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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
09 May 2019 (09.05.2019)

WIPO | PCT

(10) International Publication Number
WO 2019/088978 A1

(51) International Patent Classification:
G06F 12/14 (2006.01) *G06F 21/70* (2013.01)

(21) International Application Number:
PCT/US2017/059070

(22) International Filing Date:
30 October 2017 (30.10.2017)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P.** [US/US]; 10300 Energy Drive, Spring, Texas 77389 (US).

(72) Inventors: **VILLATEL, Maugan**; Filton Road Stoke Gifford, Pt., Bristol Bristol BS34 8QZ (GB). **DALTON, Chris**; Filton Road Stoke Gifford, Pt., Bristol Bristol BS34 8QZ (GB). **HUSCROFT, Carey**; Filton Road Stoke Gifford, Pt., Unit 700 The Quadrant, Bradley Stoke, Bristol Bristol BS34 8QZ (GB).

(74) Agent: **BURROWS, Sarah** et al.; 3390 E. Harmony Road, Mail Stop 35, 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, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, 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))*

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

(54) Title: SECURE HARDWARE INITIALIZATION

(57) Abstract: A method for secure hardware initialization during a start-up process comprises activating a protected portion of a physical memory, allocating a part of the protected portion of the physical memory for use by direct memory access, DMA, drivers and non-DMA related hardware initialization instructions, and using a memory management tool, allocating a first part of the physical memory, accessible by a device via the memory management tool, for use by data.

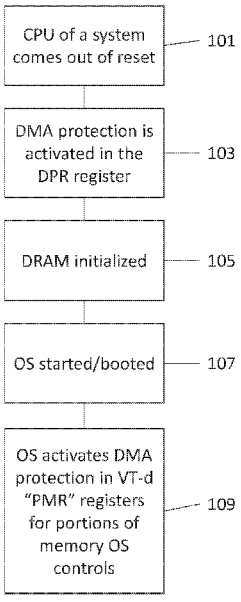

Figure 1

WO 2019/088978 A1

SECURE HARDWARE INITIALIZATION

BACKGROUND

[0001] Direct Memory Access (DMA) is a way for devices to access memory directly without the intervention of the CPU. This can significantly reduce CPU load, as the CPU does not need to read data in dynamic random-access memory (DRAM) and then push it to the device, or the other way around. Well-behaved devices will typically be fully under the control of a driver running in the Operating System (OS), and will only DMA when and where the driver wants it.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] Various features of certain examples will be apparent from the detailed description which follows, taken in conjunction with the accompanying drawings, which together illustrate, by way of example only, a number of features, and wherein:

[0003] Figure 1 is a flowchart of a boot process according to an example;

[0004] Figure 2 is a flowchart of a boot process according to an example;

[0005] Figure 3 is a flowchart of a method for secure hardware initialization during a start-up process according to an example;

[0006] Figure 4 is a schematic representation of a system according to an example;

[0007] Figure 5 is a schematic representation of a boot process according to an example; and

[0008] Figure 6 shows an example of a processor associated with a memory according to an example.

DETAILED DESCRIPTION

[0009] In the following description, for purposes of explanation, numerous specific details of certain examples are set forth. Reference in the specification to "an example" or similar language means that a particular feature, structure, or characteristic described in connection with the example is included in at least that one example, but not necessarily in other examples.

[0010] DMA attacks are a growing concern in computer security. They are very powerful, cheap, and increasingly easy to carry out due to the introduction of external DMA ports like Thunderbolt, making them very attractive to potential attackers.

[0011] There has been an increasing number of attacks using rogue DMA devices to attack systems. These attacks typically involve rogue PCIe devices that attackers would plug in to a system. PCIe (Peripheral Component Interconnect Express) is a high-speed bus that (among other things) allows PCIe devices to issue DMA transactions. It is, for example, used in desktops to connect a discrete graphics card to the motherboard.

[0012] Due to how DMA was designed, the device could then read or write any portion of memory (with the notable exception of System Management RAM SMRAM) without system software being aware of it happening, thus allowing attackers to compromise a system and/or retrieve encryption keys and other sensitive secrets from memory.

[0013] Being the first software executing on a platform, the basic input/output system (BIOS) has a critical role in enabling protection against potential rogue DMA devices, and then passing this protection to the Operating System. Failure to do so would enable an attacker to bypass security technologies like the Unified Extensible Firmware Interface (UEFI) Secure Boot for example. Recently, EFI firmware started to be a target for these attacks.

[0014] Historically, (except in the case of microkernels) drivers for all devices were running as part of the Operating System with very high privileges. However, with the growing popularity of virtualisation and hypervisors, systems

started to be shared among multiple, mutually-untrusted operating systems, separated by a hypervisor. Sharing devices between those different operating systems incurred a non-negligible performance cost, which is why there was a desire to be able to assign some peripherals to one single guest, so the guest could get the full performance of that device.

[0015] However, if this device was capable of issuing DMA transactions, it could allow the guest to read or write memory belonging to other guests, or to the hypervisor. As a result, chip makers started to develop input–output memory management unit (IOMMU) technology, which allows a privileged software (hypervisor) to restrict what memory a given DMA-capable device is able to access. This allowed hypervisors to ensure that a device assigned to a specific guest would only ever be able to read or write memory belonging to that guest. More generally, it allowed to "de-privilege" drivers of DMA-capable device, by ensuring a low-privilege driver could not order a device to DMA into memory not belonging to the driver.

[0016] According to an example, there is provided a method to secure devices against DMA attacks to prevent a DMA device from replacing executable code in memory with malicious code. For example, in a boot process, a BIOS can load itself into memory and continue its execution there. However, a DMA-capable attacker could potentially modify this image of the BIOS in memory and replace it with his own malicious BIOS. As BIOS will start executing from memory before initializing and locking SMM, the attacker could even take control of SMM.

[0017] Similarly, a trusted platform module (TPM) trusted boot module expects that software measures an image, sends that measurement to the TPM, and then executes that image. However, an attacker could potentially use DMA to modify the image between the moment it has been measured and the moment it is executed. He could also more simply attack the code responsible for measuring the image and sending the measurement to the TPM. This in turn impacts the security offered by technologies which rely on Trusted Boot and the

TPM to make sure that the encryption key is only released when an expected process has been booted.

[0018] Similar to the attack on Trusted Boot, an attacker can attack UEFI Secure Boot by modifying a UEFI image between the moment it is authenticated and the moment it is executed, or attack and deactivate the code responsible for doing the authentication.

[0019] According to an example, no window is left in which DRAM is not protected from DMA so that BIOS/EFI firmware and an Operating System, which relies on DRAM to store code, data etc., can be comprised.

[0020] There are several available DMA protection capabilities:

[0021] The Intel DPR ("DMA Protected Range") register, which is present in all recent Intel chipsets. It allows up to 256 MB of physical memory below the top of main memory segment (TSEG) to be protected against all DMA. It makes sure that only the processor can access this region of memory, and this register can also be locked until the next reboot. The areas of memory occupied by TSEG forms a memory hole with respect to OS such that the OS cannot see the TSEG area.

[0022] Intel VT-d - PMR ("Protected Memory Registers") includes registers that can define one region in low-memory and one region in high memory as being protected against DMA coming from downstream devices. These registers are described as a way to securely start to setup VT-d pagetables in memory.

[0023] Intel VT-d – Pagetables define what portion of memory downstream devices will be able to access. All other parts of memory will thus be protected against DMA coming from those devices. Pagetables must however be set up in memory, which implies that they cannot be used before memory is initialized, and that while they are being set up they must be protected from DMA using the PMR.

[0024] Similarly to the above, IOMMU (input–output memory management unit) can be used to provide memory management, and can connect a direct-memory-access–capable (DMA-capable) I/O bus to the main memory of a system. That is, there are different IOMMU specifications that can be used. For example, I/O Virtualization Technology, "AMD-Vi", originally called IOMMU as well as the Virtualization Technology for Directed I/O (VT-d).

[0025] According to an example, there is provided a method for secure hardware initialization during a start-up process. Initially, when a CPU comes out of reset, there are two ways to protect memory against DMA: the DPR and PMR. In an example, the DPR is used for early BIOS protection, as they are reserved for BIOS use (the BIOS can lock them), are independent from any IOMMU translation, and do not depend on having VT-d available on the platform. However, they can only protect 256M of memory just below TSEG. Therefore, in an example, while a BIOS is still executing in place from flash/Cache As RAM (i.e. during a pre-EFI initialisation phase, PEI), those registers are set so that the regions of DRAM that the BIOS will use will be DMA protected.

[0026] Since some drivers (e.g. platform initialisation drivers) that control DMA-capable devices will allocate memory outside of the protected region (otherwise the device will not work), the DPR protection is, in an example, disabled after SMRAM has been locked.

[0027] As such, there is a stronger assurance that SMRAM is what it should be, even in the event of malicious DMA device.

[0028] Subsequently, according to an example, the protection is extended to the entire BIOS, up to when the OS is ready to boot. In an example, the same DPR register can continue to be used to protect the BIOS. As such, drivers should allocate “normal” memory in the DMA-protected region, and drivers of DMA devices should allocate memory outside of that area.

[0029] In another example, a Driver Execution Environment (DXE) driver that enables VT-d in the BIOS can be provided. This can set up VT-d mappings so that the entire BIOS can be DMA-protected, and then disable the DPR protection. This enables hooking into existing Map() and Unmap() functions that DMA drivers call when doing DMA transactions, and setup specific VT-d mappings that would allow DMA just for these specific regions.

[0030] According to an example, for a handover to the OS, the OS can have the VT-d PMR (protected memory range) registers available, which enables the OS to retrieve full control of VT-d.

[0031] Figure 1 is a flowchart of a boot process according to an example. In block 101 a CPU of a system comes out of reset (e.g. the system is powered on). In block 103, DMA protection is activated in the DPR registers, and in block 105 DRAM is initialized. The processes in blocks 103 and 105 may be reversed. That is, DRAM can be initialized before DPR (or similar) has been used. In an example, for security, the integrity of something in DRAM (e.g. CPU instructions, critical data) should not be depended on when the DRAM is not protected. As an alternative, the following flow is valid: Initialize DRAM; Activate protections; Start relying on things in DRAM.

[0032] At this point, DRAM can be used normally (within the DMA-protected region), but DMA driver code and data is allocated in the DMA protected region, as a device is not supposed to touch those and a dedicated portion of non-protected memory can be allocated in order to enable communication with the device.

[0033] In block 107, the OS is started, and in block 109 the OS activates DMA protection in the VT-d "PMR" registers for portions of memory that the OS controls. The OS would set up VT-d pagetables there to protect only itself, as BIOS would still be under DPR protection. In an example, an OS can use VT-d pagetables instead of/in addition to PMR.

[0034] Figure 2 is a flowchart of a boot process according to an example. More particularly, figure 2 is a flowchart of a boot process using VT-d pagetables to minimize breaking changes according to an example. In block 201 a CPU of a system comes out of reset (e.g. the system is powered on). In block 203, DMA protection is activated in the DPR registers, and in block 205 DRAM is initialized. As before, at this point, DRAM can be used normally (within the DMA-protected region). In block 207, VT-d pagetables (in DMA-protected memory) are set up to ensure that the DRAM region containing the BIOS code and the VT-d mappings will be protected a second time and devices are isolated from each other, if desired. Once those mappings are set-up, the DPR registers that offered the early DMA-protection are deactivated in block 209.

[0035] In block 211 the OS is started, and in block 213 it activates DMA protection in the VT-d "PMR" registers for portions of memory that the OS controls. That is, the OS sets up VT-d pagetables there that protects memory regions containing runtime BIOS code, like ACPI tables and UEFI Runtime Services (SMRAM should not be a concern as it is always DMA-protected). If at that point the OS still relies on UEFI drivers doing DMA, the pagetables it sets up should take that into account.

[0036] Figure 3 is a flowchart of a method for secure hardware initialization during a start-up process according to an example. In block 301, a protected portion of a physical memory is activated. For example, as noted above, DMA protection can be activated in the DPR registers. In block 303, a part of the protected portion of the physical memory is allocated for execution of direct memory access drivers. In block 305, a first part of the physical memory, accessible by a device via a memory management tool, is allocated for data execution using the memory management tool. For example, the memory management tool can be the IOMMU or Vt-d controller.

[0037] Figure 4 is a schematic representation of a system according to an example. A portion of DRAM 401 of a device 403 is depicted. A protected (DPR) region 405 of the RAM 401 is depicted. In an example, as described above, BIOS and driver instructions can be executed in region 405. A memory

management tool (MMT) 407 can be used to allocate a first protected region 409 of the memory 401 that can only be accessed by the device 403 via the tool 407. An unprotected region 411 of memory 401 can remain for use by device 403. In an example, data (e.g. an executable) is received from device 403. It can then be copied to the protected region (405/409) where it can be authenticated. If authentic, the data can be executed.

[0038] In an example, as VT-d pagetables are dynamic, protected/unprotected portions can be reassigned at will. Accordingly, a device can DMA into unprotected memory, then protect this memory, and then authenticate it.

[0039] Figure 5 is a schematic representation of a boot process according to an example. During the time period when a device (e.g. device 403) BIOS is executing from power up 501, there are two processes that are provided, according to an example, to secure hardware initialization during the start-up process. In 503 of the BIOS process, DPR is used, and in 505 of the BIOS process, DPR/IOMMU pagetables are used up until 507 when there is handover to the OS.

[0040] Examples in the present disclosure can be provided as methods, systems or machine-readable instructions. Such machine-readable instructions may be included on a computer readable storage medium (including but not limited to disc storage, CD-ROM, optical storage, etc.) having computer readable program codes therein or thereon.

[0041] The present disclosure is described with reference to flow charts and/or block diagrams of the method, devices and systems according to examples of the present disclosure. Although the flow diagrams described above show a specific order of execution, the order of execution may differ from that which is depicted. Blocks described in relation to one flow chart may be combined with those of another flow chart. In some examples, some blocks of the flow diagrams may not be necessary and/or additional blocks may be added. It shall be understood that each flow and/or block in the flow charts and/or block diagrams, as well as combinations of the flows and/or diagrams in the flow

charts and/or block diagrams can be realized by machine readable instructions.

[0042] The machine-readable instructions may, for example, be executed by a general-purpose computer, a special purpose computer, an embedded processor or processors of other programmable data processing devices to realize the functions described in the description and diagrams. In particular, a processor or processing apparatus may execute the machine-readable instructions. Thus, modules of apparatus may be implemented by a processor executing machine readable instructions stored in a memory, or a processor operating in accordance with instructions embedded in logic circuitry. The term 'processor' is to be interpreted broadly to include a CPU, processing unit, ASIC, logic device, or programmable gate set etc. The methods and modules may all be performed by a single processor or divided amongst several processors.

[0043] Such machine-readable instructions may also be stored in a computer readable storage that can guide the computer or other programmable data processing devices to operate in a specific mode.

[0044] For example, the instructions may be provided on a non-transitory computer readable storage medium encoded with instructions, executable by a processor.

[0045] Figure 6 shows an example of a processor 150 associated with a memory 152. The memory 152 comprises computer readable instructions 154 which are executable by the processor 150. The instructions 154 can comprise instructions to activate a protected portion of a physical memory, instructions to allocate a part of the protected portion of the physical memory for execution of direct memory access drivers, and instructions to using a memory management tool, allocate a first part of the physical memory, accessible by a device via the memory management tool, for data execution.

[0046] Such machine-readable instructions may also be loaded onto a computer or other programmable data processing devices, so that the computer or other programmable data processing devices perform a series of operations to

produce computer-implemented processing, thus the instructions executed on the computer or other programmable devices provide a operation for realizing functions specified by flow(s) in the flow charts and/or block(s) in the block diagrams.

[0047] Further, the teachings herein may be implemented in the form of a computer software product, the computer software product being stored in a storage medium and comprising a plurality of instructions for making a computer device implement the methods recited in the examples of the present disclosure.

[0048] While the method, apparatus and related aspects have been described with reference to certain examples, various modifications, changes, omissions, and substitutions can be made without departing from the spirit of the present disclosure. In particular, a feature or block from one example may be combined with or substituted by a feature/block of another example.

[0049] The word "comprising" does not exclude the presence of elements other than those listed in a claim, "a" or "an" does not exclude a plurality, and a single processor or other unit may fulfil the functions of several units recited in the claims.

[0050] The features of any dependent claim may be combined with the features of any of the independent claims or other dependent claims.

CLAIMS

1. A method for secure hardware initialization during a start-up process, the method comprising:

activating a protected portion of a physical memory;

allocating a part of the protected portion of the physical memory for use by direct memory access, DMA, drivers and non-DMA related hardware initialization instructions; and

using a memory management tool, allocating a first part of the physical memory, accessible by a device via the memory management tool, for data.

2. A method as claimed in claim 1, further comprising:

one of: copying and moving at least a part of the data located in the first part of the physical memory into the protected portion.

3. A method as claimed in claim 1, further comprising:

authenticating the data located in the first part of the physical memory in the protected portion.

4. A method as claimed in claim 2, wherein the data is modified before, during or after moving to the protected portion.

5. A method as claimed in claim 1, further comprising:

providing a mapping to the first part of the physical memory within the protected portion of the physical memory whereby to enable the device to access, via the memory management tool, the first part.

6. A method as claimed in claim 1, further comprising:

executing an operating system in the protected part of the physical memory.

7. A method as claimed in claim 6, further comprising:

providing a DMA accessible portion of the operating system in the first part of the physical memory.

8. A system comprising a physical memory, the system to:

bootstrap into a secure starting position by securing the use of shared memory resources by activating a protected portion of the physical memory during a system start-up process;

assign a part of the protected portion of the physical memory for use by direct memory access, DMA, drivers and non-DMA related hardware initialization instructions, the system further comprising a memory management tool to:

allocate a first part of the physical memory, accessible by a device via the memory management tool, for use by data.

9. A system as claimed in claim 8, further comprising a processor to:

one of: copy and move at least a part of the data located in the first part of the physical memory into the protected portion.

10. A system as claimed in claim 9, the processor further to:

authenticate the data located in the first part of the physical memory in the protected portion.

11. A system as claimed in claim 8, the processor further to:

generate a mapping to a region of the first part of the physical memory; and

store the mapping within the protected portion.

12. A system as claimed in claim 11, the memory management tool further to:

control access to the region and the protected portion.

13. A non-transitory machine-readable storage medium encoded with instructions executable by a processor of a device for secure device hardware initialization during a start-up process, the machine-readable storage medium comprising instructions to:

allocate a part of a protected portion of the physical memory for use by direct memory access, DMA, drivers and non-DMA related hardware initialization instructions; and

using a memory management tool, allocate a first part of the physical memory, accessible by a device via the memory management tool, for data.

14. A non-transitory machine-readable storage medium as claimed in claim 13, further encoded with instructions to:

authenticate data executed in the first part of the physical memory in the protected portion.

15. A non-transitory machine-readable storage medium encoded as claimed in claim 13, further encoded with instructions to:

generate a mapping to the first part of the physical memory within the protected portion of the physical memory whereby to enable the device to access, via the memory management tool, the first part.

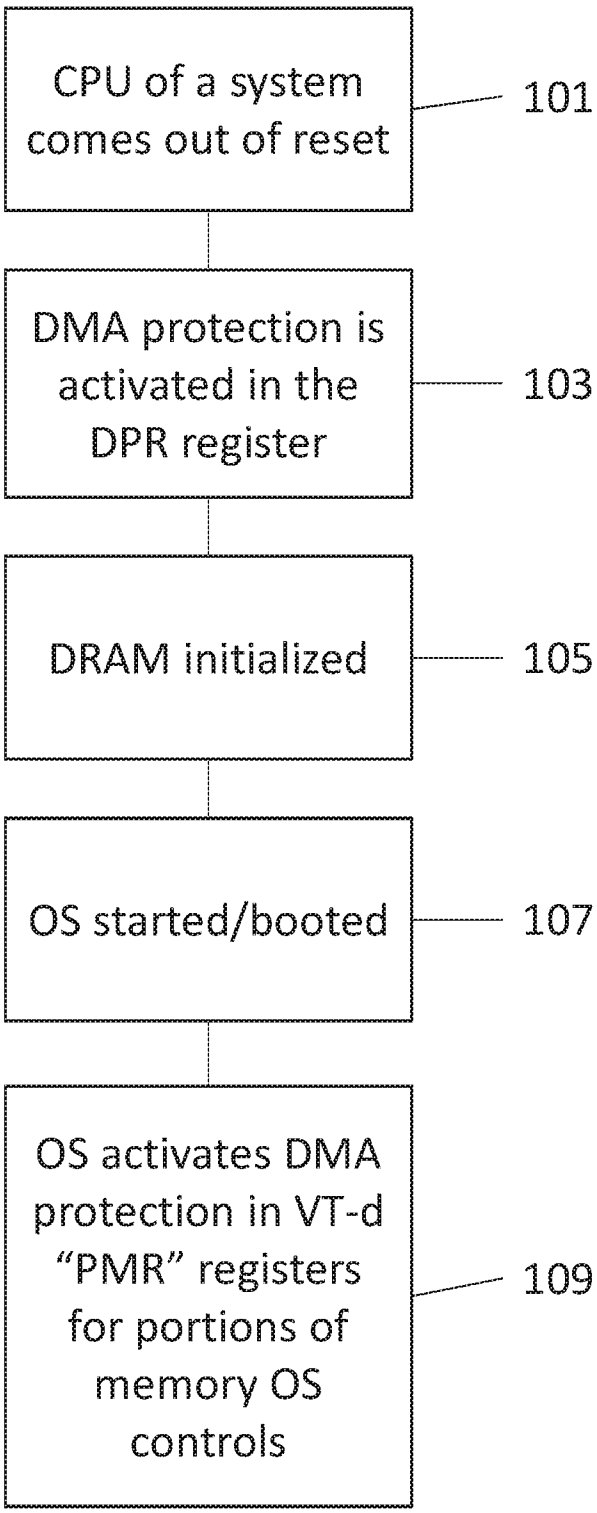
CPU of a system comes out of reset
101
DMA protection is activated in the DPR register
103
DRAM initialized
105
OS started/booted
107
OS activates DMA protection in VT-d "PMR" registers for portions of memory OS controls
109

Figure 1

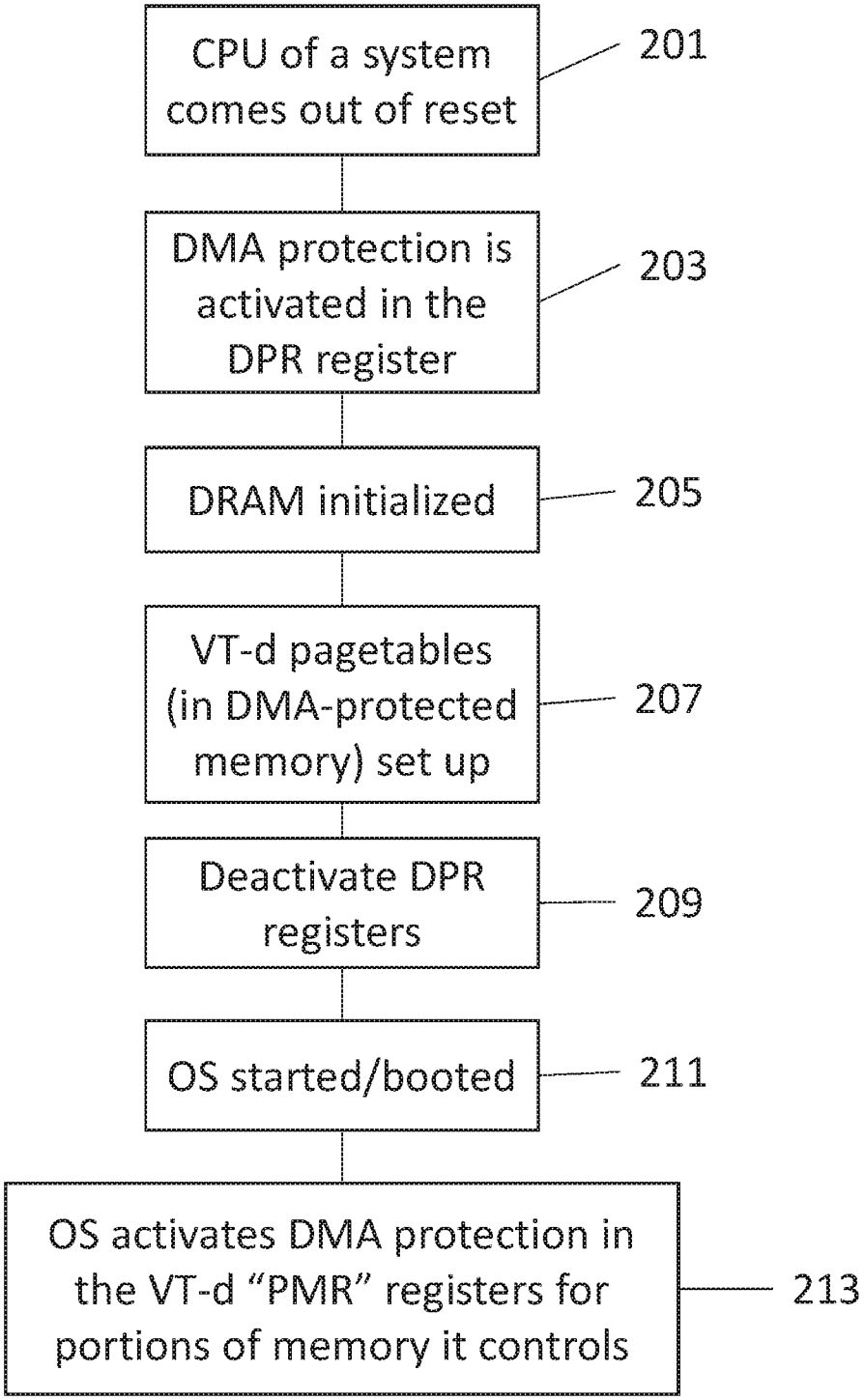
CPU of a system comes out of reset
201
DMA protection is activated in the DPR register
203
DRAM initialized
205
VT-d pagetables (in DMA-protected memory) set up
207
Deactivate DPR registers
209
OS started/booted
211
OS activates DMA protection in the VT-d "PMR" registers for portions of memory it controls
213

Figure 2

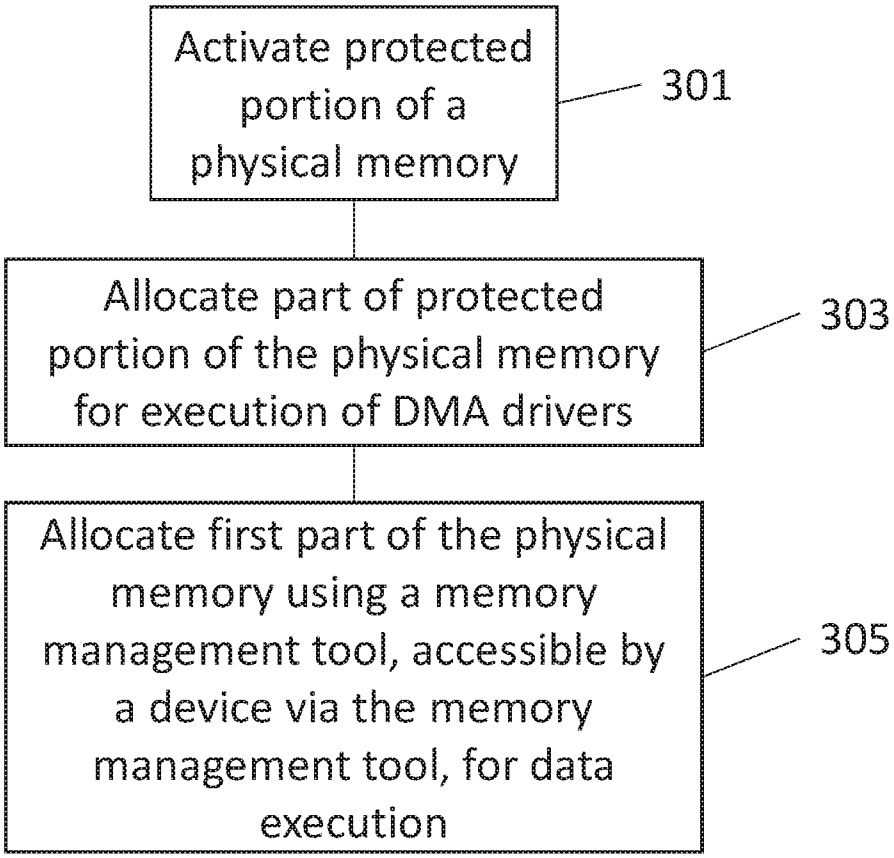
Activate protected portion of a physical memory
301
Allocate part of protected portion of the physical memory for execution of DMA drivers
303
Allocate first part of the physical memory using a memory management tool, accessible by a device via the memory management tool, for data execution
305

Figure 3

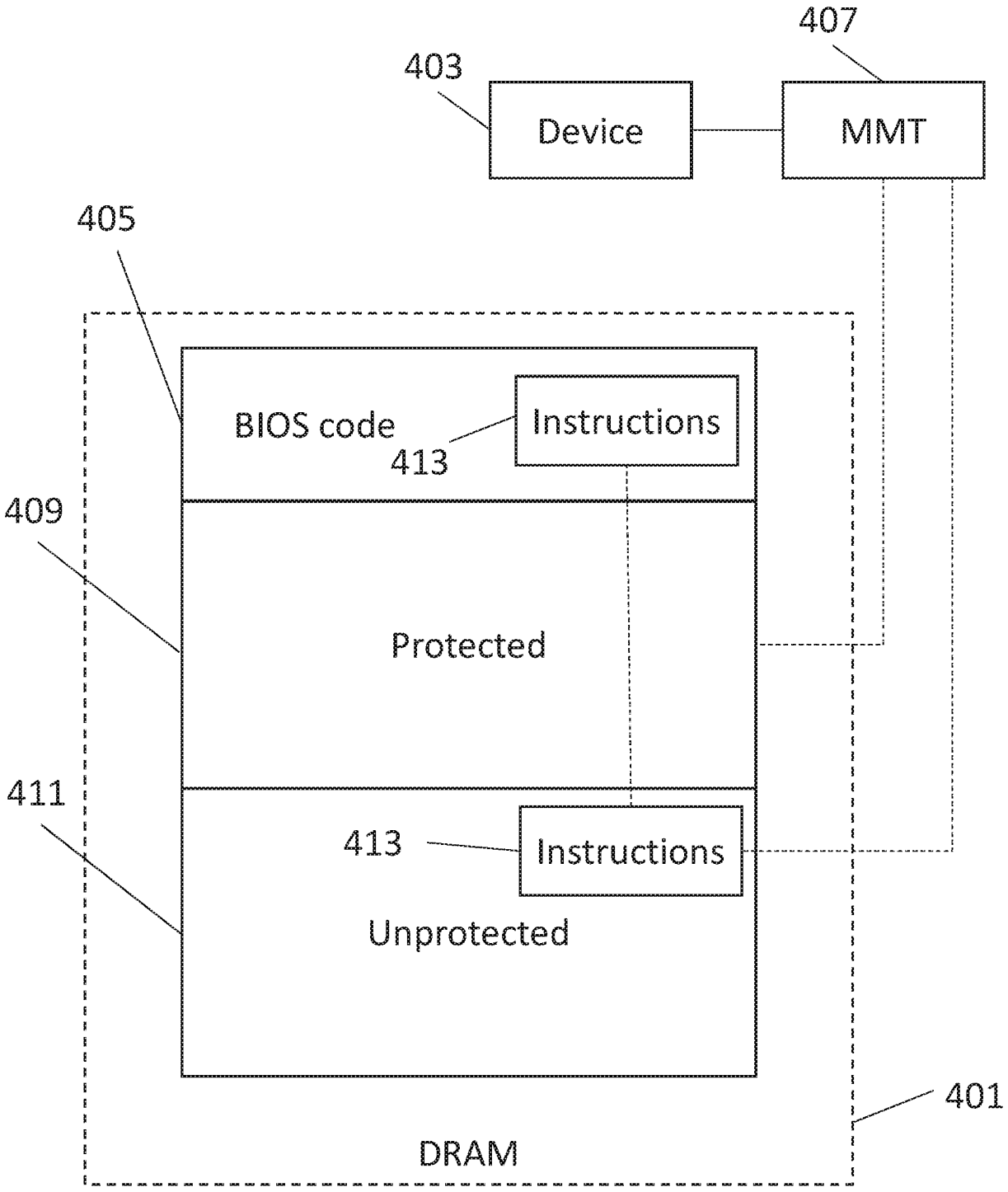
407
403
Device
MMT
405
BIOS code
Instructions
413
409
Protected
411
413
Instructions
Unprotected
401
DRAM

Figure 4

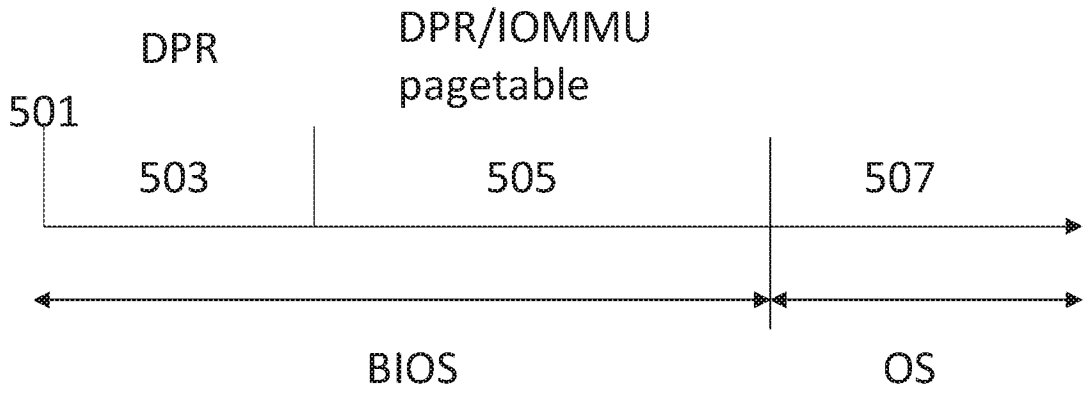

Figure 5

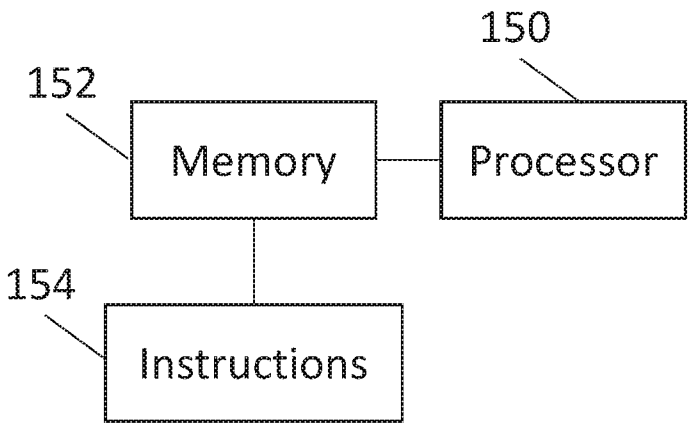

Figure 6

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US 2017/059070

A. CLASSIFICATION OF SUBJECT MATTER

G06F 12/14 *(2006.01)*
G06F 21/70 *(2013.01)*

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F 9/00, 9/44-9/445, G06F 12/00, 12/02, 12/14, 12/16, G06F 15/00, 15/16-15/177, 21/00, 21/60, 21/85, 21/70

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)

PatSearch (RUPTO internal), Esp@cenet, PAJ, USPTO, Information Retrieval System of FIPS

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2014/0173236 A1 (ADVANCED MICRO DEVICES, INC.) 19.06.2014, paragraphs [0001], [0032] - [0034], [0043], [0045] - [0050], [0052] - [0054], [0056] - [0057], [0065], claims 1-2	1, 2, 5-9, 11-13, 15
Y		3, 4, 10, 14
Y	US 2015/0089173 A1 (SIDDHARTHA CHHABRA et al.) 26.03.2015, paragraphs [0026], [0145]	3, 10, 14
Y	US 6131165 A (SUN MICROSYSTEMS, INC.) 10.10.2000, col. 1, line 44 - col. 2, line 2	4
A	EP 3086235 A1 (THALES) 26.10.2016	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 document 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
03 September 2018 (03.09.2018)	20 September 2018 (20.09.2018)

Name and mailing address of the ISA/RU:	Authorized officer
Federal Institute of Industrial Property, Berezhkovskaya nab., 30-1, Moscow, G-59, GSP-3, Russia, 125993 Facsimile No: (8-495) 531-63-18, (8-499) 243-33-37	S. Parusov Telephone No. (495)531-64-81