



(51) International Patent Classification:

G06F 21/57 (2013.01) G06F 9/44 (2006.01)
G06F 21/51 (2013.01)

(21) International Application Number:

PCT/US2016/030031

(22) International Filing Date:

29 April 2016 (29.04.2016)

(25) Filing Language:

English

(26) Publication Language:

English

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

(72) Inventor: LAFFEY, Thomas M.; 8000 Foothills Blvd., Roseville, California 95747 (US).

(74) Agent: FERGUSON, Christopher Ward et al.; Hewlett Packard Enterprise, 3404 E. Harmony Road Mail Stop 79, Fort Collins, Colorado 80528 (US).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA,

LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

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

Declarations under Rule 4.17:

- as to the identity of the inventor (Rule 4.17(i))
- as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))

Published:

- with international search report (Art. 21(3))

(54) Title: EXECUTING PROTECTED CODE

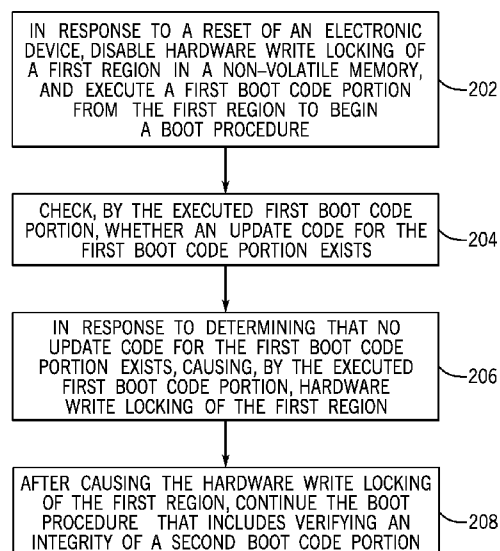


FIG. 2

(57) Abstract: In some examples, in response to a reset of an electronic device, a method disables hardware write locking of a first region in a non-volatile memory, and executes a first boot code portion from the first region to begin a boot procedure. The executed first boot code portion checks whether an update code for the first boot code portion exists. In response to determining that no update code for the first boot code portion exists, the executed first boot code portion causes hardware write locking of the first region. After causing the hardware write locking of the first region, the boot procedure continues, the boot procedure comprising verifying an integrity of a second boot code portion.

EXECUTING PROTECTED CODE

Background

[0001] An electronic device can include code to perform various tasks of the electronic device. Such code can include boot code to start up the electronic device, an operating system, and applications. The code in an electronic device can be subject to attack such that an unauthorized entity can gain access to the electronic device, or can introduce malicious code to cause damage to or to obtain information of the electronic device.

Brief Description Of The Drawings

[0002] Some implementations are described with respect to the following figures.

[0003] Fig. 1 is a block diagram of an example electronic device according to some examples.

[0004] Fig. 2 is a flow diagram of a process according to some examples.

[0005] Fig. 3 illustrates regions in a non-volatile memory according to some examples.

[0006] Figs. 4A-4B are a flow diagram of a process according to further examples.

[0007] Fig. 5 is a block diagram of the electronic device according to further examples.

[0008] Fig. 6 is a block diagram of a storage medium storing machine-readable instructions according to some examples.

Detailed Description

[0009] Examples of electronic devices can include any or some combination of the following: a communication device (e.g., a switch, a router, a gateway, etc.), a

server computer, a storage system, a desktop computer, a notebook computer, a tablet computer, a smart phone, a game appliance, a wearable device (e.g., a smart watch, smart eyeglasses, etc.), a vehicle (or an electronic device in the vehicle), a monitor device, and so forth. An electronic device can execute various code (in the form of machine-readable instructions) to perform respective tasks of the electronic device. Examples of such code include boot code (which performs tasks associated with starting up the electronic device), an operating system (which manages hardware and other resources of the electronic device and provides services to programs executing in the electronic device), applications that run within the environment of an operating system, and/or other code.

[0010] Startup tasks performed by the boot code for starting up an electronic device can include initialization of hardware components of the electronic device, and loading of the operating system to execute in the electronic device. The boot code can check hardware components to ensure that the hardware components are present and functioning properly. The boot code can also perform other tasks, including providing security mechanisms to protect against an attack, such as to protect content from unauthorized modification, providing cryptographic services, and so forth. Examples of boot code include a Basic Input/Output System (BIOS) code, a Unified Extensible Firmware Interface (UEFI) code, or another type of boot code.

[0011] Generally, the boot code executes prior to the loading of an operating system into an electronic device. In fact, the boot code is what controls the loading of the operating system. Note, however, that there may be portions of the boot code that can execute after the operating system has been loaded.

[0012] An electronic device can be attacked in one of two general ways. A first type of attack involves an attack by malicious code (in the form of machine-readable instructions). The malicious code can be introduced into the electronic device in one of several ways. For example, the malicious code can be downloaded to the electronic device from a network, or the malicious code can be introduced into the system from a storage device.

[0013] A second type of attack involves a physical attack, where an attacker (e.g., a user) can physically alter the electronic device to induce malicious behavior, such as by replacing a memory with new memory that contains unauthorized content (such as malicious code), adding an electronic component to the electronic device to cause unauthorized behavior, and so forth.

[0014] In accordance with some implementations of the present disclosure, protection mechanisms or techniques are provided to protect against attacks on the code of the electronic device, including lower level code such as the boot code as well as higher level code such as the operating system or an application. In some examples, the protection mechanisms or techniques can involve use of trusted boot code, hardware write locking of regions in a non-volatile memory (where the regions can be used to store different code portions), and update writeable regions in which update code for updating code portions can be stored.

[0015] “Trusted boot code” can refer to boot code that is provided by a trusted entity (or multiple trusted entities) and thus can be relied upon to not have been modified by any unauthorized entity. For example, the trusted boot code can be written by the manufacturer (or a trusted supplier of the manufacturer) of an electronic device in which the trusted boot code is to be executed.

[0016] In addition, according to some implementations, by providing redundant versions of code, including boot code as well as higher level code, corrupted code can be replaced with a non-corrupted version of the code to provide a self-recovery technique that allows the electronic device to recover from corrupted code.

[0017] In alternative examples, wherein the boot code includes portions that cannot be trusted, the protection mechanisms or techniques can employ a first memory region to store boot code that is accessible by a device processor, a second memory region to store boot code that is inaccessible by the device processor, and a boot controller that is able to maintain the device processor in reset while the boot code in the first memory region is validated and replaced with the boot code in the

second memory region if found to be invalid. This provides an alternative self-recovery technique.

[0018] Fig. 1 is a block diagram of an example electronic device 100, which includes a non-volatile memory 102 that includes multiple regions for storing different portions of code of the electronic device 100. In some examples, the non-volatile memory 102 can include a flash memory device (or multiple flash memory devices). In other examples, the non-volatile memory can be implemented with different types of memory devices that do not lose their content when power is removed from the electronic device 100, such as memristor device(s), spin torque memory device(s), phase change memory device(s), and so forth.

[0019] The multiple regions (which collectively make up a storage area) of the non-volatile memory 102 include a first region 104 to store a first portion of a boot code in the electronic device 100 (this first portion is referred to as a “first boot code portion”); a second region 106 to store a second portion of the boot code (referred to as the “second boot code portion”); a third region 108 to store product code, such as an operating system and/or application(s). Although just three regions are shown in Fig. 1, it is noted that in other examples, additional regions can be defined in the non-volatile memory 102 to store other information, such as configuration data that relates to a configuration of the electronic device 100. As examples, the configuration data can specify an operational feature (or features) of the electronic device 100. For example, if the electronic device 100 is a communication switch or router, then the configuration data can specify how ports are partitioned for use with respective different virtual local area networks (VLANs). Other types of configuration data can be used in other examples.

[0020] Although two regions 104 and 106 are depicted as storing respective first and second boot code portions, it is noted that the non-volatile memory 102 can be divided into more regions for storing greater than two portions of the boot code. In yet further examples, instead of storing multiple portions of the boot code in respective different regions of the non-volatile memory 102, the entirety of the boot code can be stored in just one region in the non-volatile memory 102.

[0021] The electronic device 100 further includes a memory controller 110 that manages access of the non-volatile memory 102. The memory controller 110 can be implemented with a hardware processing circuit, which can be in the form of a microcontroller, a microprocessor, a core (or cores) of a multi-core microprocessor, a programmable integrated circuit, a programmable gate array, and so forth. In further examples, the memory controller 110 can be implemented with a combination of machine-readable instructions and a hardware processing circuit.

[0022] The memory controller 110 can receive a memory access request (e.g., read request to read data or write request to write data) from a requester in the electronic device 100, where the requester can include a device processor 116, an input/output (I/O) device, or other type of device that is able to access the non-volatile memory 102. The memory controller 110 receives the memory access request and issues, to the non-volatile memory 102, a memory command (or multiple memory commands) to perform the requested access operation.

[0023] Although the memory controller 110 is depicted as being separate from the non-volatile memory 102, it is noted that in other examples, the memory controller 110 can be part of the non-volatile memory 102.

[0024] In some examples, the memory controller 110 includes a write lock control logic 112 that can selectively control write locking of the respective regions 104, 106, and 108 of the non-volatile memory 102. The write lock control logic 112 can be implemented with a portion of the hardware processing circuit of the memory controller 110, or can be implemented with machine-readable instructions executed by the memory controller 110.

[0025] Write locking a region of the non-volatile memory 102 refers to preventing or disabling a write to the region of the non-volatile memory 102. Disabling the write locking of the region of the non-volatile memory 102 refers to enabling a write to the region. Since the write lock control logic 112 is part of the memory controller 110, the write locking provided by the write lock control logic 112 is referred to as hardware write locking.

[0026] In alternative examples, instead of enabling or disabling write locking of the regions 104, 106, and 108 of the non-volatile memory 102 using the write lock control logic 112 in the memory controller 110, a write lock control logic 114 can be provided in the non-volatile memory 102, to selectively provide hardware write locking of the regions 104, 106, and 108.

[0027] In some examples, the write lock control logic 112 or 114 can be implemented as a register (or registers) containing respective indicators (e.g., bits) for the respective regions 104, 106, and 108. If a respective indicator in the register(s) is set to a first value, then hardware write locking of the respective non-volatile memory region is enabled. On the other hand, if the respective indicator in the register(s) is set to a second, different value, then hardware write locking of the respective non-volatile memory region is disabled. In other examples, the write lock control logic 112 or 14 can be implemented with other types of circuitry.

[0028] The device processor 116 is the main processor of the electronic device 100, which executes various code of the electronic device 100, including the boot code, operating system, and application(s), as examples. The device processor 116 can be implemented as a microprocessor, a core (or cores) of a multi-core microprocessor, a microcontroller, a programmable integrated circuit device, a programmable gate array, or any other hardware processing circuit.

[0029] In some examples, the various code portions in the regions 104, 106, and 108 of the non-volatile memory 102 are executable by the device processor 116.

[0030] Fig. 2 is a flow diagram of an example process of executing protected code in the electronic device 100 according to some implementations. Executing protected code can refer to executing code in a manner that prevents the electronic device from being subjected to an attack by an unauthorized entity, either in the form of malware or a physical attack.

[0031] In response to a reset of the electronic device 100, the process of Fig. 2 disables (at 202) hardware write locking (as provided by the write lock control logic

112 or write lock control logic 114 of Fig. 1) of a first region (e.g., 104) in a non-volatile memory (e.g., 102 in Fig. 1). More specifically, task 202 can include disabling hardware write locking of multiple regions (e.g., 104, 106, and 108) in the non-volatile memory 102. Disabling the hardware write locking of the first region (and possibly other regions) of the non-volatile memory 102 includes the write lock control logic 112 or 114 setting respective indicator(s) to a value to disable the hardware write locking. In response to the reset, the process further executes (at 202) a first boot code portion from the first region (e.g. 104 in Fig. 1) of the non-volatile memory 102 to being a boot procedure.

[0032] A reset of an electronic device 100 can refer to a reset that is caused by activation of a hardware component, such as a power button, a keystroke combination on a keyboard, or other hardware component, to indicate that a restart of the electronic device 100 is requested. A reset of the electronic device 100 can also be initiated by code executed in the electronic device 100.

[0033] A restart of the electronic device 100 can involve powering off at least some portion of the components of the electronic device 100, followed by re-applying power to such portion of the components to begin operation of the electronic device 100. The reset of the electronic device 100 causes the write lock control logic 112 or 114 to remove the hardware write locking of the regions 104, 106, and 108.

[0034] The process of Fig. 2 further includes checking (at 204), by the executed first boot code portion, whether an update code for the first boot code portion exists. "Update code" can refer to machine-readable instructions that are provided to replace or modify at least a portion (or the entirety) of a respective code portion, in this case the first boot code portion. In response to determining that no update code for the first boot code portion exists, the process of Fig. 2 further includes causing (at 206), by the executed first boot code portion, hardware write locking of the first region of the non-volatile memory 102. For example, the first boot code portion can program the write lock control logic 112 or 114 to enable the hardware write locking of the first region.

[0035] After causing the hardware write locking of the first region of the non-volatile memory 102, the process continues (at 208) with the remainder of the boot operation, where the boot procedure includes verifying an integrity of a second boot code portion, such as that in the second region 106 of the non-volatile memory 102.

[0036] Verifying an integrity of a code portion can refer to checking that the code portion has not been modified in any way, or that the code portion is from an authorized source. In some examples, verifying the integrity of the code portion can be based on a digital signature of the code portion. The digital signature can be produced by hashing the code portion to produce a hash value, and encrypting the hash value using a private key to produce an encrypted value. The encrypted value, along with other information, can form the digital signature of the code portion. An accessing entity that accesses the code portion can decrypt the digital signature using a public key to retrieve a hash value, which can be compared with a hash value generated by the accessing entity from the code portion at the time of accessing the code portion. If the hash values match, then the integrity of the code portion is verified.

[0037] Fig. 3 shows regions of the non-volatile memory 102 according to further implementations. The non-volatile memory 102 includes a reset block segment 302, an extended boot loader segment 303, and a product code segment 305. The reset block segment 302 includes the first region 104 that stores the first boot code portion, and an update writeable region 304 to store update code for the first boot code portion. In examples according to Fig. 3, the first boot code portion is a reset block of the boot code. In some examples, the reset block can include an initial boot loader and a core root of trust. The initial boot loader can also be referred to as a boot block, and is the initial portion of the boot code that is considered to be trustworthy and that is executed first before the rest of the boot code is allowed to execute on the device processor 116 in the electronic device 100. The initial boot loader can be used to check the integrity of the boot code as well as to perform other initial functions. The core root of trust code can refer to another part of the boot code that is to provide various protections of the boot code, including protecting content

from unauthorized modification, performing cryptographic operations including verifying digital signatures of updates of portions of the boot code, and other tasks.

[0038] The update writeable region 304 can be used to store the update code for the reset block in the region 104. If it is desired to update the reset block, the update code for the reset block is written to the update writeable region 304. During a boot procedure, the reset block in the region 104 can be replaced with the update code in the update writeable region 304.

[0039] The extended boot loader segment 303 includes redundant regions 306 and 308 and an update writeable region 310. In some examples, each region 306 or 308 stores a respective version of the second boot code portion. In some examples, the second boot code portion includes an extended boot loader, which is the portion of the boot code that follows the initial boot loader. Generally, the extended boot loader can perform initialization of hardware components of the electronic device 100 and the loading of the operating system of the electronic device 100. The redundant regions 306 and 308 are to store redundant versions of the extended boot loader. Collectively, the redundant regions 306 and 308 can correspond to the second region 106 shown in Fig. 1.

[0040] Generally, the versions of the extended boot loader stored in the extended boot loader redundant regions 306 and 308 are the same. However, due to an update or another cause (such as corruption due to malware or another source), one of the versions of the extended boot loader in one of the regions 306 and 308 can become different from the other version of the extended boot loader in the other region 306 or 308.

[0041] The update writeable region 310 in the extended boot loader segment 303 is used to store update code for the extended boot loader. If it is desired to update the extended boot loader, the update code for the extended boot loader is written to the update writeable region 310. During a boot procedure, the extended boot loader in the region 306 or 308 can be replaced with the update code in the update writeable region 310.

[0042] The product code segment 305 includes product code redundant regions 312 and 314 and an update writeable region 316. Each of the redundant regions 312 and 314 is to store a version of the product code that is to execute in the electronic device 100. The product code can include the operating system, application(s), and so forth. Collectively, the redundant regions 312 and 314 can correspond to the third region 108 shown in Fig. 1.

[0043] The redundant regions 312 and 314 usually store the same version of the product code. However, due to an update, or due to a different cause, one of the versions of the product code in the region 312 or 314 can become different from the version of the device code in the other of the regions 312 and 314.

[0044] Update code for the product code 316 can be stored in the update writeable region 316, for use in updating the product code in the region 312 or 314 during a boot procedure.

[0045] Although not shown in Fig. 3, another region of the non-volatile memory 102 can include similar regions for storing configuration data, including multiple redundant regions to store redundant versions of the configuration data, as well as an update writeable region to store update configuration data.

[0046] In some examples, the update writeable regions 304, 310, and 316 are writeable during both a boot procedure and normal operating procedure (after the operating system has been loaded) of the electronic device 100.

[0047] Figs. 4A-4B depict a flow diagram of a process of booting the electronic device 100 according to further examples. The process of Fig. 4 refers to the regions of the non-volatile memory 102 depicted in Fig. 3. In response to a reset of the electronic device 100, hardware write locking of the regions 104, 306, 308, 312, and 314 in the non-volatile memory 102 is disabled (at 402). The reset block is executed (at 404) from the region 302 of the first region 102.

[0048] The executed reset block determines (at 406) whether there is a valid update code for the reset block in the update writeable region 304 of the first region

104. The determination of whether there is a valid update code in the update writeable region 304 can be based on checking if an update code has been written to the update writeable region 304, and checking a digital signature of such update code in the update writeable region 304 is valid.

[0049] If there is no valid update code in the update writeable region 304 (i.e., an update code has not been written to the update writeable region 304 or the update code that has been written is not valid), the executed reset block causes (at 408) the write lock control logic 112 or 114 to activate hardware write locking of the region 104 that stores the reset block.

[0050] The executed reset block then checks (at 410) whether there is a valid update code for the extended boot loader in the update writeable region 310. If not, the reset block verifies (at 412) the integrity of the versions of the extended boot loader in the regions 306 and 308. If the integrity of both versions of the extended boot loader in the regions 306 and 308 is verified (i.e., both versions of the extended boot loader are valid), the executed reset block causes (at 414) the write lock control logic 112 or 114 to activate hardware write locking of the regions 306 and 308, and selects (at 416) one of the versions of the extended boot loader (e.g., at random or using another criterion) for execution.

[0051] If one of the versions of the extended boot loader in the regions 306 and 308 is valid but the other version of the extended boot loader is not valid, then the executed reset block replaces (at 418) the invalid version of the extended boot loader in one of the regions 306 and 308 with the valid version of the extended boot loader in the other one of the regions 306 and 308. The process then continues to task 414.

[0052] However, if both versions of the extended boot loader in the regions 306 and 308 are invalid, then the executed reset block indicates (at 420) an error condition, and the boot procedure is not allowed to continue.

[0053] Assuming that a version of the extended boot loader is executed, the executed reset block, the executed extended boot loader checks (at 422) whether there is a valid update code for the product code in the update writeable region 316. If there is no valid update code for the product code in the update writeable region 316, the executed extended boot loader verifies (at 424) the integrity of the versions of the product code in the regions 312 and 314.

[0054] If the integrity of both versions of the product code in the regions 312 and 314 is verified (i.e., both versions of the extended boot loader are valid), the executed extended boot loader causes (at 426) the write lock control logic 112 or 114 to activate hardware write locking of the regions 312 and 314, and selects (at 428) one of the versions of the product code (e.g., at random or using another criterion) for execution, which causes loading of the operating system and application(s).

[0055] If one of the versions of the product code in the regions 312 and 314 is valid but the other version of the product code is not valid, then the executed extended boot loader replaces (at 430) the invalid version of the product code in one of the regions 312 and 314 with the valid version of the extended boot loader in the other one of the regions 312 and 314.

[0056] However, if both versions of the product code in the regions 312 and 314 are invalid, then the executed extended boot loader indicates (at 432) an error condition, and the boot procedure is not allowed to continue.

[0057] The code update process is discussed below.

[0058] As noted above, tasks 406, 410, and 422 check for presence of respective valid update codes in respective update writeable regions 304, 310, and 316. Checking for the validity of an update code involves confirming that the digital signature of the respective update code is valid. In some examples, the checking of the validity of an update code in a respective update writeable region is performed by the same code that wrote the update code to the respective update writeable region.

[0059] If the executed reset block determines (at 406) that there is a valid update code in the update writeable region 304 of the reset block segment 304, the executed reset block erases (at 440) the region 104 that stores the reset block, and re-writes (at 442) the region 104 with the update code in the update writeable region 304. The process then continues to task 408.

[0060] If the executed reset block determines (at 410) that there is a valid update code in the update writeable region 310 of the extended boot loader segment 303, the executed reset block erases (at 444) one of the regions 306 and 308 that store the versions of the extended boot loader, and re-writes (at 446) the erased region 306 or 308 with the update code in the update writeable region 310. The process then continues to task 412.

[0061] If the executed extended boot loader determines (at 422) that there is a valid update code in the update writeable region 316 of the product code segment 305, the executed extended boot loader erases (at 448) one of the regions 312 and 314 that store the versions of the product code, and re-writes (at 450) the erased region 312 or 314 with the update code in the update writeable region 316. The process then continues to task 324.

[0062] Although not shown in Figs. 4A-4B, prior to updating the product code in the region 312 or 314 with the update code for the product code, a reset of the electronic device 100 may first be performed.

[0063] Fig. 5 is a block diagram of an electronic device 500 according to further implementations. The electronic device 500 includes a processor 502, and first and second memory regions 504 and 506. The first and second memory regions 504 and 506 can be part of different memory devices, or they can be part of the same memory device (or part of the same collection of memory devices). The second memory region 506 can be stored in a non-volatile memory device. The first memory region 504 can be part of a non-volatile memory device or a volatile memory device such as a dynamic random access memory (DRAM) or static random access memory (SRAM).

[0064] The first memory region 504 is to store boot code 508 that is accessible by the device processor 502, while the second memory region 506 is to store boot code 510 that is inaccessible by the device processor 502 but is accessible by a boot controller 512. Note that the boot code 508 in the first memory region 504 is also accessible by the boot controller 512.

[0065] The first memory region 504 is part of the address space of the device processor 502. The address space of the device processor 502 includes addresses that can be accessed by the device processor 502. The second memory region 506 has an address space that is outside the address space of the device processor 502.

[0066] In some examples, the first memory region 504 and the second memory region 506 can be partitioned into regions as depicted in Fig. 1 or 3.

[0067] The boot controller 512 can be implemented as a hardware processing circuit or as a combination of machine-readable instructions and a hardware processing circuit. The boot controller 512 outputs a reset signal 514 to the device processor 502. During booting of the electronic device 500, the boot controller 512 activates the reset signal 514 that is provided to the device processor 502. When the reset signal 514 is activated, the device processor 502 is maintained in reset. When the reset signal 514 is deactivated, the device processor 502 is allowed to execute machine-readable instructions.

[0068] Maintaining the device processor 502 in reset can refer to the device processor 502 being unable to maintain a state in which the device processor 502 is unable to execute machine-readable instructions (in other words, the device processor 502 is inactive).

[0069] During the booting of the electronic device 500, the boot controller 512 can perform the following additional tasks. The boot controller 512 can verify an integrity of the boot code 508 in the first memory region 504, and can update the boot code 508 in the first memory region 504 with the boot code 510 in the second memory region 506 in response to the verifying indicating that the boot code 508 in

the first memory region 504 is invalid. In response to verifying the integrity of the boot code 508 in the first memory region 504, the boot controller 512 deactivates the reset signal 514 to allow the device processor 502 to execute the boot code 508 in the first memory region 504.

[0070] In response to verifying the integrity of the boot code in the first memory region 504 and determining that a version of the boot code 508 in the first memory region 504 is more up-to-date than a version of the boot code 510 in the second memory region 506, the boot controller 512 updates the boot code 510 in the second memory region 506 with the boot code 508 in the first memory region 504.

[0071] Fig. 6 shows a non-transitory machine-readable storage medium 600 that stores various machine-readable instructions that are executable on a device processor. The machine-readable instructions include instructions 602 to, in response to a reset of the electronic device, disable hardware write locking of a plurality of regions in a non-volatile memory, and execute a first boot code portion from a first region of the plurality of regions to begin a boot procedure.

[0072] The machine-readable instructions further include instructions 604 to check, by the executed first boot code portion, whether an update code for the first boot code portion exists.

[0073] The machine-readable instructions further include instructions 606 to, in response to determining that no update code for the first boot code portion exists, cause, by the executed first boot code portion, hardware write locking of the first region.

[0074] The machine-readable instructions further include instructions 608 to, after causing the hardware write locking of the first region, validate, by the first code boot portion, a second boot code portion in a second memory region of the plurality of memory regions.

[0075] The machine-readable instructions further include instructions 610 to, in response to validating the second boot code portion, cause hardware write locking of the second region and execute the second boot code portion.

[0076] The storage medium 600 can include one or multiple different forms of memory including semiconductor memory devices such as dynamic or static random access memories (DRAMs or SRAMs), erasable and programmable read-only memories (EPROMs), electrically erasable and programmable read-only memories (EEPROMs) and flash memories; magnetic disks such as fixed, floppy and removable disks; other magnetic media including tape; optical media such as compact disks (CDs) or digital video disks (DVDs); or other types of storage devices. Note that the instructions discussed above can be provided on one computer-readable or machine-readable storage medium, or alternatively, can be provided on multiple computer-readable or machine-readable storage media distributed in a large system having possibly plural nodes. Such computer-readable or machine-readable storage medium or media is (are) considered to be part of an article (or article of manufacture). An article or article of manufacture can refer to any manufactured single component or multiple components. The storage medium or media can be located either in the machine running the machine-readable instructions, or located at a remote site from which machine-readable instructions can be downloaded over a network for execution.

[0077] In the foregoing description, numerous details are set forth to provide an understanding of the subject disclosed herein. However, implementations may be practiced without some of these details. Other implementations may include modifications and variations from the details discussed above. It is intended that the appended claims cover such modifications and variations.

What is claimed is:

- 1 1. A method of executing protected code, comprising:
2 in response to a reset of an electronic device, disabling hardware write locking
3 of a first region in a non-volatile memory, and executing a first boot code portion from
4 the first region to begin a boot procedure;
5 checking, by the executed first boot code portion, whether an update code for
6 the first boot code portion exists;
7 in response to determining that no update code for the first boot code portion
8 exists, causing, by the executed first boot code portion, hardware write locking of the
9 first region; and
10 after causing the hardware write locking of the first region, continuing the boot
11 procedure, the boot procedure comprising verifying an integrity of a second boot
12 code portion.
- 1 2. The method of claim 1, wherein the first boot code portion is executed on a
2 device processor in the electronic device.
- 1 3. The method of claim 1, wherein the hardware write locking of the first region is
2 performed by a controller of the non-volatile memory or by logic in the non-volatile
3 memory.
- 1 4. The method of claim 1, further comprising:
2 in response to determining that the update code for the first boot code portion
3 exists, updating, by the executed first boot code portion using the update code, the
4 first boot code portion in the first region.
- 1 5. The method of claim 4, further comprising storing the update code in an
2 update writeable region of the non-volatile memory, wherein the updating comprises
3 writing the update code stored in the update writeable region to the first region.

1 6. The method of claim 1, further comprising:
2 disabling hardware write locking of a second region in the non-volatile
3 memory in response to the reset of the electronic device;
4 verifying, by the executed first boot code portion, the integrity of the second
5 boot code portion in the second region; and
6 in response to verifying the integrity of the second boot code portion, causing,
7 by the executed first boot code portion, hardware write locking of the second region
8 and execution of the second boot code portion,
9 wherein continuing the boot procedure comprises executing the second boot
10 code portion.

1 7. The method of claim 6, further comprising:
2 checking, by the executed first boot code portion, whether a further update
3 code for the second boot code portion exists in a update writeable region for the
4 second boot code portion;
5 in response to determining that the further update code exists in the update
6 writeable region for the second boot code portion, updating, by the executed first
7 boot code portion using the further update code, the second boot code portion in the
8 second region.

1 8. The method of claim 7, wherein the non-volatile memory further includes a
2 redundant region to store a redundant version of the second boot code portion, the
3 method further comprising:
4 checking an integrity of the redundant version of the second boot code portion
5 in the redundant region; and
6 in response to detecting an invalid second boot code portion in one of the
7 second region and the redundant region, replacing the invalid second boot code
8 portion with a valid second boot code portion in another one of the second region
9 and the redundant region.

1 9. The method of claim 8, further comprising:
2 in response to detecting that both the second boot code portion in the second
3 region and redundant version of the second boot code portion of the redundant
4 region are invalid, indicating an error.

1 10. An electronic device to execute protected code, comprising:
2 a processor;
3 a first memory region to store boot code accessible by the processor;
4 a boot controller; and
5 a second memory region to store boot code that is inaccessible by the
6 processor but that is accessible by the boot controller, the boot controller to, during
7 booting of the electronic device:
8 activate a reset signal to the processor to maintain the processor in
9 reset,
10 determine an integrity of the boot code in the first memory region,
11 update the boot code in the first memory region with the boot code in
12 the second memory region in response to the determining indicating that the boot
13 code in the first memory region is invalid, and
14 in response to verifying the integrity of the boot code in the first
15 memory region, deactivate the reset signal to allow the processor to execute the
16 boot code in the first memory region.

1 11. The electronic device of claim 10, wherein the determining of the integrity of
2 the boot code in the first memory region comprises determining that a signature of
3 the boot code in the first memory region is valid.

12. The electronic device of claim 10, wherein the boot controller is to further:
in response to verifying the integrity of the boot code in the first memory region and determining that a version of the boot code in the first memory region is more up-to-date than a version of the boot code in the second memory region, update the boot code in the second memory region with the boot code in the first memory region.

13. The electronic device of claim 10, wherein the second memory region is outside an address space of the processor, and the first memory region is part of the address space of the processor.

14. A non-transitory machine-readable storage medium storing instructions to execute protected code, the instructions upon execution causing an electronic device to:

in response to a reset of an electronic device, disable hardware write locking of a plurality of regions in a non-volatile memory, and execute a first boot code portion from a first region of the plurality of regions to begin a boot procedure;
check, by the executed first boot code portion, whether an update code for the first boot code portion exists;

in response to determining that no update code for the first boot code portion exists, cause, by the executed first boot code portion, hardware write locking of the first region;

after causing the hardware write locking of the first region, validate, by the first code boot portion, a second boot code portion in a second region of the plurality of regions; and

in response to validating the second boot code portion, cause hardware write locking of the second region and execute the second boot code portion.

1 15. The non-transitory machine-readable storage medium of claim 14, wherein
2 the non-volatile memory further includes a third region storing a redundant version of
3 the second boot code portion, and wherein executing the second boot code portion
4 is in response to a random selection between the second boot code portion in the
5 second region and the redundant version of the second boot code portion in the third
6 region.

1 / 6

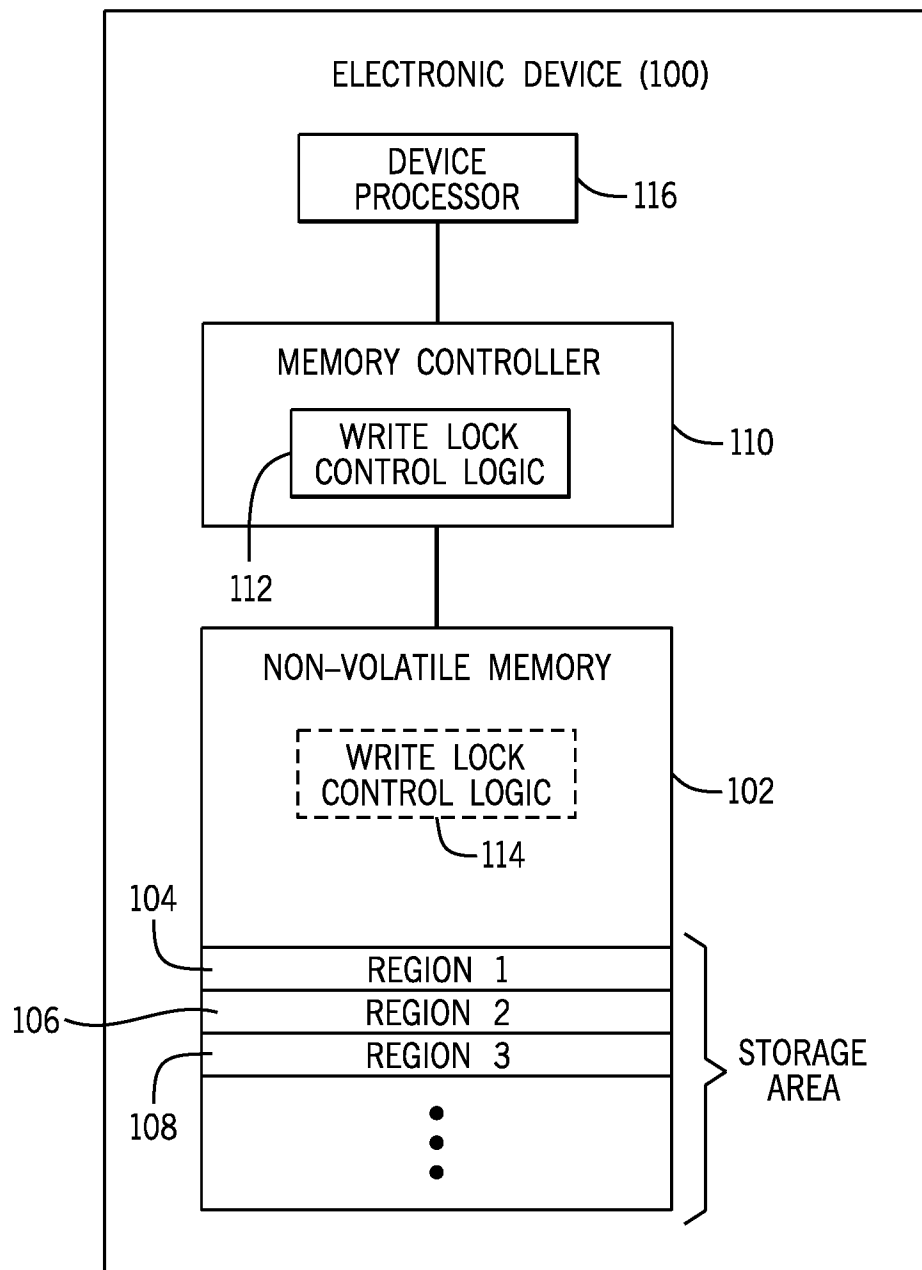


FIG. 1

2 / 6

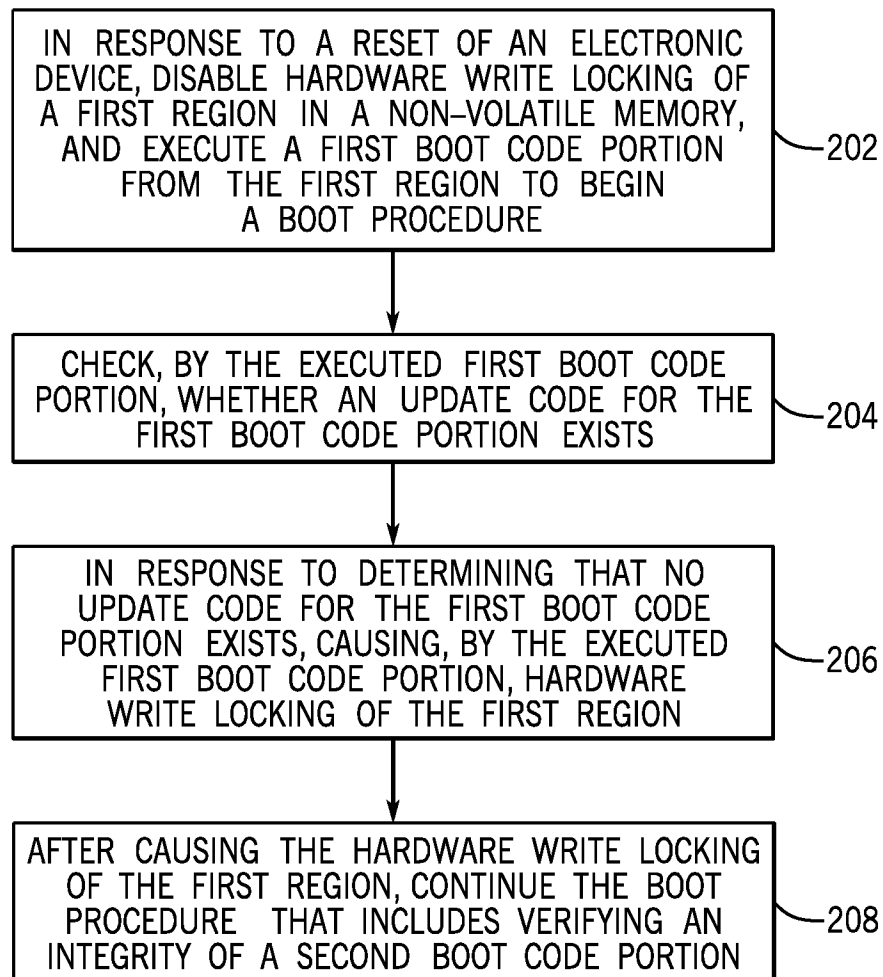


FIG. 2

3 / 6

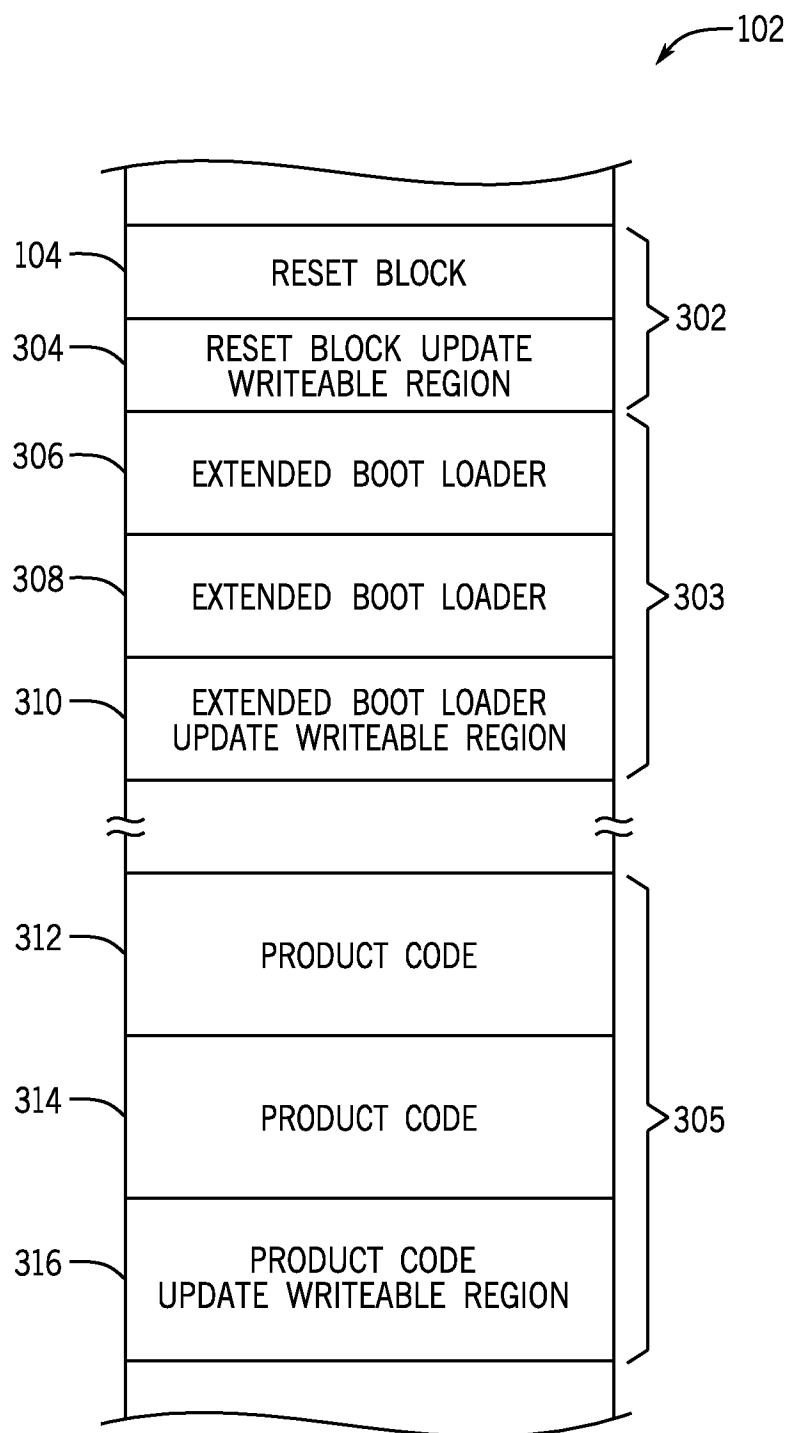


FIG. 3

4 / 6

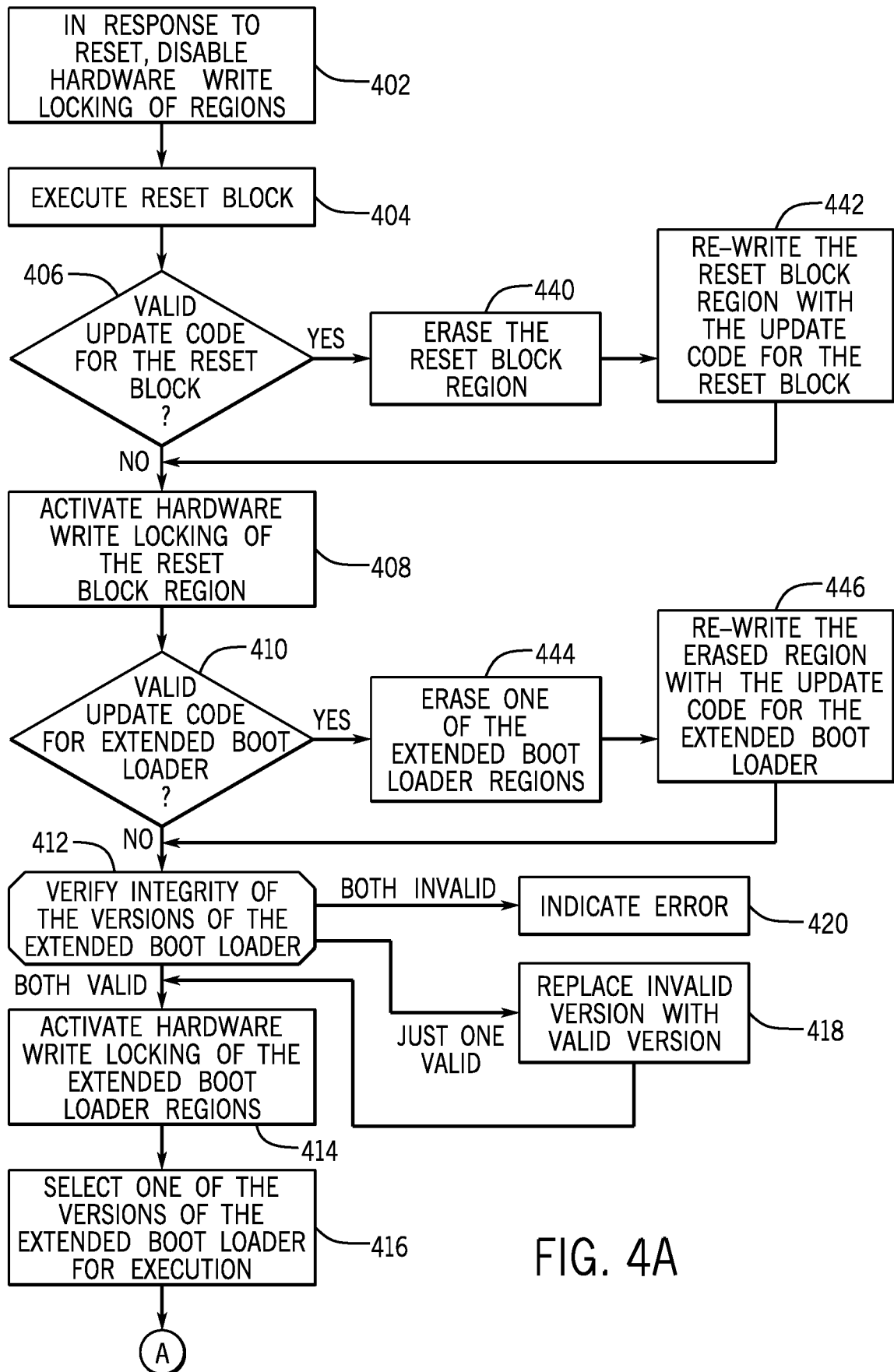


FIG. 4A

5 / 6

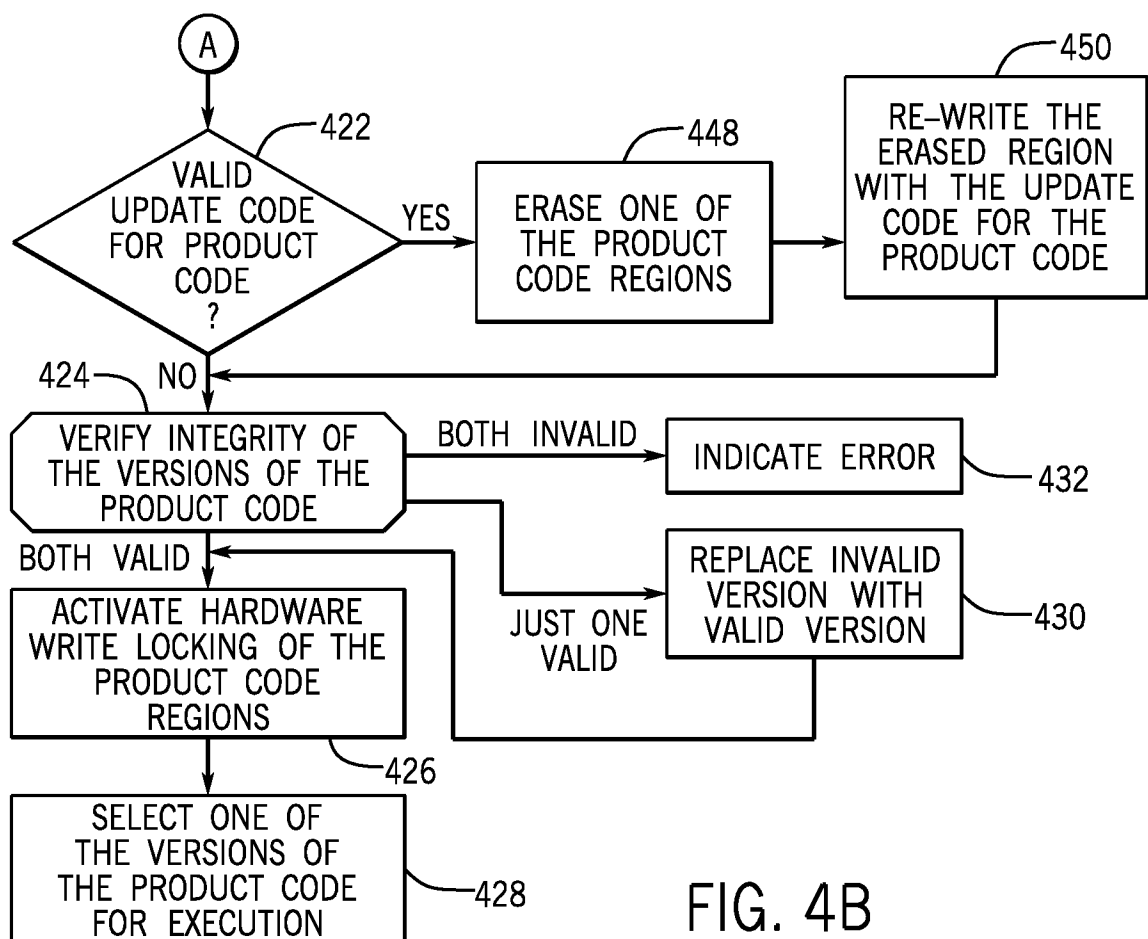


FIG. 4B

6 / 6

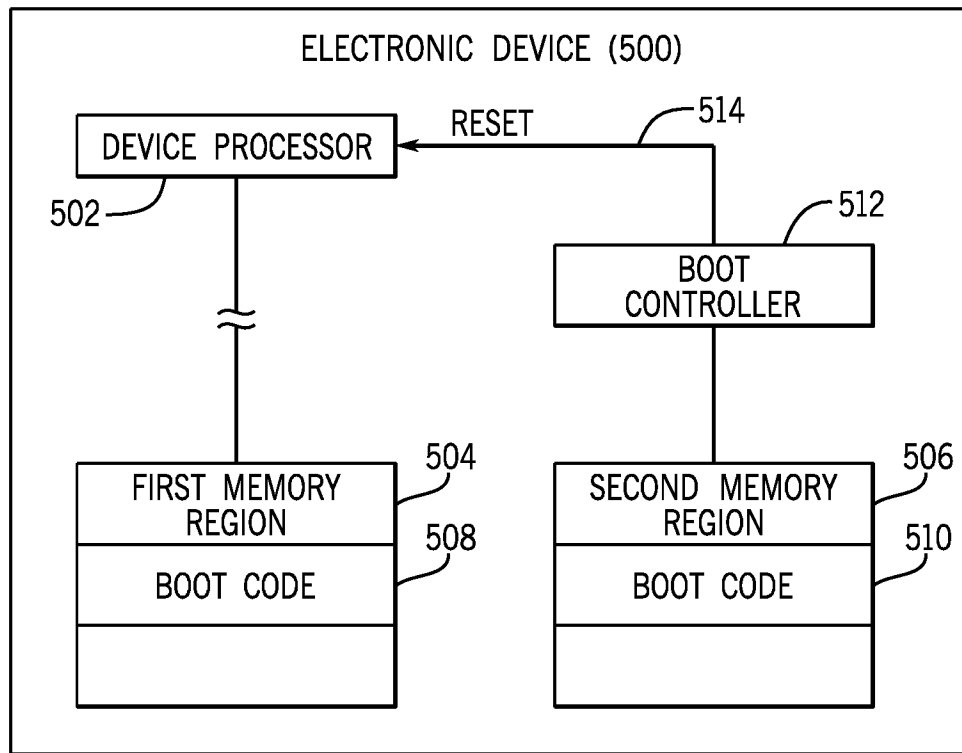


FIG. 5

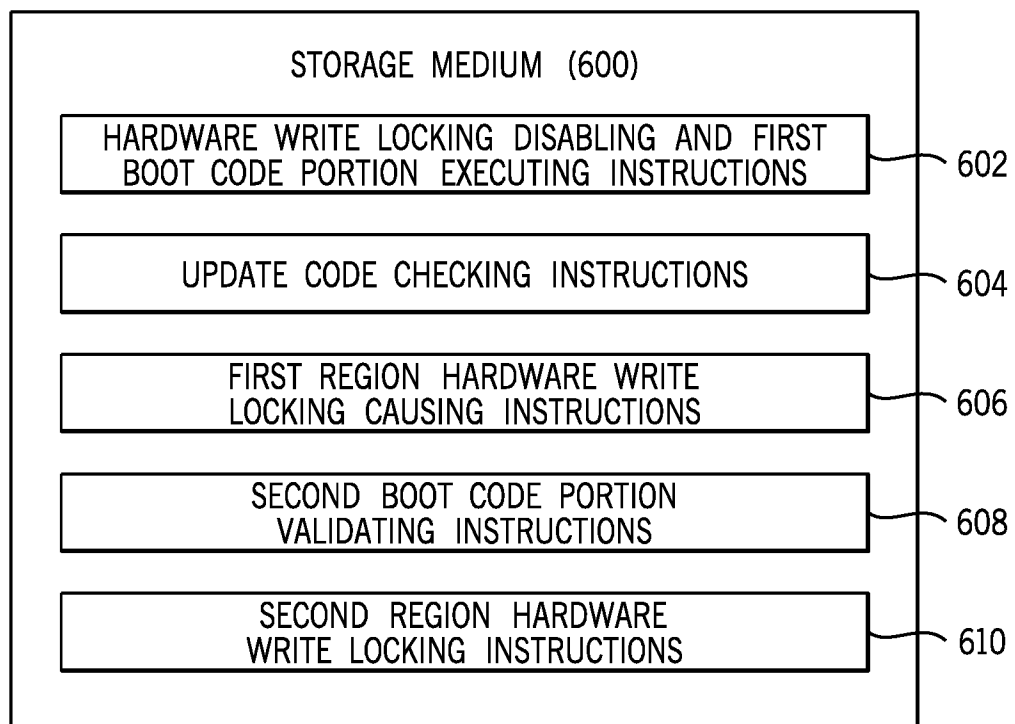


FIG. 6

A. CLASSIFICATION OF SUBJECT MATTER**G06F 21/57(2013.01)i, G06F 21/51(2013.01)i, G06F 9/44(2006.01)i**

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F 21/57; G06F 12/02; G06F 9/44; G06F 21/56; G06F 12/14; G06F 12/00; H04L 9/00; G06F 21/51

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

Korean utility models and applications for utility models

Japanese utility models and applications for utility models

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

eKOMPASS(KIPO internal) & Keywords: boot, code, integrity, reset, region, update, write locking, memory

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2016-0085558 A1 (HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P.) 24 March 2016 See paragraphs [0006], [0020], [0050]; claims 1, 9, 14; and figures 1-2.	1-15
A	US 2014-0082257 A1 (ROSS S. SCOULLER et al.) 20 March 2014 See paragraph [0013]; and figure 2.	1-15
A	US 2016-0055338 A1 (HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P.) 25 February 2016 See paragraph [0056]; and figure 3.	1-15
A	US 7058779 B1 (MARK ALAN MCCLAIN) 06 June 2006 See column 3, line 38 - column 4, line 5; and figure 2.	1-15
A	US 2006-0112266 A1 (HERBERT A. LITTLE et al.) 25 May 2006 See paragraph [0012]; and figure 2.	1-15



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

13 January 2017 (13.01.2017)

Date of mailing of the international search report

16 January 2017 (16.01.2017)

Name and mailing address of the ISA/KR

International Application Division

Korean Intellectual Property Office

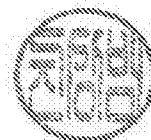
189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea

Facsimile No. +82-42-481-8578

Authorized officer

CHIN, Sang Bum

Telephone No. +82-42-481-8398



INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2016/030031

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2016-0085558 A1	24/03/2016	CN 105378663 A EP 3028147 A1 WO 2015-016867 A1	02/03/2016 08/06/2016 05/02/2015
US 2014-0082257 A1	20/03/2014	CN 103679067 A EP 2709034 A1 US 9390278 B2	26/03/2014 19/03/2014 12/07/2016
US 2016-0055338 A1	25/02/2016	CN 105122258 A EP 2989583 A1 EP 2989583 A4 I5 22838 B TW 201447630 A TW I522838 B WO 2014-175863 A1	02/12/2015 02/03/2016 19/10/2016 21/02/2016 16/12/2014 21/02/2016 30/10/2014
US 7058779 B1	06/06/2006	None	
US 2006-0112266 A1	25/05/2006	EP 1659472 A1	24/05/2006