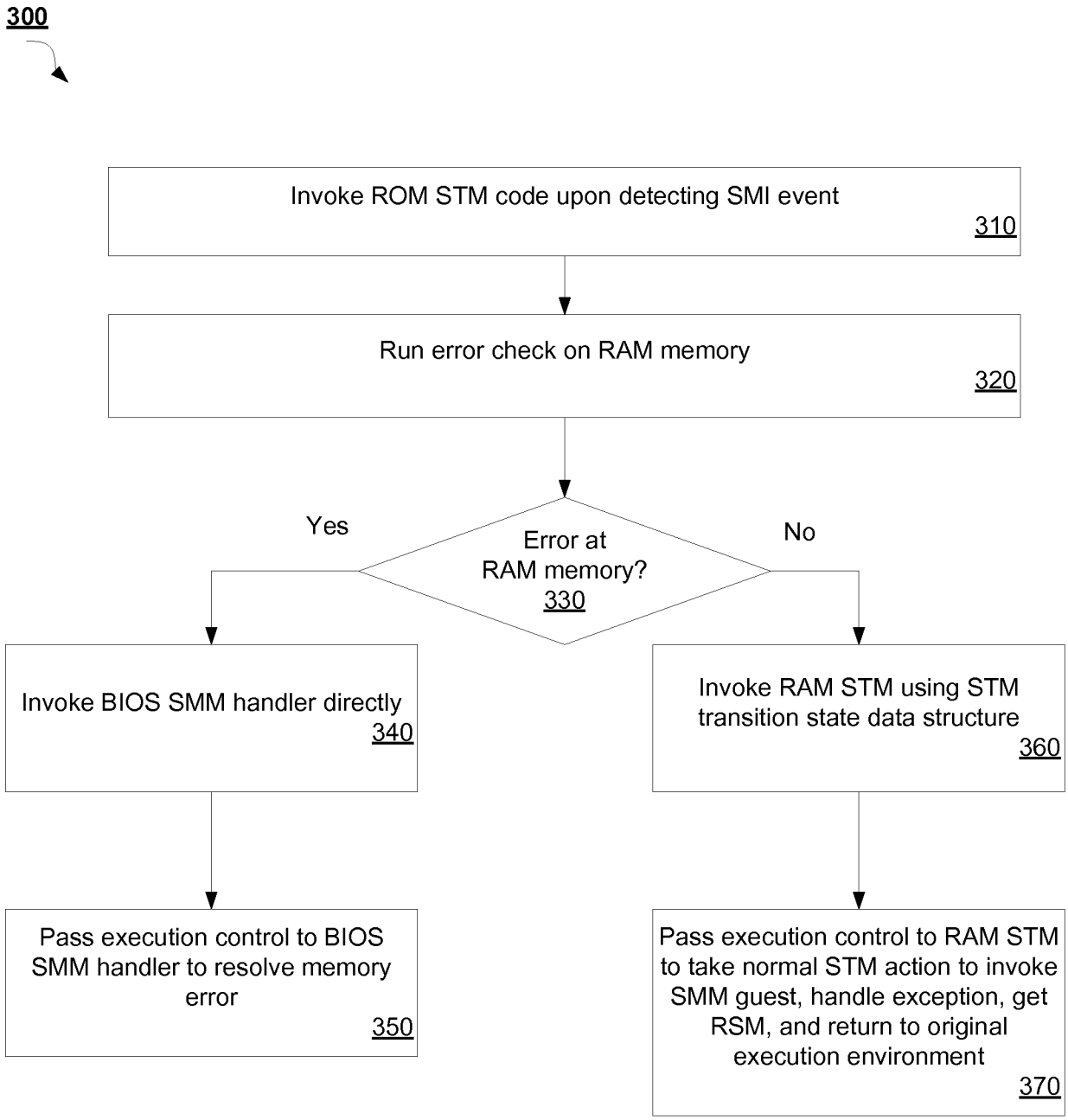
300
Invoke ROM STM code upon detecting SMI event
310
Run error check on RAM memory
320
Error at RAM memory?
330
Yes
No
Invoke BIOS SMM handler directly
340
Invoke RAM STM using STM transition state data structure
360
Pass execution control to BIOS SMM handler to resolve memory error
350
Pass execution control to RAM STM to take normal STM action to invoke SMM guest, handle exception, get RSM, and return to original execution environment
370

Figure 3

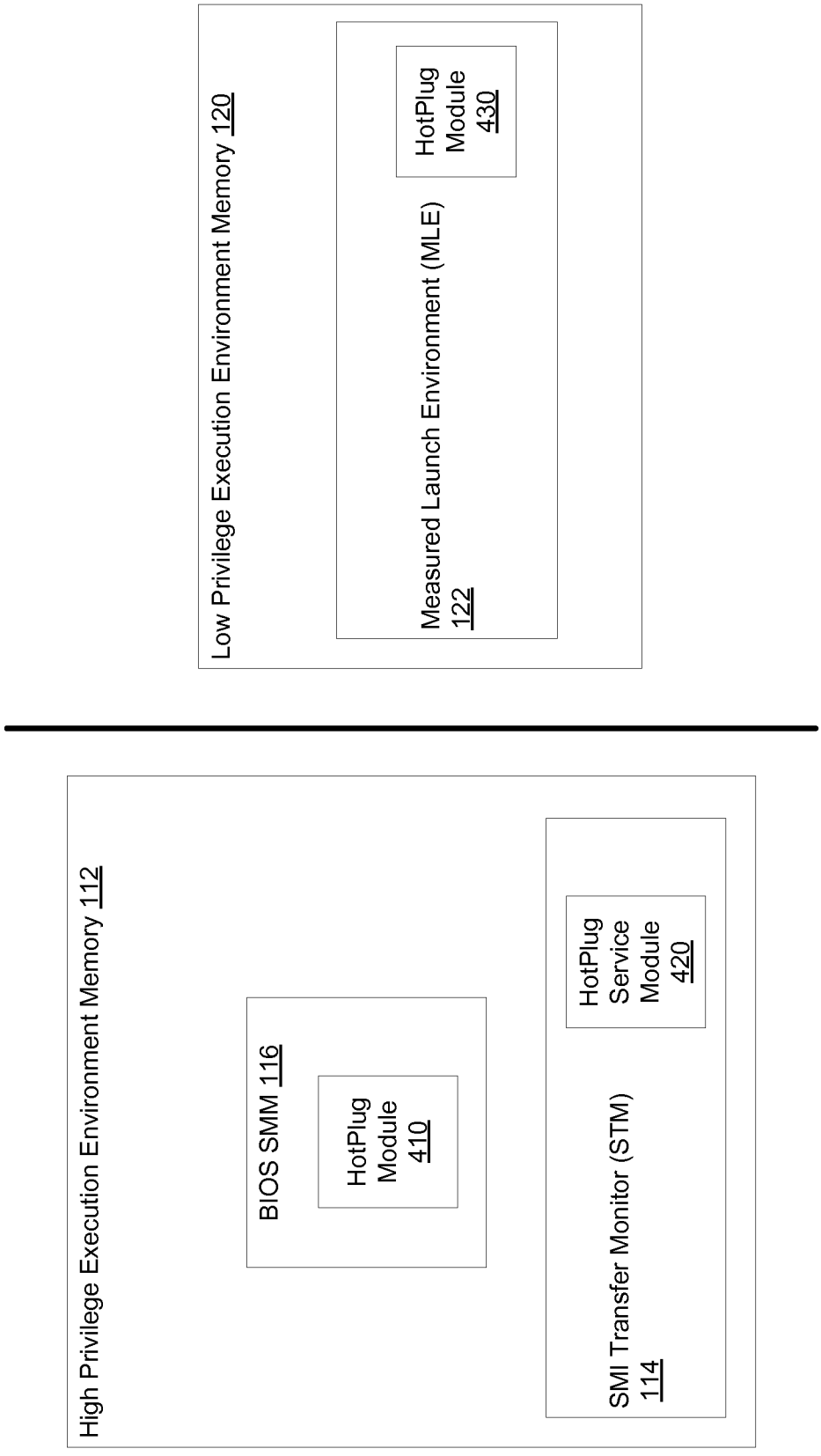
Low Privilege Execution Environment Memory 120
Measured Launch Environment (MLE) 122
HotPlug Module 430
High Privilege Execution Environment Memory 112
BIOS SMM 116
HotPlug Module 410
SMI Transfer Monitor (STM) 114
HotPlug Service Module 420

Figure 4

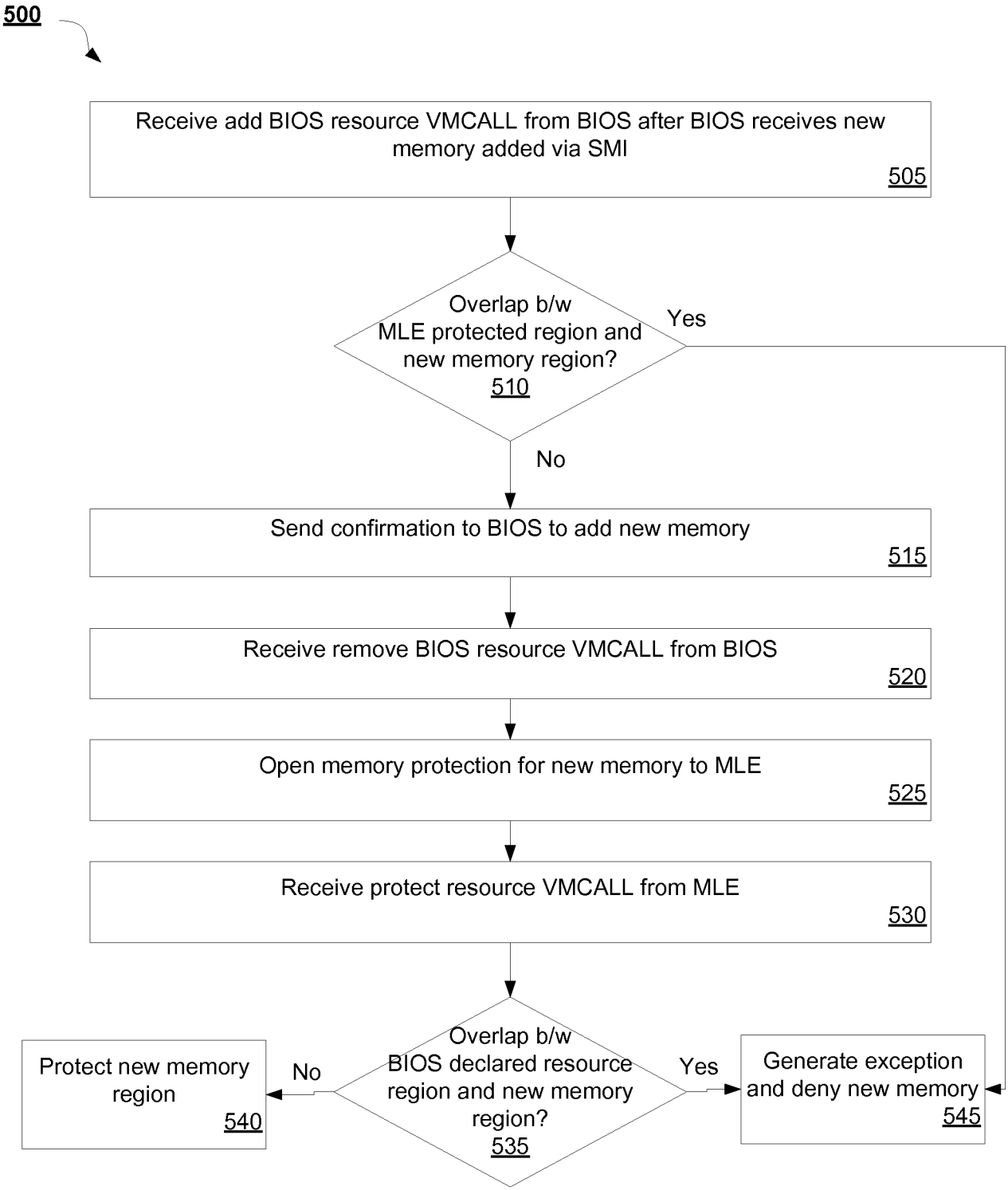
500
Receive add BIOS resource VMCALL from BIOS after BIOS receives new memory added via SMI
505
Overlap b/w MLE protected region and new memory region?
510
Yes
No
Send confirmation to BIOS to add new memory
515
Receive remove BIOS resource VMCALL from BIOS
520
Open memory protection for new memory to MLE
525
Receive protect resource VMCALL from MLE
530
Overlap b/w BIOS declared resource region and new memory region?
535
No
Yes
Protect new memory region
540
Generate exception and deny new memory
545

Figure 5A

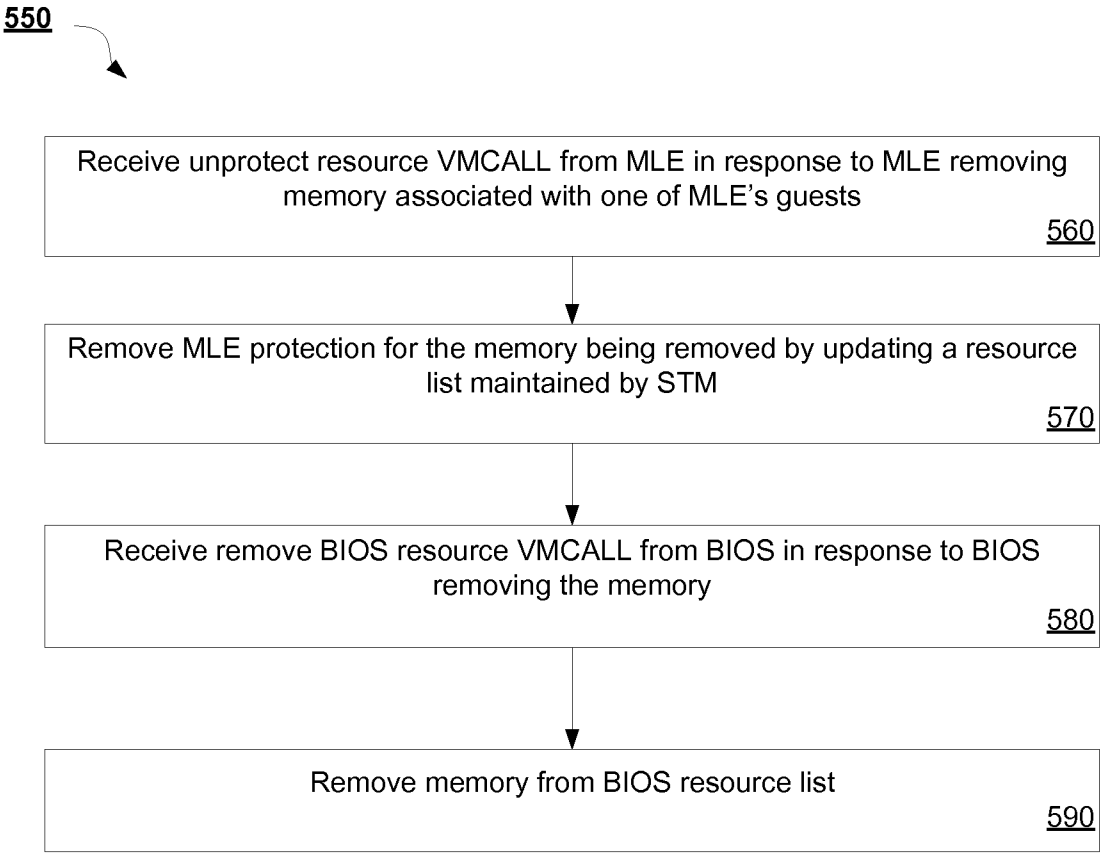
550
Receive unprotect resource VMCALL from MLE in response to MLE removing memory associated with one of MLE's guests
560
Remove MLE protection for the memory being removed by updating a resource list maintained by STM
570
Receive remove BIOS resource VMCALL from BIOS in response to BIOS removing the memory
580
Remove memory from BIOS resource list
590

Figure 5B

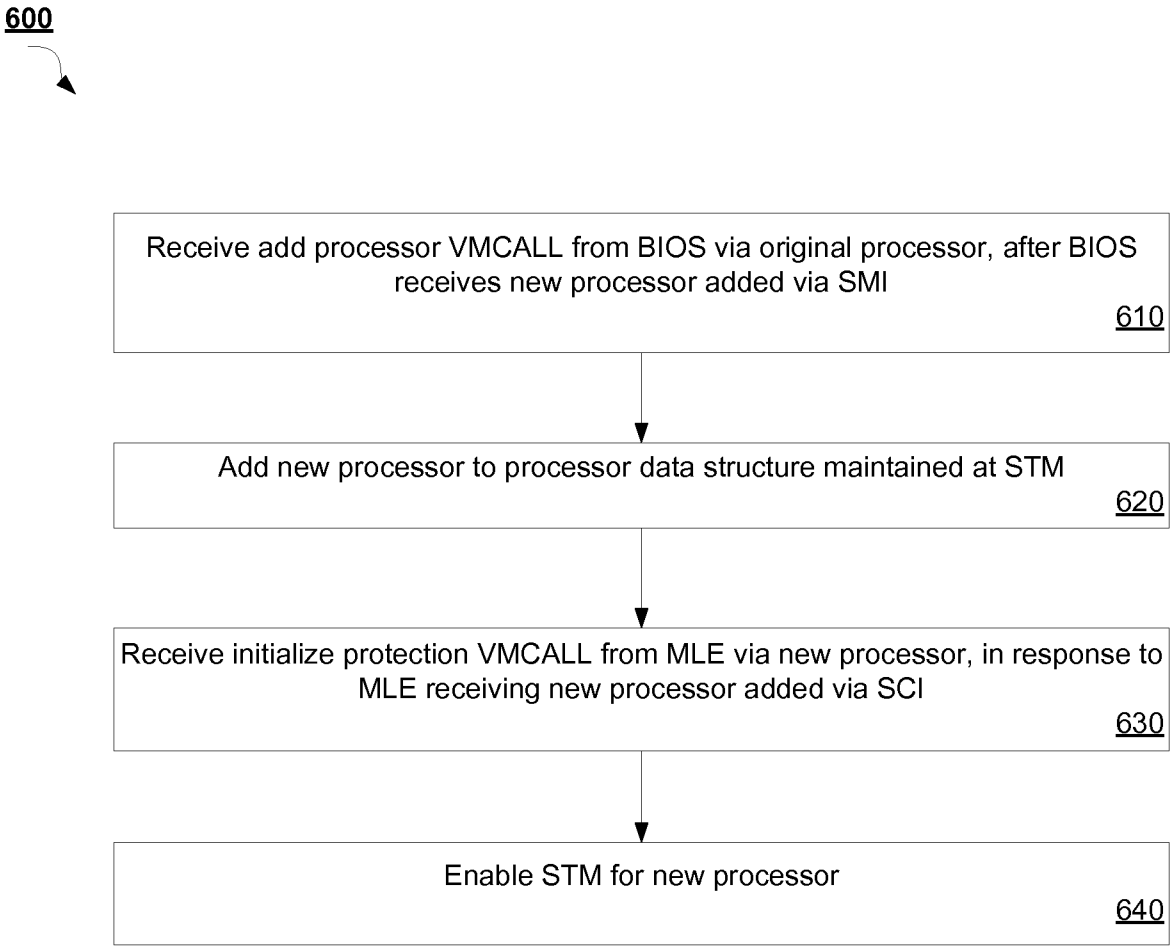
600
Receive add processor VMCALL from BIOS via original processor, after BIOS receives new processor added via SMI
610
Add new processor to processor data structure maintained at STM
620
Receive initialize protection VMCALL from MLE via new processor, in response to MLE receiving new processor added via SCI
630
Enable STM for new processor
640

Figure 6A

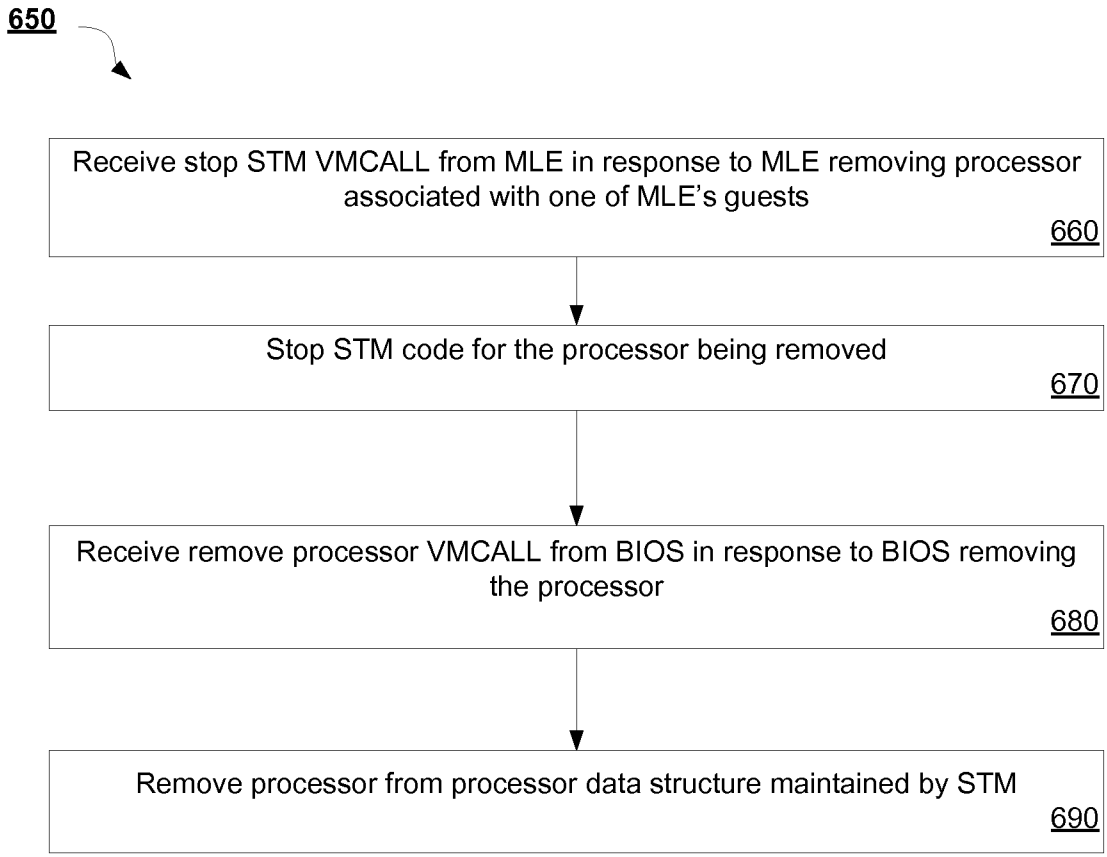
650
Receive stop STM VMCALL from MLE in response to MLE removing processor associated with one of MLE's guests
660
Stop STM code for the processor being removed
670
Receive remove processor VMCALL from BIOS in response to BIOS removing the processor
680
Remove processor from processor data structure maintained by STM
690

Figure 6B

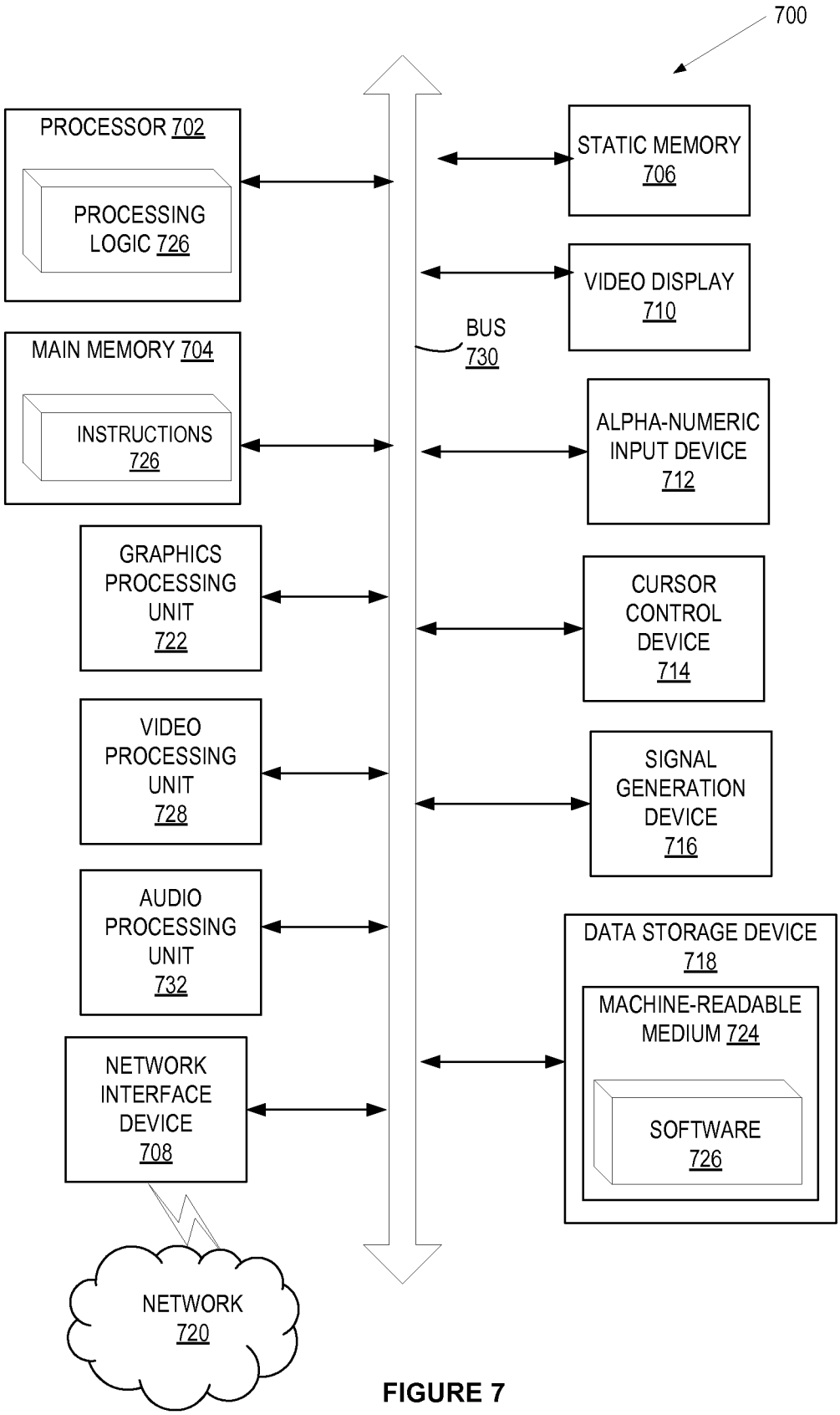
700
PROCESSOR 702
PROCESSING LOGIC 726
MAIN MEMORY 704
INSTRUCTIONS 726
GRAPHICS PROCESSING UNIT 722
VIDEO PROCESSING UNIT 728
AUDIO PROCESSING UNIT 732
NETWORK INTERFACE DEVICE 708
NETWORK 720
BUS 730
STATIC MEMORY 706
VIDEO DISPLAY 710
ALPHA-NUMERIC INPUT DEVICE 712
CURSOR CONTROL DEVICE 714
SIGNAL GENERATION DEVICE 716
DATA STORAGE DEVICE 718
MACHINE-READABLE MEDIUM 724
SOFTWARE 726

FIGURE 7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2013/072294

A. CLASSIFICATION OF SUBJECT MATTER

G06F 21/00 (2013.01) 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)

IPC: G06F 21/-, 17/-

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

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

CNABS;CNTXT;CNKI;VEN;USTXT;EPTXT;WOTXT;CATXT: INTEL,SMI, interrupt, SMM, BIOS, STM, transfer+, monitor, manager, privilege, hot, plug, memory, ROM, RAM, processor, CPU, error, check, verify, pass+, control

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y A	US 2006294149 A1 (INTEL CORP) 28 December 2006 (28.12.2006) see claims 17-20, description, paragraphs [0022]-[0034] and figures 1-4	1-7, 9-14, 16, 17, 25 8,15,18-24
Y	US 2008163209 A1 (ROZAS C V et al.) 03 July 2008 (03.07.2008) see figure 1 and description, paragraphs [0021]-[0024]	1-7, 9-14, 16, 17, 24
A	US 2009119748 A1(LONG Q et al.) 07 May 2009(07.05.2009) see the whole document	1-25
A	US 7447943 B2(HEWLETT-PACKARD DEV CO LP)04 November 2008(04.11.2008) see the whole document	1-25
A	US 7117311 B1 (INTEL CORP) 03 October 2006(03.10.2006) see the whole document	1-25

☐ 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
31 October 2013 (31.10.2013)	**21 Nov. 2013 (21.11.2013)**
Name and mailing address of the ISA/CN The State Intellectual Property Office, the P.R.China 6 Xitucheng Rd., Jimen Bridge, Haidian District, Beijing, China 100088 Facsimile No. 86-10-62019451	Authorized officer WANG,Yue Telephone No. (86-10)62411848

Form PCT/ISA /210 (second sheet) (July 2009)

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2013/072294

Patent Documents referred in the Report	Publication Date	Patent Family	Publication Date
US 2006294149 A1	28.12.2006	None	
US 2008163209 A1	03.07.2008	None	
US 2009119748 A1	07.05.2009	None	
US 7447943 B2	04.11.2008	US 2004243884 A1	02.12.2004
US 7117311 B1	03.10.2006	None	