



(51) International Patent Classification:
G06F 9/455 (2006.01)

(21) International Application Number:
PCT/US2016/063379

(22) International Filing Date:
22 November 2016 (22.11.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
14/976,936 21 December 2015 (21.12.2015) US

(63) Related by continuation (CON) or continuation-in-part (CIP) to earlier application:
US 14/976,936 (CON)
Filed on 21 December 2015 (21.12.2015)

(71) Applicant: INTEL CORPORATION [US/US]; 2200 Mission College Blvd, Santa Clara, California 95054 (US).

(72) Inventors: SUN, Mingqiu; 15360 SW Kiwanda Lane, Beaverton, Oregon 97007 (US). ZIMMER, Vincent J.; 1838 Pointe Woodworth Dr NE, Tacoma, Washington 98422 (US). POORNACHANDRAN, Rajesh; 15317 NW Twoponds Drive, Portland, Oregon 97229 (US). SELVA-RAJE, Gopinath; 12658 NW Ally Elizabeth Ct, Portland, Oregon 97229 (US).

(74) Agents: SCHEER, Bradley, W. et al.; P O Box 2938, Minneapolis, Minnesota 55402 (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, JP, KE, KG, 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).

Published:

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

(54) Title: EMBEDDED ARCHITECTURE BASED ON PROCESS VIRTUAL MACHINE

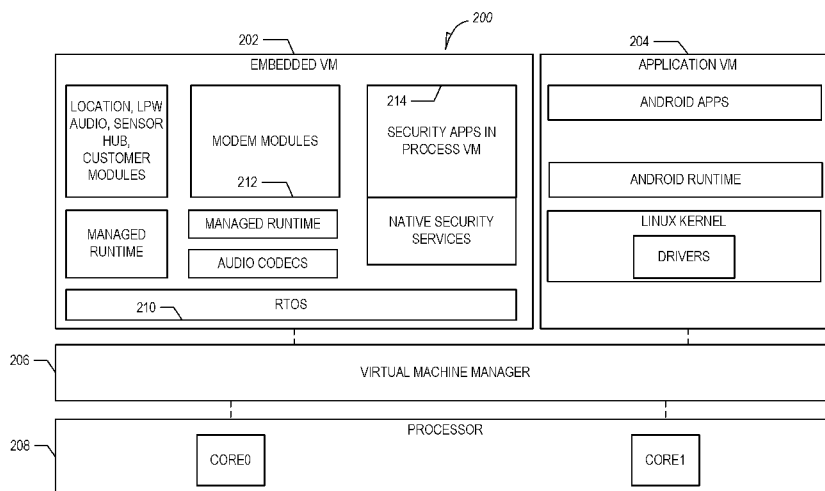


FIG. 2

(57) Abstract: A system on a chip (SoC) may comprise at least one processor with at least one core and a storage device comprising a first system, virtual machine configured to be executed on the at least one processor. The storage device may comprise a second system virtual machine configured to be executed by the at least one processor. The second system virtual machine may include at least one process virtual, machine; a modem, configured as one of the at least one process virtual machine; and a real-time operating system (RTOS) to schedule execution of the at least one process virtual machine on the at least one processor.

EMBEDDED ARCHITECTURE BASED ON PROCESS VIRTUAL MACHINE

5

CLAIM OF PRIORITY

[0001] This PCT application claims the priority benefit of U.S. Patent Application Serial No. 14/976,936 filed on December 21, 2015 which is incorporated herein by reference in its entirety.

10

TECHNICAL FIELD

[0002] Embodiments described herein generally relate to system architecture and in particular, but without limitation, to embedded architecture based on process virtual machine.

15

BACKGROUND

[0003] A system-on-a-chip, or SoC, is a single integrated circuit that includes the components (e.g., microprocessor, memory, input/output logic) for a particular system. An SoC may be useful when power usage is a concern such as in IoT devices and smartphones.

20

BRIEF DESCRIPTION OF THE DRAWINGS

[0004] In the drawings, which are not necessarily drawn to scale, like numerals may describe similar components in different views. Like numerals having different letter suffixes may represent different instances of similar components. Some embodiments are illustrated by way of example, and not limitation, in the figures of the accompanying drawings in which:

25

[0005] FIG. 1 illustrates a software architecture for a System on a Chip (SoC), according to various examples.

30

[0006] FIG. 2 illustrates a software architecture for an SoC, according to various examples.

[0007] FIG. 3 illustrates a schematic diagram of a modem module, according to various examples;

[0008] FIG. 4 illustrates a runtime for loading a modem module, according to various examples;

[0009] FIG. 5 illustrates a flowchart of a method of executing virtual machines on a system on a chip, according to various examples; and

- 5 [0010] FIG. 6 is a block diagram illustrating an example machine upon which any one or more of the techniques (e.g., methodologies) discussed herein may be performed, according to an example embodiment.

DETAILED DESCRIPTION

- 10 [0011] In various examples, a system-on-a-chip, or SoC, is a single integrated circuit that includes the components (e.g., microprocessor, memory, input/output logic) for a particular system. An SoC may be useful when power usage is a concern such in IoT devices and smartphones; however, SoCs are not limited to such uses and are increasingly used in higher-end computing devices.
- 15 For convenience, this disclosure generally discusses an SoC as it pertains to a mobile device with a cellular modem.

[0012] FIG. 1 illustrates a software architecture for an SoC, according to various examples. The software architecture includes multiple system virtual machines (VMs): modem VM 102, secure VM 104, and application VM 106.

- 20 The software architecture further includes virtual machine manager 108 and processor 110. The processor 110 may include more than one core in various examples. In an example, modem VM 102 may be used to transmit/receive data over one or more cellular antennas. The secure VM 104 may be used for security functions running on the SoC that are to be isolated from user applications
- 25 executing on application VM 106.

- [0013] A system VM may be a platform that allows execution of other processes (e.g., applications) via a guest operating system. The host and guest operating systems may be different or the same. For example, the host operating system may be Linux based, but the guest operating system may be Microsoft
- 30 Windows® variant. The guest operating system may provide a complete system for the execution of applications, etc., created for the guest operating system. In other words, a system VM may include drivers, libraries, etc., that would regularly exist if the guest operating system was the main operating system of a computing device.

[0014] An application running via a system VM would appear to have access to the processor(s) of the computing device but in actuality would be implemented as virtual processors—such as VCPU0 in modem VM 102. In some instances, a system VM uses resources of the host computing device regardless of whether an application is running on the system VM.

[0015] Manufacturers of SoCs are presented with a variety of challenges when potential customers want to add dedicated functionality beyond that provided in modem VM 102, secure VM 104, and application VM 106. For example, some customers may want a dedicated system VM for processing sensor input on a smartphone. Another dedicated system VM may be used to listen for audio at all times, while another system VM processes location information. The performance of an SoC may be negatively impacted when utilizing this many system VMs due to the overhead requirements of each system VM.

[0016] Scheduling becomes another challenge when using multiple system VMs (e.g., via virtual machine manager 108). Many VMs may create producer-consumer pair problems such as modem VM 102 receiving/sending data that application VM 106 is also processing. Because the same physical processors (e.g., processor 110) may be exposed to the same system VMs, both VMs may try to preempt the other VM leading to significant performance and user experience problems.

[0017] FIG. 2 illustrates a software architecture 200 for an SoC, in various examples. In contrast to FIG. 1, the software architecture 200 includes only two system VMs: embedded VM 202 and application VM 204. The embedded VM 202 may include at least one process VM such as modem modules 212 and security apps 214. The security apps may run on top of a process virtual machine, which may interface with native security services. In an example, a real-time operating system (RTOS) 210 is used as the guest operating system for embedded VM 202. The software architecture 200 also includes virtual machine manager 206.

[0018] Application VM 204 may include the operating system as used by an end user. For example, in application VM 204 a Linux kernel is included in addition to an Android Run Time. Android Apps may be executed using the Android Run Time. As in FIG. 1, the application VM 204 is separated from

security apps 214 and modem modules 212. In an example, different embedded process virtual machines may be implemented on a common managed runtime infrastructure to minimize system footprint.

[0019] In contrast to system VMs, a process VM may be executed via a host operating system and may act as a regular application would on the host operating system. However, the process VM may include the resources necessary to run a specific application and nothing more. In other words, the resource overhead for a process VM may be considerably less than a system VM, which includes an entire guest operating system.

10 [0020] A managed runtime environment may support the execution of a process VM via a host operating system. The runtime may abstract away many of the specifics of the underlying host operating system. In an example, the runtime environment handles many of the generic tasks that the developers used to have to anticipate and build in. For example, a managed runtime may handle heap management, security, class loading, garbage collection, and memory allocation, which may allow developers to concentrate on the business logic specific to their applications.

[0021] As seen, FIG. 2 illustrates a marked change from the software architecture of FIG. 1. Both modem and security apps were system level VMs in FIG. 1 but are process VMs in FIG. 2, (e.g., modem modules 212 and security apps 214). Furthermore, the other system VMs in FIG. 1 (e.g., sensor hub, not shown in FIG. 1) are process VMs within embedded VM 202.

[0022] Using the process VMs instead of system VMs helps to overcome a number of the challenges described above with respect to use of a software architecture as in FIG. 1. For example, the consumer-producer problem may be significantly reduced due to the number of system VMs matching the number of cores of processor 208 and each VM may be scheduled on a separate core. In contrast, there are many more system VMs than available cores for scheduling in FIG. 1, which may lead to performance issues as the VMs fight for access to the processing cores.

20 [0023] With reference back to FIG. 2, RTOS 210 may be used for scheduling the individual process VMs on processor 208. Furthermore, RTOS 210 may be used to isolate execution of a process VM from other process VMs.

[0024] Each process VM may be scheduled based on priority levels of a process VM. The priority level may be set by the RTOS or dynamically provisioned. For example, assuming that the SoC is part of a mobile phone, a software update may be installed that updates the priority level of a specific process VM. The process VMs may be dynamically provisioned overtime—
5 either at creation of the SoC or after the SoC is in the hands of an end user.

[0025] The priority level may be changed during execution of one or more VMs. For example, the priority level of a sensor hub process VM may begin low but if the sensor process VM has not been run for a certain period of time, the
10 priority level may increase. In another example, if a process VM is responsible for processing audio input, the priority level may be low if there is no detected noise, but increase when noise above a certain threshold is detected.

[0026] FIG. 3 illustrates a schematic diagram of a modem module 300, according to various examples. The modem module 300 may be transmitted to
15 update an existing modem module in modem modules 212 or create a new modem module. The modem module 300 includes modem Java classes 302, native modem module 304, and manifest and signature portion 306.

[0027] The modem Java classes 302 may be the code used by applications in order to access functionality of the modem. The native modem module 304
20 may be the actual communication primitives (e.g., transmit/receive) for the modem according to one or more communication standards.

[0028] In an example, the manifest and signature portion 306 includes attribution information. The manifest and signature portion 306 may include a hash of modem Java classes 302 and native modem module 304 encrypted with a
25 private key of the sender. The manifest and signature portion 306 may also include the public key of the sender.

[0029] FIG. 4 illustrates a runtime 400 for loading a modem module, according to an example embodiment. The runtime may be a process VM within embedded VM 202. The runtime 400 may include package loader 402, signature
30 verifier 404, class loader 406, native module loader 408, dynamic Java Native Interface (JNI) stub generator 410, execution engine 412, and API access control 414.

[0030] The package loader 402 may be used to parse modem module 300. Parsing may include accessing manifest and signature portion 306 to identify

what sections are supposed to be present in modem module 300. In some example, not all modem modules include both modem Java classes 302 and native modem module 304. There may be instances where only modem Java classes 302 or native modem module 304 is updated. If the manifest and
5 signature portion 306 indicates both sections are included, but package loader 402 only sees one section, the modem module may be rejected.

[0031] The signature verifier 404 may be used to verify the sender of modem 300. Verification may include using the included public key to decrypt the hash. Signature verifier 404 may independently compute a hash of the
10 included data (e.g., modem Java classes 302 and native modem module 304). If the computed hash matches the decrypted hash, the modem may be considered authentic. In an example, the public key is compared to a trusted repository of keys either on the device the SoC is part of or retrieved from an external source.

[0032] The class loader 406 and native module loader 408 may be used to
15 update—or install if no modem is already present—the classes and communication primitives using execution engine 412. Updating/installing may include modifying data stored on a storage device communicatively coupled to the SoC or on the SoC itself.

[0033] The manifest and signature portion 306 may indicate the type of
20 modem represented by modem Java classes 302 and native modem module 304. The class loader 406 and native module loader 408 may use this information to determine if a modem already exists within modem modules 212. For example, embedded VM 202 may include a datafile of installed process VMs and version information if available. Installing may include updating/adding resources files
25 contained with modem modules 212 specific to the modem identified in modem modules 300. API access control 414 may be used to set limits on which processes/application have access to issue calls to control the modem.

[0034] The manifest may also include what versions of modem Java
30 classes 302 and native modem module 304 are included. Accordingly, class loader 406 may examine the current version numbers (if any) of classes in a matching modem module to determine if an update is warranted. Similarly, native module loader 408 may determine if the included communication primitives are newer than what is currently in modem modules 212.

[0035] In an example, dynamic JNI stub generator 410 is used to generate code stubs for use by a native operating system of embedded VM 202 with respect to the modem.

[0036] Although updating/installing modems was discussed in FIG. 3 and FIG. 4, software architecture of FIG. 2 is not so limited. For example, a new feature (e.g., a low power mode audio listening mode) of a mobile phone may be loaded after the SoC was produced. To add the new feature, a datafile may be transmitted to the hosing device. The datafile may include the code to run via a runtime in embedded VM 202. The datafile may also include a suggested priority level for the feature.

[0037] As with the modem module 300, a manifest and signature section may be used to parse the datafile and determine its authenticity. Also, before installing the feature, a check may be made to see if a process VM already exists for the feature.

[0038] FIG. 5 illustrates a flowchart of a method of executing virtual machines on a system on a chip, according to various examples. The method may be performed by the SoC described herein. At operation 502, a first system virtual machine is executed on a first processing core of the system on a chip. At operation 504, in an example, a user downloaded application is executed on the first processing core of the least one processor. For example, a the SoC may be part of a mobile phone. The user may download an application from an app store. The application may be executed using the first virtual machine using a runtime configured to execute the application on the first processing core.

[0039] At operation 506, in an example, a second system virtual machine is executed on a second core of at least one processor. The second system virtual machine may include at least one process virtual machine. In an example, a modem is configure as one of the process virtual machines. The SoC may be receive a datafile including an update to the modem. The datafile may also include a signature portion. The second system virtual machine may verify the authenticity of the datafile based on the signature portion.

[0040] The datafile may be part of an update to a mobile device that includes that SoC. The update may be transmitted to the mobile device from a carrier of the mobile device. The datafile may include data representing

communication primitives (e.g., transmit/receive) for the modem. The modem may be updated based on the data.

[0041] In an example, a datafile may be received representing a feature of a mobile device. It may be determined that the feature is not already configured on the second system virtual machine a new process virtual machine may be established for the feature. Establishing may include configuring a managed runtime to execute code included in the datafile.

[0042] In an example, the datafile includes an update to an existing feature. The process virtual machine representing the feature may be updated based on data included in the datafile.

[0043] At operation 508, execution of the at least one process virtual machine on the second core of the at least one processor is scheduled using a real-time operating system of the second virtual machine. The scheduling may be based on priority levels of individual process virtual machines.

EXAMPLE COMPUTER SYSTEM

[0044] Embodiments described herein may be implemented in one or a combination of hardware, firmware, and software. Embodiments may also be implemented as instructions stored on a machine-readable storage device, which may be read and executed by at least one processor to perform the operations described herein. A machine-readable storage device may include any non-transitory mechanism for storing information in a form readable by a machine (e.g., a computer). For example, a machine-readable storage device may include read-only memory (ROM), random-access memory (RAM), magnetic disk storage media, optical storage media, flash-memory devices, and other storage devices and media.

[0045] Examples, as described herein, may include, or may operate on, logic or a number of components, modules, or mechanisms. Modules may be hardware, software, or firmware communicatively coupled to one or more processors in order to carry out the operations described herein. Modules may hardware modules, and as such modules may be considered tangible entities capable of performing specified operations and may be configured or arranged in a certain manner. In an example, circuits may be arranged (e.g., internally or with respect to external entities such as other circuits) in a specified manner as a

module. In an example, the whole or part of one or more computer systems (e.g., a standalone, client or server computer system) or one or more hardware processors may be configured by firmware or software (e.g., instructions, an application portion, or an application) as a module that operates to perform
5 specified operations. In an example, the software may reside on a machine-readable medium. In an example, the software, when executed by the underlying hardware of the module, causes the hardware to perform the specified operations. Accordingly, the term hardware module is understood to encompass a tangible entity, be that an entity that is physically constructed, specifically
10 configured (e.g., hardwired), or temporarily (e.g., transitorily) configured (e.g., programmed) to operate in a specified manner or to perform part or all of any operation described herein. Considering examples in which modules are temporarily configured, each of the modules need not be instantiated at any one moment in time. For example, where the modules comprise a general-purpose
15 hardware processor configured using software; the general-purpose hardware processor may be configured as respective different modules at different times. Software may accordingly configure a hardware processor, for example, to constitute a particular module at one instance of time and to constitute a different module at a different instance of time. Modules may also be software or
20 firmware modules, which operate to perform the methodologies described herein.

[0046] FIG. 6 is a block diagram illustrating a machine in the example form of a computer system 600, within which a set or sequence of instructions may be executed to cause the machine to perform any one of the methodologies
25 discussed herein, according to an example embodiment. In alternative embodiments, the machine operates as a standalone device or may be connected (e.g., networked) to other machines. In a networked deployment, the machine may operate in the capacity of either a server or a client machine in server-client network environments, or it may act as a peer machine in peer-to-peer (or
30 distributed) network environments. The machine may be an onboard vehicle system, wearable device, personal computer (PC), a tablet PC, a hybrid tablet, a personal digital assistant (PDA), a mobile telephone, or any machine capable of executing 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. Similarly, the term “processor-based system” shall be taken to include any set of one or more machines that are controlled by or operated by a processor (e.g., a computer) to individually or jointly execute instructions to perform any one or more of the methodologies discussed herein.

[0047] Example computer system 600 includes at least one processor 602 (e.g., a central processing unit (CPU), a graphics processing unit (GPU) or both, processor cores, compute nodes, etc.), a main memory 604 and a static memory 606, which communicate with each other via a link 608 (e.g., bus). The computer system 600 may further include a video display unit 610, an alphanumeric input device 612 (e.g., a keyboard), and a user interface (UI) navigation device 614 (e.g., a mouse). In one embodiment, the video display unit 610, input device 612 and UI navigation device 614 are incorporated into a touch screen display. The computer system 600 may additionally include a storage device 616 (e.g., a drive unit), a signal generation device 618 (e.g., a speaker), a network interface device 620, and one or more sensors (not shown), such as a global positioning system (GPS) sensor, compass, accelerometer, or other sensor.

[0048] The storage device 616 includes a machine-readable medium 622 on which is stored one or more sets of data structures and instructions 624 (e.g., software) embodying or utilized by any one or more of the methodologies or functions described herein. The instructions 624 may also reside, completely or at least partially, within the main memory 604, static memory 606, and/or within the processor 602 during execution thereof by the computer system 600, with the main memory 604, static memory 606, and the processor 602 also constituting machine-readable media.

[0049] While the machine-readable medium 622 is illustrated in an example embodiment to be a single medium, the term “machine-readable medium” may 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 instructions 624. The term “machine-readable medium” shall also be taken to include any tangible medium that is capable of storing, encoding or carrying

instructions for execution by the machine and that cause the machine to perform any one or more of the methodologies of the present disclosure or that is capable of storing, encoding or carrying data structures utilized by or associated with such instructions. The term “machine-readable medium” shall accordingly be
5 taken to include, but not be limited to, solid-state memories, and optical and magnetic media. Specific examples of machine-readable media include non-volatile memory, including but not limited to, by way of example, semiconductor memory devices (e.g., electrically programmable read-only memory (EPROM), electrically erasable programmable read-only memory
10 (EEPROM)) and flash memory devices; magnetic disks such as internal hard disks and removable disks; magneto-optical disks; and CD-ROM and DVD-ROM disks.

[0050] The instructions 624 may further be transmitted or received over a communications network 626 using a transmission medium via the network
15 interface device 620 utilizing any one of a number of well-known transfer protocols (e.g., HTTP). Examples of communication networks include a local area network (LAN), a wide area network (WAN), the Internet, mobile telephone networks, plain old telephone (POTS) networks, and wireless data networks (e.g., Wi-Fi, 6G, and 4G LTE/LTE-A or WiMAX networks). The term
20 “transmission medium” shall be taken to include any intangible medium that is capable of storing, encoding, or carrying instructions for execution by the machine, and includes digital or analog communications signals or other intangible medium to facilitate communication of such software.

[0051] Example 1 is a system on a chip (SoC) for executing virtual
25 machines, the SoC comprising: at least one processor with at least one core; and a storage device comprising a first system virtual machine configured to be executed on the at least one processor; wherein the storage device further comprises a second system virtual machine configured to be executed by the at least one processor, the second system virtual machine including: at least one
30 process virtual machine; a modem configured as one of the at least one process virtual machine; and a real-time operating system (RTOS) to schedule execution of the at least one process virtual machine on the at least one processor.

[0052] In Example 2, the subject matter of Example 1 optionally includes, wherein the at least one processor is configured, when executing the second

system virtual machine, to: receive a datafile including an update to the modem; and update communication primitives of the modem based on data received in the datafile.

[0053] In Example 3, the subject matter of Example 2 optionally includes,
5 wherein the datafile comprises a signature portion.

[0054] In Example 4, the subject matter of Example 3 optionally includes, wherein the at least one processor is configured, when executing the second system virtual machine, to: verify the authenticity of the datafile based on the signature portion.

10 [0055] In Example 5, the subject matter of any one or more of Examples 1–4 optionally include, wherein the at least one processor is configured, when executing the second system virtual machine, to: receive a datafile representing a feature of a mobile device; determine that the feature is not already configured on the second system virtual machine; and establish a new process virtual
15 machine for the feature.

[0056] In Example 6, the subject matter of any one or more of Examples 1–5 optionally include, wherein the at least one processor is configured, when executing the second system virtual machine, to: receive a datafile representing an update to a feature of a mobile device; determine that the feature is
20 configured on the second system virtual machine as one of the at least one process virtual machines; and update the process virtual machine according to the update.

[0057] In Example 7, the subject matter of any one or more of Examples 1–6 optionally include, wherein the RTOS schedules execution of the at least
25 one process virtual machine based on priority levels of respective process virtual machines of the at least one process virtual machine.

[0058] In Example 8, the subject matter of any one or more of Examples 1–7 optionally include, wherein the first system virtual machine further includes: a runtime configured to execute applications on behalf of a user.

30 [0059] Example 9 is a method of executing virtual machines on a system on a chip, the method comprising: executing a first system virtual machine on a first processing core of the system on a chip; executing a user downloaded application on the first processing core of the least one processor; executing a second system virtual machine on a second core of at least one processor,

wherein the second system virtual machine includes at least one process virtual machine; and scheduling execution of the at least one process virtual machine on the second core of the at least one processor using a real-time operating system of the second virtual machine.

- 5 [0060] In Example 10, the subject matter of Example 9 optionally includes, wherein the at least one process virtual machine includes a modem configured as a process virtual machine.

 [0061] In Example 11, the subject matter of Example 10 optionally includes, wherein executing the second system virtual machine on a second core
10 of at least one processor includes: receiving a datafile including an update to the modem; and updating communication primitives of the modem based on data received in the datafile.

 [0062] In Example 12, the subject matter of Example 11 optionally includes, wherein the datafile comprises a signature portion.

- 15 [0063] In Example 13, the subject matter of Example 12 optionally includes, wherein executing the second system virtual machine on a second core of at least one processor includes: verifying the authenticity of the datafile based on the signature portion.

 [0064] In Example 14, the subject matter of any one or more of Examples
20 9–13 optionally include, wherein executing the second system virtual machine on a second core of at least one processor includes: receiving a datafile representing a feature of a mobile device; determining that the feature is not already configured on the second system virtual machine; and establishing a new process virtual machine for the feature.

- 25 [0065] In Example 15, the subject matter of any one or more of Examples 9–14 optionally include, wherein executing the second system virtual machine on a second core of at least one processor includes: receiving a datafile representing an update to a feature of a mobile device; determining that the feature is configured on the second system virtual machine as one of the at least
30 one process virtual machines; and updating the process virtual machine according to the update.

 [0066] In Example 16, the subject matter of any one or more of Examples 9–15 optionally include, wherein the RTOS schedules execution of process

virtual machines of the at least one process virtual machine based on priority levels of the respective process virtual machines.

[0067] In Example 17, the subject matter of any one or more of Examples 9–16 optionally include, wherein the first system virtual machine further
5 includes a runtime configured to execute applications on behalf of a user.

[0068] Example 18 is at least one machine-readable medium including instructions, which when executed by a machine, cause the machine to perform operations of any of the methods of Examples 10-17.

[0069] Example 19 is an apparatus comprising means for performing any
10 of the methods of Examples 10-17.

[0070] Example 20 is a apparatus for executing virtual machines on a system on a chip, the apparatus comprising: means for executing a first system virtual machine on a first processing core of the system on a chip; means for executing a user downloaded application on the first processing core of the least
15 one processor; means for executing a second system virtual machine on a second core of at least one processor, wherein the second system virtual machine includes at least one process virtual machine; and means for scheduling execution of the at least one process virtual machine on the second core of the at least one processor using a real-time operating system of the second virtual
20 machine.

[0071] In Example 21, the subject matter of Example 20 optionally includes, wherein the at least one process virtual machine includes a modem configured as a process virtual machine.

[0072] In Example 22, the subject matter of Example 21 optionally
25 includes, wherein the means for executing the second system virtual machine on a second core of at least one processor includes: means for receiving a datafile including an update to the modem; and means for updating communication primitives of the modem based on data received in the datafile.

[0073] In Example 23, the subject matter of Example 22 optionally
30 includes, wherein the datafile comprises a signature portion.

[0074] In Example 24, the subject matter of Example 23 optionally includes, wherein the means for executing the second system virtual machine on a second core of at least one processor includes: means for verifying the authenticity of the datafile based on the signature portion.

[0075] In Example 25, the subject matter of any one or more of Examples 20–24 optionally include, wherein the means for executing the second system virtual machine on a second core of at least one processor includes: means for receiving a datafile representing a feature of a mobile device; means for
5 determining that the feature is not already configured on the second system virtual machine; and means for establishing a new process virtual machine for the feature.

[0076] In Example 26, the subject matter of any one or more of Examples 20–25 optionally include, wherein the means for executing the second system
10 virtual machine on a second core of at least one processor includes: means for receiving a datafile representing an update to a feature of a mobile device; means for determining that the feature is configured on the second system virtual machine as one of the at least one process virtual machines; and means for updating the process virtual machine according to the update.

15 [0077] In Example 27, the subject matter of any one or more of Examples 20–26 optionally include, wherein the RTOS schedules execution of process virtual machines of the at least one process virtual machine based on priority levels of the respective process virtual machines.

[0078] In Example 28, the subject matter of any one or more of Examples
20 20–27 optionally include, wherein the first system virtual machine further includes a runtime configured to execute applications on behalf of a user.

[0079] The above detailed description includes references to the accompanying drawings, which form a part of the detailed description. The drawings show, by way of illustration, specific embodiments that may be
25 practiced. These embodiments are also referred to herein as “examples.” Such examples may include elements in addition to those shown or described. However, also contemplated are examples that include the elements shown or described. Moreover, also contemplate are examples using any combination or permutation of those elements shown or described (or one or more aspects
30 thereof), either with respect to a particular example (or one or more aspects thereof), or with respect to other examples (or one or more aspects thereof) shown or described herein.

CLAIMS

1. A system on a chip (SoC) for executing virtual machines, the SoC comprising:
- 5 at least one processor with at least one core; and
a storage device comprising a first system virtual machine configured to be executed on the at least one processor;
- wherein the storage device further comprises a second system virtual machine configured to be executed by the at least one processor, the second
- 10 system virtual machine including:
- at least one process virtual machine;
- a modem configured as one of the at least one process virtual machine; and
- a real-time operating system (RTOS) to schedule execution of
- 15 the at least one process virtual machine on the at least one processor.
2. The SoC of claim 1, wherein the at least one processor is configured, when executing the second system virtual machine, to:
- receive a datafile including an update to the modem; and
- 20 update communication primitives of the modem based on data received in the datafile.
3. The SoC of claim 2, wherein the datafile comprises a signature portion.
- 25
4. The SoC of claim 3, wherein the at least one processor is configured, when executing the second system virtual machine, to:
- verify the authenticity of the datafile based on the signature portion.
- 30 5. The SoC of claim 1, wherein the at least one processor is configured, when executing the second system virtual machine, to:
- receive a datafile representing a feature of a mobile device;
- determine that the feature is not already configured on the second system virtual machine; and

establish a new process virtual machine for the feature.

6. The SoC of claim 1, wherein the at least one processor is configured, when executing the second system virtual machine, to:

5 receive a datafile representing an update to a feature of a mobile device;

determine that the feature is configured on the second system virtual machine as one of the at least one process virtual machines; and

update the process virtual machine according to the update.

10

7. The SoC of claim 1, wherein the RTOS schedules execution of the at least one process virtual machine based on priority levels of respective process virtual machines of the at least one process virtual machine.

15 8. The SoC of claim 1, wherein the first system virtual machine further includes:

a runtime configured to execute applications on behalf of a user.

9. A method of executing virtual machines on a system on a chip, the
20 method comprising:

executing a first system virtual machine on a first processing core of the system on a chip;

executing a user downloaded application on the first processing core of the at least one processor;

25 executing a second system virtual machine on a second core of at least one processor, wherein the second system virtual machine includes at least one process virtual machine; and

scheduling execution of the at least one process virtual machine on the second core of the at least one processor using a real-time operating system of
30 the second virtual machine.

10. The method of claim 9, wherein the at least one process virtual machine includes a modem configured as a process virtual machine.

11. The method of claim 10, wherein executing the second system virtual machine on a second core of at least one processor includes:
receiving a datafile including an update to the modem; and
updating communication primitives of the modem based on data
5 received in the datafile.

12. The method of claim 11, wherein the datafile comprises a signature portion.

10 13. The method of claim 12, wherein executing the second system virtual machine on a second core of at least one processor includes:
verifying the authenticity of the datafile based on the signature portion.

15 14. The method of claim 9, wherein executing the second system virtual machine on a second core of at least one processor includes:
receiving a datafile representing a feature of a mobile device;
determining that the feature is not already configured on the second system virtual machine; and
20 establishing a new process virtual machine for the feature.

15. The method of claim 9, wherein executing the second system virtual machine on a second core of at least one processor includes:
receiving a datafile representing an update to a feature of a mobile
25 device;
determining that the feature is configured on the second system virtual machine as one of the at least one process virtual machines; and
updating the process virtual machine according to the update.

30 16. The method of claim 9, wherein the RTOS schedules execution of process virtual machines of the at least one process virtual machine based on priority levels of the respective process virtual machines.

17. The method of claim 9, wherein the first system virtual machine further includes a runtime configured to execute applications on behalf of a user.
- 5 18. At least one machine-readable medium including instructions, which when executed by a machine, cause the machine to perform operations of any of the methods of claims 10-17.
- 10 19. An apparatus comprising means for performing any of the methods of claims 10-17.

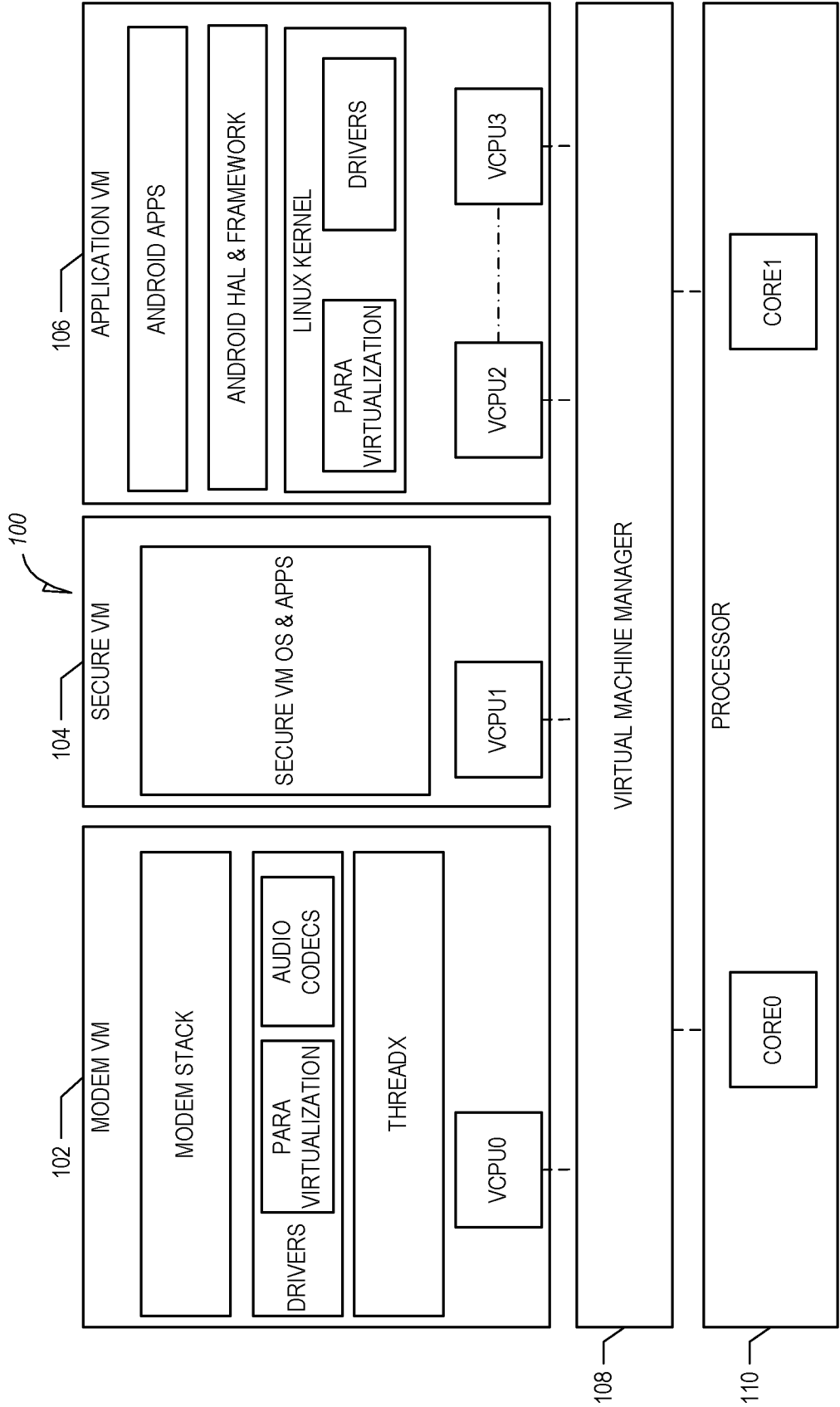


FIG. 1

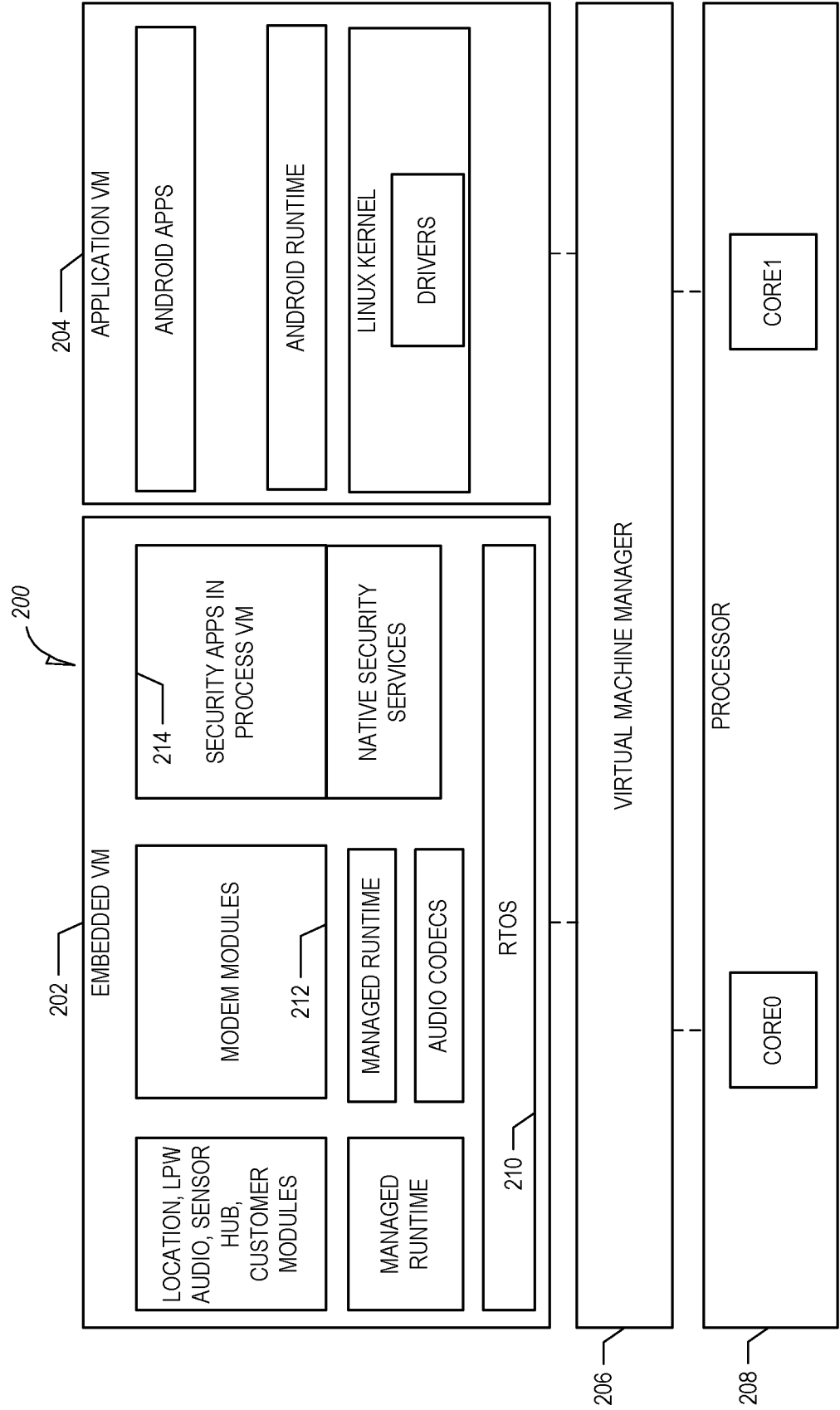


FIG. 2

300

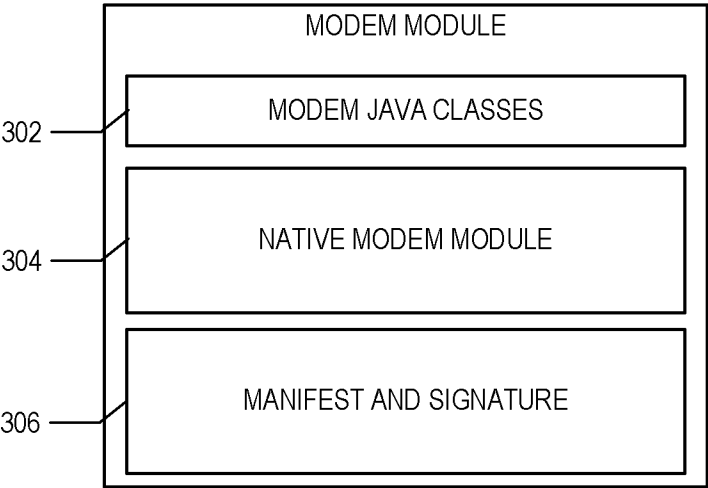


FIG. 3

4/6

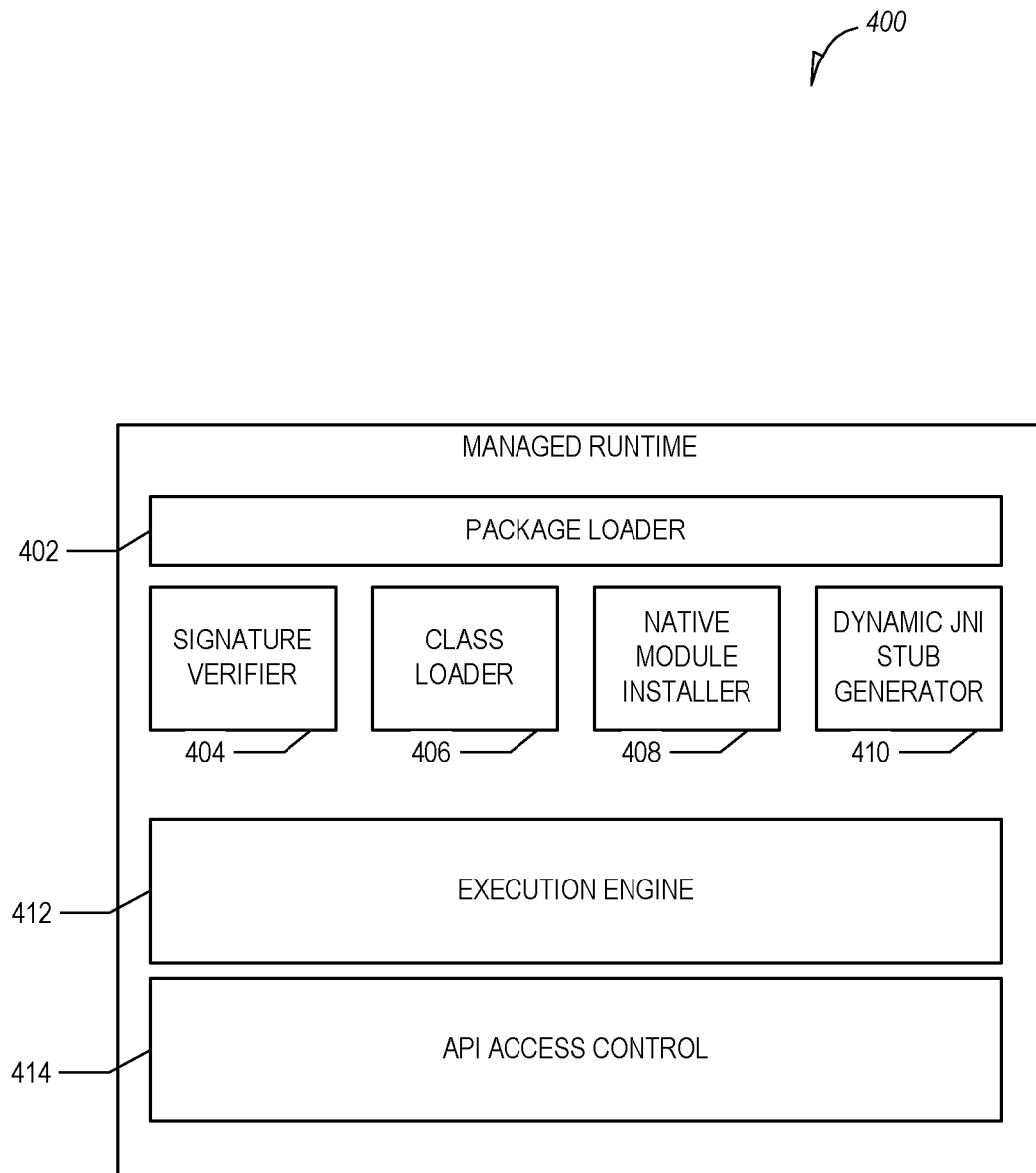


FIG. 4

5/6

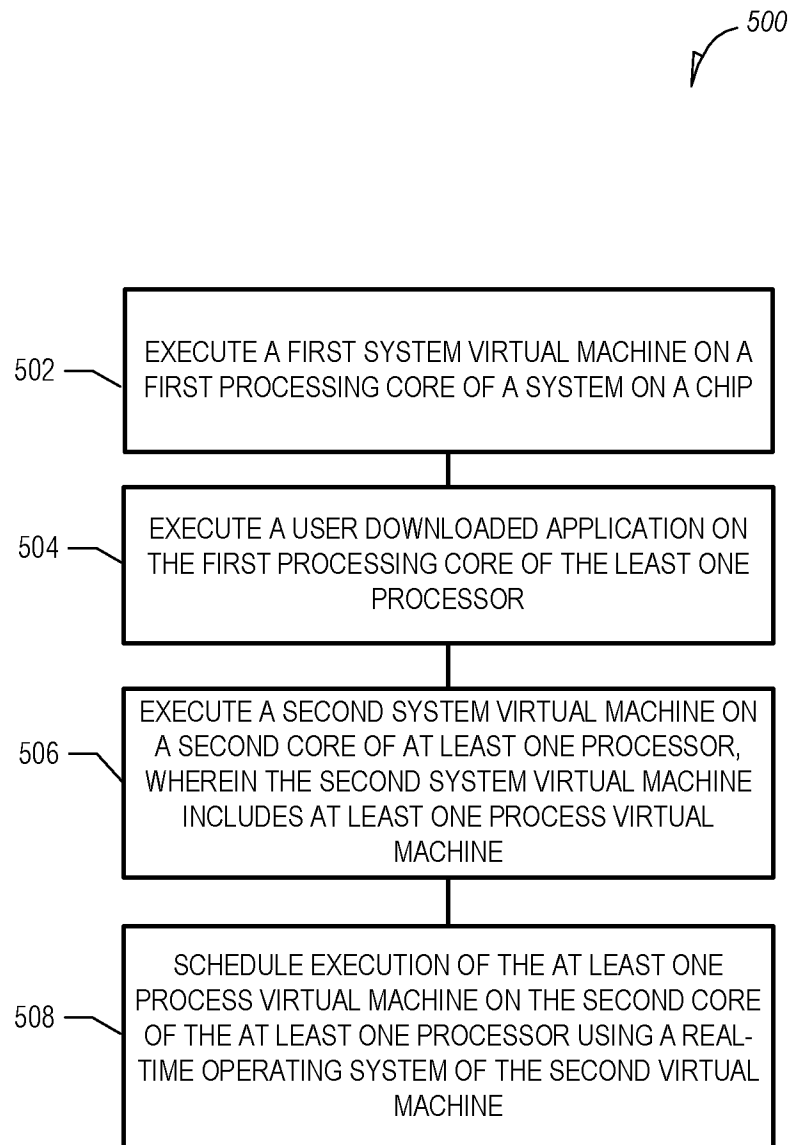


FIG. 5

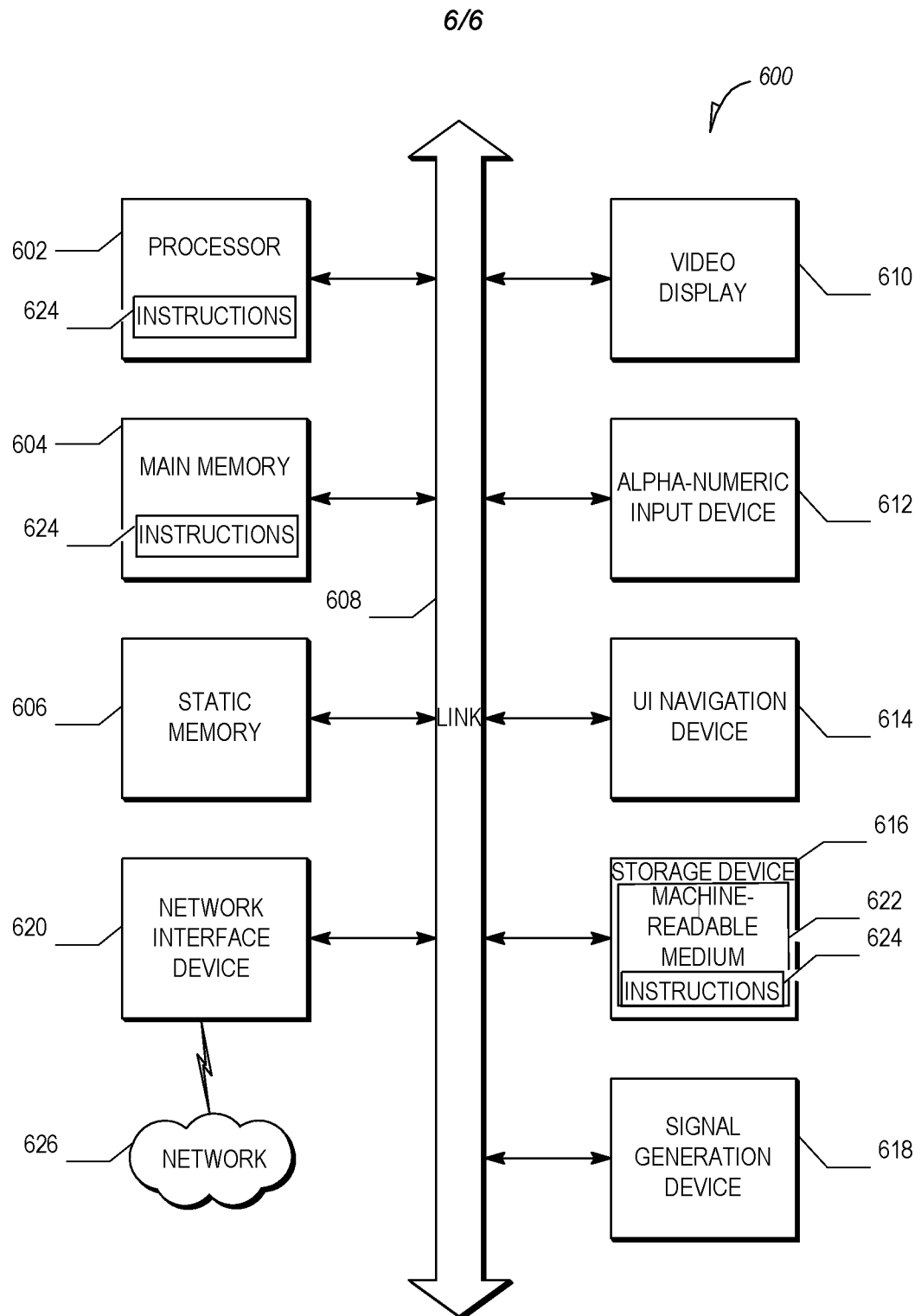


FIG. 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2016/063379**A. CLASSIFICATION OF SUBJECT MATTER****G06F 9/455(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 9/455; G06F 15/16; G06F 9/44

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: process virtual machine, real time operating system (RTOS), schedule, modem, update

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2014-0282507 A1 (QUALCOMM INCORPORATED) 18 September 2014 See paragraphs [0024]-[0044]; claim 1; and figures 1-3.	1, 7-10, 16-19
Y		2-6, 11-15
Y	US 2012-0159469 A1 (DOR LAOR) 21 June 2012 See paragraphs [0029]-[0040]; claims 1-4; and figures 2-4.	2-6, 11-15
A	US 2013-0125117 A1 (SIEMENS AKTIENGESELLSCHAFT) 16 May 2013 See paragraphs [0022]-[0027]; claim 1; and figures 1-3.	1-19
A	US 2014-0089925 A1 (NETAPP, INC.) 27 March 2014 See paragraphs [0018]-[0032]; claims 1, 3; and figures 1-3.	1-19
A	US 2006-0218536 A1 (VIATCHESLAV KIRILLINE et al.) 28 September 2006 See paragraphs [0036]-[0039], [0090]; and figures 5, 12.	1-19



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

10 March 2017 (10.03.2017)

Date of mailing of the international search report

10 March 2017 (10.03.2017)

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/US2016/063379

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2014-0282507 A1	18/09/2014	CN 105051690 A EP 2972851 A1 JP 2016-511489 A KR 10-2015-0132218 A TW 201447766 A TW I552076 B US 2016-299780 A1 US 9396012 B2 WO 2014-159444 A1	11/11/2015 20/01/2016 14/04/2016 25/11/2015 16/12/2014 01/10/2016 13/10/2016 19/07/2016 02/10/2014
US 2012-0159469 A1	21/06/2012	None	
US 2013-0125117 A1	16/05/2013	CN 103197949 A CN 103197949 B EP 2575039 A1 EP 2575039 B1	10/07/2013 24/08/2016 03/04/2013 14/12/2016
US 2014-0089925 A1	27/03/2014	US 2010-0058335 A1 US 8627312 B2 US 9141417 B2	04/03/2010 07/01/2014 22/09/2015
US 2006-0218536 A1	28/09/2006	US 7895594 B2	22/02/2011