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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
28 June 2012 (28.06.2012)

WIPO | PCT

(10) International Publication Number
WO 2012/087533 A1

(51) International Patent Classification:
G06F 1/32 (2006.01) *G06F 9/50* (2006.01)

(21) International Application Number:
PCT/US2011/062934

(22) International Filing Date:
1 December 2011 (01.12.2011)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/425,677 21 December 2010 (21.12.2010) US
61/544,087 6 October 2011 (06.10.2011) US
13/291,784 8 November 2011 (08.11.2011) US

(71) Applicant *(for all designated States except US)*: **QUALCOMM INCORPORATED** [US/US]; Attn: International Ip Administration, 5775 Morehouse Drive, San Diego, California 92121 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **STUBBS, Joshua, H.** [US/US]; 5775 Morehouse Drive, San Diego, California 92121 (US). **FRANTZ, Andrew, J.** [US/US]; 5775 Morehouse Drive, San Diego, California 92121 (US). **GARGASH, Norman, S.** [US/US]; 5775 Morehouse Drive, San Diego, California 92121 (US).

(74) Agent: **COLE, Nicholas, Albert**; 5775 Morehouse Drive, San Diego, California 92121 (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, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, 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,

[Continued on next page]

(54) Title: MINIMIZING RESOURCE LATENCY BETWEEN PROCESSOR APPLICATION STATES IN A PORTABLE COMPUTING DEVICE BY USING A NEXT-ACTIVE STATE SET

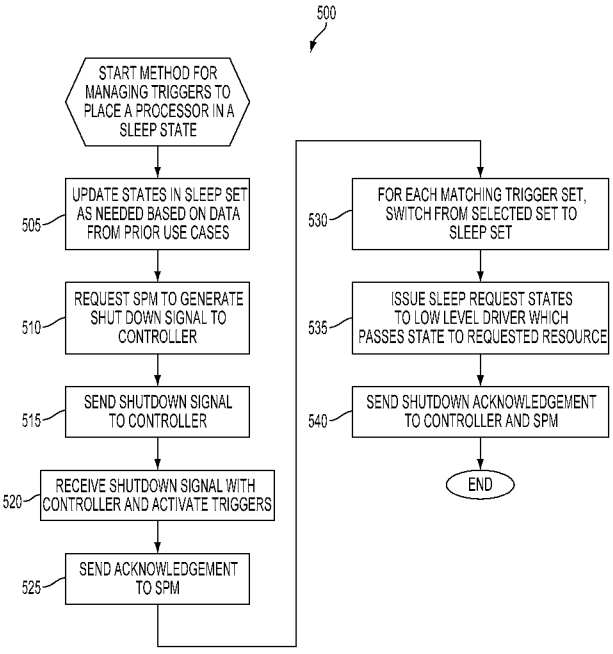

FIG. 5

(57) Abstract: Resource state sets of a portable computing device are managed. A sleep set of resource states, an active set of resource states and a next-active set of resource states are maintained in memory. A request may be issued for a processor to enter into a sleep state or otherwise change from one application state corresponding to one resource state set to another application state corresponding to another application state set. This causes a controller to review a trigger set to determine if a shut down condition for the processor matches one or more conditions listed in the trigger set. If a trigger set matches a shut down condition, then switching states of one or more resources in accordance with the sleep set may be made by the controller. Providing a next-awake set of resource states that is immediately available to the processor upon a wake-up event helps minimize resource latency.

WO 2012/087533 A1

LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

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

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

MINIMIZING RESOURCE LATENCY BETWEEN PROCESSOR APPLICATION STATES IN A PORTABLE COMPUTING DEVICE BY USING A NEXT-ACTIVE STATE SET

PRIORITY AND RELATED APPLICATIONS STATEMENT

[0001] The benefit of the filing date of which and of U.S. Provisional Patent Application Serial No. 61/425,677, filed on December 21, 2010, entitled "METHOD AND SYSTEM FOR RAPID ENTRY INTO AND FOR RAPID EXITING FROM SLEEP STATES FOR PROCESSORS OF A PORTABLE COMPUTING DEVICE," and of U.S. Provisional Patent Application Serial No. 61/544,087, filed on October 6, 2011, entitled "MINIMIZING RESOURCE LATENCY BETWEEN PROCESSOR APPLICATION STATES BY USING A NEXT-ACTIVE STATE SET," the specifications of which are also incorporated herein in their entireties by this reference, are hereby claimed. This application is related to co-pending U.S. Patent Application Serial No. _____, filed _____, entitled "MINIMIZING RESOURCE LATENCY BETWEEN PROCESSOR APPLICATION STATES IN A PORTABLE COMPUTING DEVICE BY SCHEDULING RESOURCE SET TRANSITIONS," and this application is related to co-pending U.S. Patent Application Serial No. 13/069,071, filed March 22, 2011, entitled "METHOD AND SYSTEM FOR RAPID ENTRY INTO AND FOR RAPID EXITING FROM SLEEP STATES FOR PROCESSORS OF A PORTABLE COMPUTING DEVICE," both of which are assigned to the assignee of the present application.

DESCRIPTION OF THE RELATED ART

[0002] Portable computing devices ("PCDs") are becoming necessities for people on personal and professional levels. These devices may include cellular telephones, portable digital assistants ("PDAs"), portable game consoles, palmtop computers, and other portable electronic devices.

[0003] PCDs typically have complex and compact electronic packaging that is generally made of multiple processing units that include central processing units, digital signal processors, and the like. Much of this hardware may be part of a system on a chip ("SOC") design as understood by one of ordinary skill in the art.

[0004] Conventional PCD's usually experience significant lag time when respective processors of different SOCs try to enter into low power states. Low power states, in

which a processor or similar subsystem is not executing any application program or is otherwise effectively idle, are also referred to as sleep states, as understood by one of ordinary skill in the art.

[0005] One problem faced by conventional processors is that several communications usually take place in software in order for a processor to enter into a sleep state. This problem is further complicated by the fact that some resources are shared resources whose state needs to be coordinated between multiple SOC subsystems.

[0006] Within a given subsystem of SOC, the management of local resources is usually easy and can be done from the respective operating system's idle context. However, to manage the shutdown of a shared resources state usually has to be coordinated with the controller of that resource. Conventional solutions have worked around this shutdown complication through use of synchronous handshake in software before the subsystems are permitted to enter a sleep state. This approach is disadvantageous for several reasons: Software handshakes are slow. Software handshakes are prone to all sorts of delay, particularly interrupt service and context switch problems.

[0007] Software handshakes delay power savings. Because a handshake is in software, the instruction processing core needs to remain on until the full handshake is complete. Processor cores are large and complex, thus this is a considerable penalty in power savings to pay.

[0008] Accordingly, what is needed in the art is a method and system for allowing processors of PCDs to enter sleep states without software handshakes.

SUMMARY

[0009] A method and system for managing sleep states of a portable computing device are described. Three resource state sets are managed. In exemplary embodiment, a sleep set of resource states, an active set of resource states and a next-awake set of resource states are provided to a controller memory. The resource state sets may be modified based on prior usage of the portable computing device. A request may be issued for a processor to enter into a sleep state. This causes the controller to review a trigger set to determine if a shut down condition for the processor matches one or more conditions listed in the trigger set. Each trigger set may comprise a "trigger event" that may allow a controller to select a specific resource set which is desired by a particular processor based on a trigger event detected by a system power manager. If a trigger set

matches a shut down condition, then switching states of one or more resources in accordance with the sleep set may be made by the controller without using a software handshake. Providing a next-active set of resource states, such as a next-awake set, which is immediately available to the processor upon a wake-up or other processor application state change event, helps minimize resource latency.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] In the figures, like reference numerals refer to like parts throughout the various views unless otherwise indicated. For reference numerals with letter character designations such as "102A" or "102B", the letter character designations may differentiate two like parts or elements present in the same figure. Letter character designations for reference numerals may be omitted when it is intended that a reference numeral to encompass all parts having the same reference numeral in all figures.

[0011] FIG. 1 is a functional block diagram illustrating an embodiment of a portable computing device (PCD);

[0012] FIG. 2 is a functional block diagram illustrating relationships among a controller, a system power manager, master processors, low-level drivers, shared resources, and local resources;

[0013] FIG. 3 is a functional block diagram illustrating details about the controller and trigger sets;

[0014] FIG. 4 illustrates an exemplary active-sleep trigger set for a processor;

[0015] FIG. 5 is a logical flowchart illustrating a method for managing trigger sets and otherwise transitioning a processor from a first application state, such as an awake state to a second application state, such as a sleep state;

[0016] FIG. 6 is a logical flowchart illustrating a method for managing triggers sets and otherwise transitioning a processor from the second application state, such as a sleep state to a third application state, such as an awake state;

[0017] FIG. 7 is a functional block diagram of controller buffer memory;

[0018] FIG. 8 is a logical flowchart illustrating an alternative method for transitioning a processor from a first application state, such as an awake state, to a second application state, such as a sleep state;

[0019] FIG. 9 is a functional block diagram of an alternative controller buffer memory; and

[0020] FIG. 10 is a logical flowchart illustrating another alternative method for transitioning a processor from a first application state, such as an awake state, to a second application state, such as a sleep state.

DETAILED DESCRIPTION

[0021] The word "exemplary" is used herein to mean "serving as an example, instance, or illustration." Any aspect described herein as "exemplary" is not necessarily to be construed as preferred or advantageous over other aspects.

[0022] In this description, the term "application" may also include files having executable content, such as: object code, scripts, byte code, markup language files, and patches. In addition, an "application" referred to herein, may also include files that are not executable in nature, such as documents that may need to be opened or other data files that need to be accessed.

[0023] The term "content" may also include files having executable content, such as: object code, scripts, byte code, markup language files, and patches. In addition, "content" referred to herein, may also include files that are not executable in nature, such as documents that may need to be opened or other data files that need to be accessed.

[0024] As used in this description, the terms "component," "database," "module," "system," and the like are intended to refer to a computer-related entity, either hardware, firmware, a combination of hardware and software, software, or software in execution. For example, a component may be, but is not limited to being, a process running on a processor, a processor, an object, an executable, a thread of execution, a program, and/or a computer. By way of illustration, both an application running on a computing device and the computing device may be a component. One or more components may reside within a process and/or thread of execution, and a component may be localized on one computer and/or distributed between two or more computers. In addition, these components may execute from various computer readable media having various data structures stored thereon. The components may communicate by way of local and/or remote processes such as in accordance with a signal having one or more data packets (*e.g.,* data from one component interacting with another component in a local system, distributed system, and/or across a network such as the Internet with other systems by way of the signal).

[0025] In this description, the terms "communication device," "wireless device," "wireless telephone," "wireless communication device," and "wireless handset" are used interchangeably. With the advent of third generation ("3G") and fourth generation ("4G") wireless technology, greater bandwidth availability has enabled more portable computing devices with a greater variety of wireless capabilities.

[0026] In this description, the term "portable computing device" ("PCD") is used to describe any device operating on a limited capacity power supply, such as a battery. Although battery operated PCDs have been in use for decades, technological advances in rechargeable batteries coupled with the advent of third generation ("3G") and fourth generation ("4G") wireless technology, have enabled numerous PCDs with multiple capabilities. Therefore, a PCD may be a cellular telephone, a satellite telephone, a pager, a PDA, a smartphone, a navigation device, a smartbook or reader, a media player, a combination of the aforementioned devices, and a laptop computer with a wireless connection, among others.

FIG. 1: ELEMENTS OF PCD 100 FOR MINIMIZING RESOURCE LATENCY BETWEEN PROCESSOR APPLICATION STATES

[0027] Referring to FIG. 1, this FIG. is a functional block diagram of an exemplary, non-limiting aspect of a PCD 100 in the form of a wireless telephone for implementing methods and systems for managing rapid sleep states of processors 110, 126 within the PCD 100. As shown, the PCD 100 includes an on-chip system 102 that includes a multi-core, first central processing unit ("CPU") 110A, a second CPU 110B that is a single-core type, and an analog signal processor 126.

[0028] These three processors 110A, 110B, and 126 may be coupled together. The first CPU 110A may comprise a zeroth core 222, a first core 224, and an Nth core 230 as understood by one of ordinary skill in the art. In an alternate embodiment, instead of using two CPUs 110, two digital signal processors ("DSPs") may also be employed as understood by one of ordinary skill in the art. In a further exemplary embodiment, any of the aforementioned may used in a combination as understood by one of ordinary skill in the art.

[0029] FIG. 1 includes one or more controller module(s) 101. For the remainder of this description, the controller module(s) 101 will be referred to in the singular, as a controller 101, and not plural. One of ordinary skill in the art will recognize that the controller 101 may be divided into various parts and executed by different processors

110, 126 without departing from the invention. Alternatively, the controller 101 may be organized as a single element and executed by a single processor 110 or 126.

[0030] FIG. 1 also illustrates system power manager 157. The system power manager ("SPM") 157 is coupled to the CPU 110A and the controller 101. The SPM 157 generally comprises hardware, such as a processor. However, software and/or firmware may be employed for the SPM 157 as understood by one of ordinary skill in the art. The SPM 157 may be responsible for monitoring the state of a processor 110, 126 and a power rail. The SPM 157 may detect when a processor 110, 126 is about to enter a sleep state or is about to leave a sleep state. The SPM 157 may communicate these states of a processor 110, 126 to the controller 101. More generally, the SPM 157 may detect when a processor 110, 126 is about to transition from one application state to another. Application states of a processor 110, 126 may include not only a sleep state in which the processor 110, 126 is effectively idle or not executing any application programs and an awake or active state in which it is executing one or more application programs but also, or alternatively, any of the following: a state in which the processor 110, 126 is operating at a higher or lower speed than it operates in another state; a state defined by the processor 110, 126 executing an application program that is different from another state defined by the processor 110, 126 executing another application program; and a state defined by the processor 110, 126 concurrently executing a number of application programs that is different from another state defined by the processor 110, 126 concurrently executing a different number of application programs.

[0031] The controller 101 may comprise software which is executed by the CPUs 110. However, the controller 101 may also be formed from hardware and/or firmware as understood by one of ordinary skill in the art.

[0032] In general, the controller 101 may be responsible for promoting the rapid entry into sleep states and the rapid exiting from sleep states for the processors 110, 126. The controller 101 may include one or more tables that comprise resource sets and trigger sets as will be described in further detail below in connection with FIG. 3. The controller 101 may also have its own interrupt controller (not illustrated) for when all other hardware elements in the PCD 100 are placed in a low power state and are not functional.

[0033] The controller 101 also manages resource requests among one or more master processors 110, 126. Resource requests may be issued by a master processor 110 to request an action or function from a resource 105 (See FIG. 2).

[0034] Resources 105 are described more generally below but may include, for example, clocks and other low-level processors that support tasks, commands, and features of software applications that are executed by one or more master processors 110, 126. The controller 101 may be designed to prevent resource request conflicts among a plurality of master processors 110, 126.

[0035] FIG. 1 shows that the PCD 100 may include memory 112. The controller 101 running on the CPUs 110 may access the memory 112 to facilitate rapid sleep states and to facilitate rapid exiting from sleep states as will be described in further detail below.

[0036] In a particular aspect, one or more of the method steps described herein may implemented by executable instructions and parameters stored in the memory 112 that form the controller 101. These instructions that form the controller 101 may be executed by the CPUs 110, the analog signal processor 126, or another processor. Further, the processors, 110, 126, the memory 112, the instructions stored therein, or a combination thereof may serve as a means for performing one or more of the method steps described herein.

FIG. 1: OTHER ELEMENTS OF THE PCD 100

[0037] As illustrated in FIG. 1, a display controller 128 and a touchscreen controller 130 are coupled to the digital signal processor 110. A touchscreen display 132 external to the on-chip system 102 is coupled to the display controller 128 and the touchscreen controller 130.

[0038] FIG. 1 is a schematic diagram illustrating an embodiment of a portable computing device (PCD) that includes a video coder/decoder ("codec") 134, e.g., a phase-alternating line ("PAL") encoder, a sequential couleur avec memoire ("SECAM") encoder, a national television system(s) committee ("NTSC") encoder or any other type of video encoder 134. The video codec 134 is coupled to the multicore central processing unit ("CPU") 110. A video amplifier 136 is coupled to the video encoder 134 and the touchscreen display 132. A video port 138 is coupled to the video amplifier 136. As depicted in FIG. 1, a universal serial bus ("USB") controller 140 is coupled to the CPU 110. Also, a USB port 142 is coupled to the USB controller 140. A subscriber identity module (SIM) card 146 may also be coupled to the CPU 110. Further, as shown in FIG. 1, a digital camera 148 may be coupled to the CPU 110. In an exemplary aspect, the digital camera 148 is a charge-coupled device ("CCD") camera or a complementary metal-oxide semiconductor ("CMOS") camera.

[0039] As further illustrated in FIG. 1, a stereo audio CODEC 150 may be coupled to the analog signal processor 126. Moreover, an audio amplifier 152 may be coupled to the stereo audio CODEC 150. In an exemplary aspect, a first stereo speaker 154 and a second stereo speaker 156 are coupled to the audio amplifier 152. FIG. 1 shows that a microphone amplifier 158 may be also coupled to the stereo audio CODEC 150. Additionally, a microphone 160 may be coupled to the microphone amplifier 158. In a particular aspect, a frequency modulation ("FM") radio tuner 162 may be coupled to the stereo audio CODEC 150. Also, an FM antenna 164 is coupled to the FM radio tuner 162. Further, stereo headphones 166 may be coupled to the stereo audio CODEC 150.

[0040] FIG. 1 further indicates that a radio frequency ("RF") transceiver 168 may be coupled to the analog signal processor 126. An RF switch 170 may be coupled to the RF transceiver 168 and an RF antenna 172. As shown in FIG. 1, a keypad 174 may be coupled to the analog signal processor 126. Also, a mono headset with a microphone 176 may be coupled to the analog signal processor 126. Further, a vibrator device 178 may be coupled to the analog signal processor 126. FIG. 1 also shows that a power supply 180, for example a battery, is coupled to the on-chip system 102. In a particular aspect, the power supply 180 includes a rechargeable DC battery or a DC power supply that is derived from an alternating current ("AC") to DC transformer that is connected to an AC power source.

[0041] As depicted in FIG. 1, the touchscreen display 132, the video port 138, the USB port 142, the camera 148, the first stereo speaker 154, the second stereo speaker 156, the microphone 160, the FM antenna 164, the stereo headphones 166, the RF switch 170, the RF antenna 172, the keypad 174, the mono headset 176, the vibrator 178, thermal sensors 157B, and the power supply 180 are external to the on-chip system 322.

[0042] Some of the above-described elements of the PCD 100 may comprise hardware, while others may comprise software, and still others may comprise a combination of hardware and software. The term "resource" is used herein to refer to any such element, whether hardware, software or a combination thereof, that is controllable by a processor. A resource may be defined in one aspect as an encapsulation of the functionality of such an element. Except where it may otherwise be indicated, the term "processor" or "master processor" is used herein to refer to a processor such as the first CPU 110A, the second CPU 110B, the analog signal processor 126, or to any other processor, controller or similar element that operates

under the control of software, firmware, or similar control logic. As described in further detail below, an example of a resource is a software element that executes on a processor. A thread of execution on a processor, such as, for example, a thread relating to an executing application program, may access a resource by causing a "request" to be issued on the resource.

[0043] In different application states, it may be necessary or desirable for a processor to request different configurations or states of resources. For example, a bus resource may control the speed of a bus clock. In one application state a processor may request a bus clock that allows the processor to operate at a rate of, for example, 100 million instructions per second (MIPS), while in another application state the processor may request a bus clock that allows it to operate at a rate of, for example, 150 MIPS. In the case of a processor preparing to enter an application state that is a sleep state, the processor may request a bus clock of zero MIPS. Similarly, in one application state defined by a processor executing a first application program the processor may request 100 MIPS, while in another application state defined by the processor executing a second application program the processor may request 150 MIPS. Likewise, in one application state defined by a processor concurrently executing a certain number of application programs the processor may request 100 MIPS, while in a second application state defined by the processor concurrently executing a different number of application programs the processor may request 150 MIPS. It should be understood that the above-referenced bus clock is intended only as an example of a resource that can be configured by a processor issuing a resource request, and also that the numbers "100" and "150" are intended as arbitrary examples of processing speeds.

[0044] Resource configurations or states may be grouped into resource state sets. A resource state set defines the configurations or states of one or more resources that are used together by a processor in a certain processor application state. For example, a certain resource state set may include configuration or state information for a bus clock resource to provide a processor with a certain number of MIPS of processing speed, and configuration or state information for a decoder (i.e., another example of a resource) to provide a decoding function to the processor.

[0045] FIG. 2 is a functional block diagram illustrating relationships among the controller 101, system power manager 157, master processors 110, 126, low-level drivers 103, shared resources 105A-C, and local resources 105D-H that form a system 103. FIG. 2 also illustrates how the touchscreen 132 may be coupled to the touchscreen

driver/controller 130. The touchscreen driver/controller 130 may be coupled to clock code 113A of a first master processor 110A.

[0046] The system 103 may switch among resource state sets desired by a processor 110 in a manner that minimizes resource latency. The term "resource latency" refers to the delay or latency that occurs between a time at which a master processor 110, 126 begins preparing controller 101 and system power manager 157 to transition to another resource state set and the time that the resources of that set become configured to the specified states and ready for use by the processor. As described below, resource state sets can be broadly categorized into: active resource state sets, in which a processor is provided with resources configured to aid the processor in executing application programs and otherwise providing processing power; and a sleep resource state, in which a processor is provided only with resources that aid the processor in maintaining a sleep state, i.e., a state in which the processor is not executing application programs or otherwise providing processing power. Although a processor in a sleep state may maintain low-level functions, the processor does not execute software that would be understood by one of ordinary skill in the art to be an application program. It should be understood that the "next-active state" feature described below may be applied to transitions between any resource state sets, regardless of whether they may be active sets or sleep sets.

[0047] In the exemplary embodiment shown in FIG. 2, the first master processor 110A may be coupled to the system power manager 157 and the controller 101. The controller 101 may be coupled to the clock code 113A of the first master processor 110A. The controller 101 may comprise one or more low-level drivers 103. The one or more low-level drivers 103 may be responsible for communicating with one or more shared resources 105A-C. Shared resources 105A-C may comprise any type of device that supports tasks or functions of a master processor 110. Shared resources 105A-C may include devices such as clocks of other processors as well as single function elements like graphical processors, decoders, and the like.

[0048] The shared resources 105A-C may be coupled to one or more local resources 105D-H. The one or more local resources 105D-H may be similar to the shared resources 105A-C in that they may comprise any type of device that supports or aids tasks or functions of a master processor 110. Local resources 105D-H may include devices such as clocks of other processors as well as single function elements like graphical processors, decoders, and the like. The local resources 105D-H may comprise

leaf nodes. Leaf nodes are understood by one of ordinary skill in the art as local resources 105D-H that usually do not refer or include other dependent resources 105.

[0049] The controller 101 may be responsible for managing requests that are issued from the one or more master processors 110, 126. For example, the controller 101 may manage a request that originates from the first master processor 110A. The first master processor 110A may issue this request in response to an operator manipulating the touchscreen 132. The touchscreen 132 may issue signals to the touchscreen driver/controller 130. The touchscreen driver/controller 130 may in turn issue signals to the clock code 113A of the first master processor 110A.

[0050] The controller 101 may also be responsible for managing the sleep states for a particular processor 110. Prior to entering a sleep state, a processor 110 will provide information for managing sleep states. Information for managing sleep states includes the entry into and exiting from a sleep state. This information for managing sleep states will be referred to below as triggers and resource states. A resource state set may include resource information for configuring one or more resources in a manner that supports a sleep state of a processor.

[0051] Triggers may define events that cause a processor 110 to either enter into a sleep state or to leave a sleep state. Triggers will generally reference resource states that are contained within or that are accessible by the controller 101. Resource states define a desired state of resources 105 needed by particular processor 110. In an exemplary embodiment, each processor 110 may provide at least two resource state sets to a controller 101: an active set of resource states and a sleep set of resource states. However, in other embodiments a processor may provide resource state sets in addition to a single active set and a single sleep set or resource state sets that are different from a single active set and a single sleep set. Such other resource state sets may correspond to one or more of the processor application states described above. That is, for any application state, the processor may provide a corresponding resource state set.

[0052] In the exemplary embodiment, the active set of resource states may define states of resources 105 for when the processor 110 is actively performing processing functions and requiring action/functions from its resources 105. The sleep set of resource states may define states of resources 105 when the processor 110 is in a sleep or idle state. Further details about triggers and resource states will be described below in connection with FIG. 3.

[0053] FIG. 3 is a functional block diagram illustrating details about the controller 101, resource sets 304, and trigger sets 314. As noted previously, the controller 101 may comprise software executed by one or more of the processors 110, 126 of the PCD 100. The controller 101 may store information in memory 112 or in an area within the controller 101, such as local storage as understood by one of ordinary skill in the art. This information may comprise a resource table 302 that includes resource sets 304 that are assigned to each master processor 110 which is serviced by the controller 101. This information may also comprise trigger sets 314 that are also assigned to each master processor 110 and which may be unique to each master processor 110.

[0054] Each resource set 304 generally comprises information relating to states of resources 105 desired by a particular master processor 110. Each resource set 304 assigned to a particular master processor 110 may comprise an active resource set 306, and a sleep resource set 308. The active resource set 306 may define or describe states of resources 105 when a particular master processor 110 is active or functioning normally. The sleep resource set 308 may define or describe states of resources 105 when a particular master processor is in a sleep or dormant state as understood by one of ordinary skill in the art. Each resource set 304 may also comprise additional sets such as "set 1" and "set 2" assigned to the first master processor 110 in the exemplary embodiment illustrated in FIG. 3.

[0055] As an example, the active resource set 306 for the first master processor (A) 110A as illustrated in FIG. 3 has assigned the following values for each of its resources 105: for the first shared resource (SR#1) 105A the value is one; the value for the second shared resource (SR#2) 105B is one; the value for the Nth shared resource (SR#N) 105C is one; while the four values for the first local resource (LR#1) 105D are one, zero, one, and one.

[0056] As noted previously, states of resources 105 are not limited to single values and may include a plurality of values. Further, states of resources may include any of a number of different types of parameters. For example, a state may designate hundreds of megahertz for the amount of clock speed of a particular clock that may function as a resource 105.

[0057] As another example, the sleep resource set 308A for the first master processor (A) 110A as illustrated in FIG. 3 has assigned the following values for each of its resources 105: for the first shared resource (SR#1) 105A, this resource has been assigned value of zero; the second shared resource (SR#2) 105B has an assigned value

of zero; while the Nth shared resource (SR#N) 105C has an assigned value of zero. The first local resource (LR#1) 105D may have assigned values of zero, one, zero and zero.

[0058] Each trigger set 314 assigned to a particular master processor 110 may comprise at least three fields: an interrupt field 316; a "from set" 318; and a "go to set" 320. Each of these three fields of a trigger set 314 may also include a corresponding set of three columns: a trigger start column 322; a clear column 324; and a timer column 326.

[0059] The interrupt field 316 describes the action or activity that may be generated and/or detected by the system power manager 157. The interrupt field 316 may be generally characterized as the "trigger event" that may allow a controller 101 to select a specific resource set 304 which is desired by a particular processor 110 based on the trigger event detected by the SPM 157. The selection of a resource set 304 by the controller 101 may avoid the time consuming software handshake described above in the background section.

[0060] Reviewing the first trigger set (trigger set #1) of FIG. 3 for the first master processor (A) 110A, the fields of the set are discussed in order by columns. Starting with the first column of the trigger set 314A, the trigger start column 322 has an action listed as "decode interrupt" in its first row corresponding to the interrupt field 316.

[0061] As noted previously, the interrupt field 316 may define parameters that cause the controller 101 to activate the states of a resource set 304 in response to the detection of the trigger start field 322. In the exemplary embodiment illustrated in FIG. 3, the interrupt field 316A has been defined or described as a "decode interrupt" which means that when the system power manager 110 detects a "decode interrupt," such as when a PCD 100 is decoding video, then this event may alert the controller 101 to review the "from set" field 318 in the first column 322A1 under the "trigger start" column.

[0062] The "from set" field 318 may comprise a value that denotes what the current resource set 304 should be for the particular master processor 110 being reviewed by the controller 101. This field 318 may list a resource set 304 by its identifier such as the "active set," the "sleep set," or a set number like "set 1" or "set 2," The field 320 may also comprise a "wild card" like an asterisk.

[0063] A wildcard designation in the "from set" field 318 may cause the controller 101 to retrieve the last known active resource set 304 that was being used by a particular master processor 101. In the exemplary embodiment illustrated in FIG. 3, the "from set" row 318A and trigger start column 322A1 have a value of an asterisk or wildcard.

[0064] The "go to set" 320, like the "from set" 318, may comprise a listing of a resource set 304 by its identifier such as the "active set", the "sleep set", or a set number like "set 1" or "set 2". The field 320 may also comprise a "wild card" like an asterisk that means the last resource set 304 being utilized by a processor 110. In the exemplary embodiment illustrated in FIG. 3, the "go to set" field 320A and the trigger start field column 322 A1 has a value of "set 1" which is the resource set 1 listed in column 310A of the first resource set 304A.

[0065] For the example illustrated in FIG. 3, when a decode interrupt event is detected by the SPM 157, it alerts the controller 101. The controller 101 reviews the first trigger set for the first master processor 110. Since the trigger start column 322A1 lists a matching value (a decode interrupt), the controller 101 reviews the "from set" field 318A and determines that the value is a wildcard value or asterisk. The controller 101 then reviews the "go to" field 320A which has a value of "set 1" that designates a particular resource set 304A. Based on this information reviewed by the controller 101, the controller 101 will switch the current resource set 304A for the first master processor 110A from its current set to the resource set "set 1." Resource Set 1 is listed in column 310A of the resource set 304A assigned to the first master processor 110A.

[0066] Further, when the SPM 157 or the controller 101 detects a "not decode" event such as illustrated in the clear column 324A1 of the first trigger set, then the controller 101 will then review the "from set" field 318A and determine that this value comprises "set 1." The controller 101 will then review the "go to set" field 320 which has a value of a wildcard or an asterisk in this example. This means that the controller 101 will switch the resource set 304A of the first master processor 110A from the "set 1" resource set to the last active resource set used by the processor 110A.

[0067] The timer field 326 of the trigger set may denote an amount of time that a particular resource set 304 may be used by the controller 101. So for the exemplary embodiment illustrating FIG. 3, for the timer field 326A1 of the first trigger set, this field has a value of three milliseconds. This means that when the decode interrupt event is matched with the trigger start field 322A1 of the first trigger set, then the controller 101 utilizes the resource set 304 specified in the "go to set" field 320A for only a period of three milliseconds. In other exemplary embodiments, situations may occur or exist in which there is no information in the timer field 326 or the value is defined to correspond with a value that indicates that there is no timer trigger 326 for this transition and that the transition only applies to the no decode field. In a situation in which the timer field

is defined, such as illustrated in FIG. 3 – timer fields 326A1 and 326A2, then whichever event occurs first between the timer field 326 and the Clear field 324 will usually initiate the transition.

[0068] FIG. 4 illustrates an exemplary active-sleep trigger set 314 for a processor 110. In this exemplary embodiment, the interrupt field 316 in the first column 322 define a "shut down" event as the action to initiate a sleep set 308 (FIG. 3) for a particular processor 110. The "shut down" event may include action like an operator selecting an on/off button for shutting down a PCD 100.

[0069] In the exemplary embodiment in FIG. 4, when a "shut down" event is detected, the controller 101 transitions the current active resource set 306 to a sleep set 308. The sleep set 308 is listed in a master resource set 304 of table 302 in FIG. 3.

[0070] When the controller 101 receives a message from the SPM 157 that a "bring up" event has occurred, such as a power-on event initiated by an operator of the PCD 100, then the controller would transition the processor 110 from its sleep set 308 to the last active resource set 304 based on the wildcard or asterisk value listed in the "go to set" field 320 of the trigger set 314.

[0071] As described above, the system 103 is not limited to active and sleep sets 306, 308. The system 103 may be used for switching between resource sets 304 for events other than entering or exiting sleep states as illustrated in FIG. 3.

[0072] FIG. 5 is a logical flowchart illustrating a method 500 for managing trigger sets 314 to place a processor 110 into a sleep state. Block 505 is the first step of the method 500. In block 505, each processor 110 may update its resource sets 304 as well as its trigger sets 314 in the controller 101 (FIGS. 1-2) as needed based on data from prior use cases of the PCD 100.

[0073] In block 510, a processor 110 may request the SPM 157 (FIG. 2) to generate a shutdown signal to the controller 101. In block 515, the SPM 157 may send the shutdown signal to the controller 101.

[0074] The controller 101 may receive the shutdown signal in block 520 and activate the trigger sets 314 which may be assigned to a shutdown event as illustrated in FIG. 4. In the exemplary embodiment illustrated in FIG. 4, the shutdown signal is matched against the interrupt field 316 of the trigger set 314. The trigger set 314 directs the controller 101 to access a sleep set 308 as indicated in the "go to set" field 320. In block 525, the controller 101 may immediately send an acknowledgment signal to the

SPM 157 while the controller 101 continues to activate resource sets 304 that are referenced by the trigger sets 314 which match the shutdown signal event.

[0075] In block 530, for each matching trigger set 314, such as the matching trigger set 314 listing the "shutdown" event in the corresponding interrupt field 316 illustrated in FIG. 4, the controller 101 may switch the current resource set 304 to a sleep set 308, such as the sleep set 308A of the first resource set 305A for the master processor 110A of FIG. 3.

[0076] Next, in block 535, the controller 101 may issue sleep request states to low-level drivers 103 such as illustrated in FIG. 2. The low-level drivers 103 may pass the requested states to the corresponding resources 105.

[0077] In block 540, each resource 105 may issue a shutdown signal acknowledgment to the controller 101 and the SPM 157. The method 500 may then end.

[0078] FIG. 6 is a logical flowchart illustrating a method 600 for managing trigger sets 314 to place a processor 110 in an active state from a sleep state. Block 605 is the first step in method 600. In block 605, a wake-up condition or wake-up event is detected with the SPM 157, or the wake-up event is detected directly by the controller 101, which may have its own interrupt controller (not illustrated). Exemplary embodiments may be designed such that wakeup interrupts may not be detectable by the SPM 157. In such exemplary embodiments, the controller 101 may use its interrupt controller to detect them and have these "mapped" to sleep set requirements for a master processor 110.

[0079] Next, in block 610 the SPM 157 may send a wake-up signal to the controller 101. In block 615, the controller 101 may receive the wake-up signal from the SPM 157 and activate one or more trigger sets 314 that matched the wake-up signal. For example, the controller 101 may match the wake-up signal with the "bring up" event listed in the interrupt field 316 in the "active" column of the trigger set 314 of FIG. 4. In the exemplary embodiment of FIG. 4, the "go to field' 320 in the active column 324 directs the controller to the last resource set 304 which was used by the current processor 110.

[0080] So in block 620, the controller 101 would change the current resource set 304 for a processor 110 based on this matching trigger set 314. One of ordinary skill in the art recognizes that the controller 101 will cycle through all of its trigger sets that it maintains as illustrated in FIG. 3.

[0081] Next, in block 625, the controller 101 may send a wake-up acknowledgment to the SPM 157 identifying which master processors 110 have been awakened from the sleep state. Next, in block 630, each processor 110 with a matching wake up trigger set 314 is released from a sleep state and restored to its active state with power supplied by the SPM 157. The method 600 then ends.

[0082] FIGS. 7-10 illustrate another feature, which is referred to in this description as "next-active resource state set" or "next-active set." One example of a next-active set is a next-awake set. The next-awake set or other next-active set may be used in the same manner described above with regard to FIG. 6 and the resource set 304 to which the controller 101 switches upon a wake-up event.

[0083] FIG. 7 is similar to FIG. 3 in that it represents information stored in the controller 101. In an exemplary embodiment, the controller 101 may include three memory buffers, referred to in this description for convenience as the "A" memory buffer 702, the "B" memory buffer 704, and the "C" memory buffer 706.

[0084] FIG. 8 is a logical flowchart similar to FIG. 5 in that it illustrates a method 800 for placing a processor into a sleep state. Block 805 is the first step of the method 800 and is similar to block 505 described above with regard to FIG. 5. Block 805 indicates that processor 110 may update not only an active or awake resource state set and a sleep resource state set but also a next-awake rsource state set. As shown in FIG. 8, the processor may cause the active set to be stored in the "A" buffer 702 (FIG. 7) of the controller 101, the sleep set to be stored in the "B" buffer 704 (FIG. 7) of the controller 101, and the next-awake set to be stored in the "C" buffer 706 (FIG. 7) of the controller 101. Other aspects of block 805 are the same as described above with regard to block 505 and are therefore not described here.

[0085] Blocks 810, 815, 820, 825, 830, 835 and 840 are the same as blocks 510, 515, 520, 525, 530, 535 and 540, respectively, of FIG. 5 and are therefore not described here. Note that when the processor begins shutting down, it is in the awake application state corresponding to the awake set stored in the "A" buffer 702 (FIG. 7). The processor then enters the sleep application state corresponding to the sleep set that is stored in the "B" buffer 704 (FIG. 7) in the same way as described above with regard to FIG. 5. The processor awakes (FIG. 6) from the sleep application state in the next-awake application state corresponding to the next-awake set that is stored in the "C" buffer 706 (FIG. 7). By pre-storing the next-awake set updates in the "C" buffer 706 and applying them as soon as possible, the controller 101 is able to immediately begin configuring the

resources specified by that next-awake set upon a wake-up event, thereby helping to minimize resource latency.

[0086] FIG. 9 relates to another exemplary embodiment, in which the controller 101 does not have sufficient memory to simultaneously store all three of the above-described resource state sets. In this embodiment, the controller 101′ has only an "A" buffer 902 and a "B" buffer 904, and there is no memory space available for a "C" buffer. In such an instance, the "A" buffer 902 is re-used so that at different times it stores the (then-current) awake set as well as the next-awake set.

[0087] FIG. 10 is a logical flowchart similar to FIGS. 5 and 9 in that it illustrates a method 1000 for placing a processor into a sleep state. Block 1005 is the first step of the method 800 and is similar to block 805 described above with regard to FIG. 8 but does not include storing the next-awake set in a "C" buffer. Rather, the processor may cause the active set to be stored in the "A" buffer 902 of the controller 101' and the sleep set to be stored in the "B" buffer 904 of the controller 101', but the processor waits until after it has reached a "point of no return" (as the term is understood by one of ordinary skill in the art) in transitioning to the sleep application state before re-using the "A" buffer to store the next-awake set. Other aspects of block 1005 are the same as described above with regard to block 505 (FIG. 5) and are therefore not described here.

[0088] In block 1008, the processor performs what may be referred to as a pseudo-update or virtual update of the next-awake set. Note that in the above-described block 1005 the processor may perform actual updates of resource state sets by writing the resource state sets to the "A" buffer 902 and "B" buffer 904 in the controller 101′. The updates are actual because the controller 101′ receives an interrupt from the processor to notify it that the buffer contents have been updated, causing the controller 101′ to act upon or apply the updates. The controller 101′ applies the updates by performing various tasks that may be necessary to prepare the updated resource state set information for use. If the sleep set in buffer "B" is updated, the controller 101′ may prepare the updated sleep set information for use in case a shutdown event or similar event that requires switching resource state sets subsequently occurs. If the active set in "A" buffer 902 (FIG. 9) is updated, the controller 101′ may cause the resources to be adjusted accordingly. The pseudo-update that the processor performs in block 1008 includes storing updates for the next-awake set in "A" buffer 902 (FIG. 9) without sending an interrupt to the controller 101′. Because the controller 101′ receives no interrupt, it does not yet apply the updates that occurred in "A" buffer 902 (FIG. 9).

This pseudo-update occurs after a point of no return in which the processor will request the SPM 157 (FIG. 2) to signal a shutdown to the controller 101′ and is assured not to make any further updates to the then-active resource set state information in the "A" buffer 902 (FIG. 9).

[0089] Blocks 1010, 1015, 1020 and 1025 are the same as described above with regard to blocks 510, 515, 520 and 525, respectively, and are therefore not described here.

[0090] Then, in block 1027 the controller 101′ responds to the handshake that occurs between it and the processor (blocks 1020, 1025) by checking the "A" buffer 902 (FIG. 9) for updates and stores the updates to be used in the wake-up method of FIG. 6. (It may be noted that the memory buffers are also referred to as "message RAM" due to the way an interrupt is used to notify the recipient controller 101′ of "messages" that the processor has written to the buffers.) Thus, by pre-storing the next-awake set in the "A" buffer 902 (FIG. 9), the controller 101′ is able to immediately begin configuring the resources specified by that next-awake set upon a wake-up event, thereby helping to minimize resource latency.

[0091] Blocks 1030, 1035 and 1040 are the same as blocks 530, 535 and 540, respectively, of FIG. 5 and are therefore not described here. The processor then accordingly enters the sleep application state corresponding to the sleep set that is stored in the "B" buffer 904 (FIG. 9) in the same way as described above with regard to FIG. 5. The processor awakes (FIG. 6) from the sleep application state in the next-awake application state corresponding to the next-awake set that is stored in the "B" buffer 904 (FIG. 9). By pre-storing the next-awake set and applying it as soon as possible, the controller 101′ can immediately begin configuring the resources specified by that next-awake set upon a wake-up event, thereby helping to minimize resource latency.

[0092] Certain steps in the processes or process flows described in this specification naturally precede others for the invention to function as described. However, the invention is not limited to the order of the steps described if such order or sequence does not alter the functionality of the invention. That is, it is recognized that some steps may performed before, after, or parallel (substantially simultaneously with) other steps without departing from the disclosed system and method. In some instances, certain steps may be omitted or not performed without departing from the method as understood by one of ordinary skill in the art. Further, words such as "thereafter",

"then", "next", etc. are not intended to limit the order of the steps. These words are simply used to guide the reader through the description of the exemplary method.

[0093] In view of the disclosure above, one of ordinary skill in programming is able to write computer code or identify appropriate hardware and/or circuits to implement the disclosed invention without difficulty based on the flow charts and associated description in this specification, for example. Therefore, disclosure of a particular set of program code instructions or detailed hardware devices is not considered necessary for an adequate understanding of how to make and use the invention. The inventive functionality of the claimed computer implemented processes is explained in more detail in the above description and in conjunction with the FIGs. which may illustrate various process flows.

[0094] In one or more exemplary aspects, the functions described may be implemented in hardware, software, firmware, or any combination thereof. If implemented in software, the functions may be stored as one or more instructions or code on a computer-readable medium. A computer-readable medium may include any available non-transitory media that may be accessed by a computer. By way of example, and not limitation, such computer-readable media may comprise RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage or other magnetic storage devices, or any other medium that may be used to carry or store desired program code in the form of instructions or data structures and that may be accessed by a computer.

[0095] Disk and disc, as used herein, includes compact disc ("CD"), laser disc, optical disc, digital versatile disc ("DVD"), floppy disk and blu-ray disc where disks usually reproduce data magnetically, while discs reproduce data optically with lasers. Combinations of the above should also be included within the scope of computer-readable media.

[0096] Therefore, although selected aspects have been illustrated and described in detail, it will be understood that various substitutions and alterations may be made therein without departing from the spirit and scope of the present invention, as defined by the following claims.

CLAIMS

What is claimed is:

1. A method for managing application states of a portable computing device having at least one processor and a plurality of processor resources, comprising:

maintaining a first resource state set, a second resource state set, and a third resource state set in memory;

updating the first resource state set, the second resource state set, and the third resource state set based on prior usage of the portable computing device;

issuing a request for a processor operating in a first application state corresponding to the first resource state set to transition from the first application state to a second application state corresponding to the second resource state set;

reviewing a trigger set to determine if a processor condition matches one or more conditions listed in the trigger set;

if the processor condition matches one or more conditions listed in the trigger set, then switching states of one or more resources from states indicated by the first resource state set to states indicated by the second resource state set;

issuing a request for the processor operating in the second application state corresponding to the second resource state set to transition from the second application state to a third application state corresponding to the third resource state set;

reviewing the trigger set to determine if a processor condition matches one or more conditions listed in the trigger set; and

if the processor condition matches one or more conditions listed in the trigger set, then switching states of one or more resources from states indicated by the second resource state set states to states indicated by the third resource state set.

2. The method of Claim 1, wherein:

the first resource state set is an active resource state set corresponding to a first active application state of the processor;

the second resource state set is a sleep resource state set corresponding to a sleep application state of the processor; and

the third resource state set is a next active resource state set corresponding to a next active application state of the processor.

3. The method of Claim 1, wherein the steps of reviewing the trigger set to determine if shut down conditions for the processor have been met is performed by a controller.

4. The method of Claim 1, wherein the step of maintaining a first resource state set, a second resource state set, and a third resource state in memory comprises maintaining a first, second and third resource set in memory for each master processor of a portable computing device comprising a plurality of master processors.

5. The method of Claim 1, wherein the portable computing device comprises at least one of: a cellular telephone, a satellite telephone, a pager, a personal digital assistant (PDA), a smartphone, a navigation device, a smartbook or reader, a media player, and a laptop computer with a wireless connection.

6. A method for managing application states of a portable computing device having at least one processor and a plurality of processor resources, comprising:

maintaining a first memory buffer and a second memory buffer of a controller of the portable computing device;

updating a first resource state set and a second resource state set by a processor operating in a first application state storing updates for the first resource state set in the first memory buffer and storing updates for the second second resource state set in the second memory buffer, the updates of the first resource state set and second resource state set being based on prior usage of the portable computing device;

accessing contents of the first and second memory buffers through the controller in response to the updates while the processor is operating in the first application state;

issuing a request for the processor operating in the first application state corresponding to the first resource state set to transition from the first application state to a second application state corresponding to the second resource state set;

the processor storing updates for a third resource state set in the first memory buffer after the request to transition is issued but before the controller is enabled to access the updates in the first memory buffer;

the controller reviewing a trigger set to determine if a processor condition matches one or more conditions listed in the trigger set;

if the processor condition matches one or more conditions listed in the trigger set, then the controller switching states of one or more resources from states indicated by the first resource state set to states indicated by the second resource state set; and

accessing the contents of the first memory buffer through the controller in response to the processor transitioning from the first application state to the second application state.

7. The method of Claim 6, wherein:

the updating step comprises the processor issuing an interrupt contemporaneously with the updates to indicate that information has been stored in a memory buffer, and the controller accessing contents of the first and second memory buffers in response to the interrupt; and

the processor storing updates for the third resource state set comprises the processor storing the updates in the first memory buffer without issuing the interrupt contemporaneously with the updates.

8. The method of Claim 6, wherein the controller does not have sufficient memory to simultaneously store the first, second and third resource sets.

9. The method of Claim 6, wherein:

the first resource state set is an active resource state set corresponding to a first active application state of the processor;

the second resource state set is a sleep resource state set corresponding to a sleep application state of the processor; and

the third resource state set is another active resource state set corresponding to a next active application state of the processor.

10. The method of Claim 6, wherein the step of maintaining a first memory buffer and a second memory buffer of a controller of the portable computing device comprises maintaining a resource set for only one processor at a time in each of the first and second memory buffers of a portable computing device comprising a plurality of master processors.

11. The method of Claim 6, wherein the portable computing device comprises at least one of: a cellular telephone, a satellite telephone, a pager, a personal digital assistant (PDA), a smartphone, a navigation device, a smartbook or reader, a media player, and a laptop computer with a wireless connection.

12. A computer system for managing application states of a portable computing device having at least one processor and a plurality of processor resources, the computer system comprising:

a processing entity operable for:

maintaining a first resource state set, a second resource state set, and a third resource state set in memory;

updating the first resource state set, the second resource state set, and the third resource state set based on prior usage of the portable computing device;

issuing a request for a processor operating in a first application state corresponding to the first resource state set to transition from the first application state to a second application state corresponding to the second resource state set;

reviewing a trigger set to determine if a processor condition matches one or more conditions listed in the trigger set;

if the processor condition matches one or more conditions listed in the trigger set, then switching states of one or more resources from states indicated by the first resource state set to states indicated by the second resource state set;

issuing a request for the processor operating in the second application state corresponding to the second resource state set to transition from the second application state to a third application state corresponding to the third resource state set;

reviewing the trigger set to determine if a processor condition matches one or more conditions listed in the trigger set; and

if the processor condition matches one or more conditions listed in the trigger set, then switching states of one or more resources from states indicated by the second resource state set states to states indicated by the third resource state set.

13. The computer system of Claim 12, wherein:

the first resource state set is an active resource state set corresponding to a first active application state of the processor;

the second resource state set is a sleep resource state set corresponding to a sleep application state of the processor; and

the third resource state set is a next active resource state set corresponding to a next active application state of the processor.

14. The computer system of Claim 12, wherein the steps of reviewing the trigger set are performed by a controller.

15. The computer system of Claim 12, wherein the step of maintaining a first resource state set, a second resource state set, and a third resource state in memory comprises maintaining a first, second and third resource set in memory for each master processor of a portable computing device comprising a plurality of master processors.

16. The computer system of Claim 12, wherein the portable computing device comprises at least one of: a cellular telephone, a satellite telephone, a pager, a personal digital assistant (PDA), a smartphone, a navigation device, a smartbook or reader, a media player, and a laptop computer with a wireless connection.

17. A computer system for managing application states of a portable computing device having at least one processor and a plurality of processor resources, the computer system comprising:

a processing entity operable for:

maintaining a first memory buffer and a second memory buffer of a controller of the portable computing device;

updating a first resource state set and a second resource state set by a processor operating in a first application state storing updates for the first resource state set in the first memory buffer and storing updates for the second resource state set in the second memory buffer, the updates of the first resource state set and second resource state set being based on prior usage of the portable computing device;

accessing contents of the first and second memory buffers through the controller in response to the updates while the processor is operating in the first application state;

issuing a request for the processor operating in the first application state corresponding to the first resource state set to transition from the first application state to a second application state corresponding to the second resource state set;

the processor storing updates for a third resource state set in the first memory buffer after the request to transition is issued but before the controller is enabled to access the updates in the first memory buffer;

the controller reviewing a trigger set to determine if a processor condition matches one or more conditions listed in the trigger set;

if the processor condition matches one or more conditions listed in the trigger set, then the controller switching states of one or more resources from states indicated by the first resource state set to states indicated by the second resource state set; and

accessing the contents of the first memory buffer through the controller in response to the processor transitioning from the first application state to the second application state.

18. The computer system of Claim 17, wherein:

the updating step comprises the processor issuing an interrupt contemporaneously with the updates to indicate that information has been stored in a memory buffer, and the controller accessing contents of the first and second memory buffers in response to the interrupt; and

the processor storing updates for the third resource state set comprises the processor storing the updates in the first memory buffer without issuing the interrupt contemporaneously with the updates.

19. The computer system of Claim 17, wherein the controller does not have sufficient memory to simultaneously store the first, second and third resource sets.

20. The computer system of Claim 17, wherein:

the first resource state set is an active resource state set corresponding to a first active application state of the processor;

the second resource state set is a sleep resource state set corresponding to a sleep application state of the processor; and

the third resource state set is another active resource state set corresponding to a next active application state of the processor.

21. The computer system of Claim 17, wherein the step of maintaining a first memory buffer and a second memory buffer of a controller of the portable computing device comprises maintaining a resource set for only one processor at a time in each of the first and second memory buffers of a portable computing device comprising a plurality of master processors.

22. The method of Claim 17, wherein the portable computing device comprises at least one of: a cellular telephone, a satellite telephone, a pager, a personal digital assistant (PDA), a smartphone, a navigation device, a smartbook or reader, a media player, and a laptop computer with a wireless connection.

23. A computer system for managing application states of a portable computing device having at least one processor and a plurality of processor resources, the computer system comprising:

means for maintaining a first resource state set, a second resource state set, and a third resource state set in memory;

means for updating the first resource state set, the second resource state set, and the third resource state set based on prior usage of the portable computing device;

means for issuing a request for a processor operating in a first application state corresponding to the first resource state set to transition from the first application state to a second application state corresponding to the second resource state set;

means for reviewing a trigger set to determine if a processor condition matches one or more conditions listed in the trigger set;

means for switching states of one or more resources if the processor condition matches one or more conditions listed in the trigger set from states indicated by the first resource state set to states indicated by the second resource state set;

means for issuing a request for the processor operating in the second application state corresponding to the second resource state set to transition from the second application state to a third application state corresponding to the third resource state set;

means for reviewing the trigger set to determine if a processor condition matches one or more conditions listed in the trigger set; and

means for switching states of one or more resources if the processor condition matches one or more conditions listed in the trigger set from states indicated by the second resource state set states to states indicated by the third resource state set.

24. The computer system of Claim 1, wherein:

the first resource state set is an active resource state set corresponding to a first active application state of the processor;

the second resource state set is a sleep resource state set corresponding to a sleep application state of the processor; and

the third resource state set is a next active resource state set corresponding to a next active application state of the processor.

25. The computer system of Claim 23, wherein the means for reviewing the trigger set comprises a controller.

26. The computer system of Claim 23, wherein the means for maintaining a first resource state set, a second resource state set, and a third resource state in memory comprises means for maintaining a first, second and third resource set in memory for each master processor of a portable computing device comprising a plurality of master processors.

27. The computer system of Claim 23, wherein the portable computing device comprises at least one of: a cellular telephone, a satellite telephone, a pager, a personal digital assistant (PDA), a smartphone, a navigation device, a smartbook or reader, a media player, and a laptop computer with a wireless connection.

28. A computer system for managing application states of a portable computing device having at least one processor and a plurality of processor resources, comprising:

means for maintaining a first memory buffer and a second memory buffer of a controller of the portable computing device;

means for updating a first resource state set and a second resource state set by a processor operating in a first application state storing updates for the first resource state set in the first memory buffer and storing updates for the second resource state set in the second memory buffer, the updates of the first resource state set and second resource state set being based on prior usage of the portable computing device;

means for accessing contents of the first and second memory buffers through the controller in response to the updates while the processor is operating in the first application state;

means for issuing a request for the processor operating in the first application state corresponding to the first resource state set to transition from the first application state to a second application state corresponding to the second resource state set;

means for storing updates for a third resource state set in the first memory buffer after the request to transition is issued but before the controller is enabled to access the updates in the first memory buffer;

means for reviewing a trigger set to determine if a processor condition matches one or more conditions listed in the trigger set;

means for switching states of one or more resources if the processor condition matches one or more conditions listed in the trigger set from states indicated by the first resource state set to states indicated by the second resource state set; and

means for accessing the contents of the first memory buffer through the controller in response to the processor transitioning from the first application state to the second application state.

29. The computer system of Claim 28, wherein:

the means for updating comprises the processor issuing an interrupt contemporaneously with the updates to indicate that information has been stored in a memory buffer, and the controller accessing contents of the first and second memory buffers in response to the interrupt; and

the means for storing updates for the third resource state set comprises the processor storing the updates in the first memory buffer without issuing the interrupt contemporaneously with the updates.

30. The computer system of Claim 28, wherein the controller does not have sufficient memory to simultaneously store the first, second and third resource sets.

31. The computer system of Claim 28, wherein:

the first resource state set is an active resource state set corresponding to a first active application state of the processor;

the second resource state set is a sleep resource state set corresponding to a sleep application state of the processor; and

the third resource state set is another active resource state set corresponding to a next active application state of the processor.

32. The computer system of Claim 28, wherein the means for maintaining a first memory buffer and a second memory buffer of a controller of the portable computing device comprises means for maintaining a resource set for only one processor at a time in each of the first and second memory buffers of a portable computing device comprising a plurality of master processors.

33. The computer system of Claim 28, wherein the portable computing device comprises at least one of: a cellular telephone, a satellite telephone, a pager, a personal digital assistant (PDA), a smartphone, a navigation device, a smartbook or reader, a media player, and a laptop computer with a wireless connection.

34. A computer program product comprising a computer usable non-transitory medium having a computer readable program code embodied therein, said computer readable program code adapted to be executed to implement a method for managing application states of a portable computing device having at least one processor and a plurality of processor resources, said method comprising:

maintaining a first resource state set, a second resource state set, and a third resource state set in memory;

updating the first resource state set, the second resource state set, and the third resource state set based on prior usage of the portable computing device;

issuing a request for a processor operating in a first application state corresponding to the first resource state set to transition from the first application state to a second application state corresponding to the second resource state set;

reviewing a trigger set to determine if a processor condition matches one or more conditions listed in the trigger set;

if the processor condition matches one or more conditions listed in the trigger set, then switching states of one or more resources from states indicated by the first resource state set to states indicated by the second resource state set;

issuing a request for the processor operating in the second application state corresponding to the second resource state set to transition from the second application state to a third application state corresponding to the third resource state set;

reviewing the trigger set to determine if a processor condition matches one or more conditions listed in the trigger set; and

if the processor condition matches one or more conditions listed in the trigger set, then switching states of one or more resources from states indicated by the second resource state set states to states indicated by the third resource state set.

35. The computer program product of Claim 34, wherein:

the first resource state set is an active resource state set corresponding to a first active application state of the processor;

the second resource state set is a sleep resource state set corresponding to a sleep application state of the processor; and

the third resource state set is a next active resource state set corresponding to a next active application state of the processor.

36. The computer program product of Claim 34, wherein the steps of reviewing the trigger set are performed by a controller.

37. The computer program product of Claim 34, wherein the step of maintaining a first resource state set, a second resource state set, and a third resource state in memory comprises maintaining a first, second and third resource set in memory for each master processor of a portable computing device comprising a plurality of master processors.

38. The computer program product of Claim 34, wherein the portable computing device comprises at least one of: a cellular telephone, a satellite telephone, a pager, a personal digital assistant (PDA), a smartphone, a navigation device, a smartbook or reader, a media player, and a laptop computer with a wireless connection.

39. A computer program product comprising a computer usable non-transitory medium having a computer readable program code embodied therein, said computer readable program code adapted to be executed to implement a method for managing application states of a portable computing device having at least one processor and a plurality of processor resources, said method comprising:

maintaining a first memory buffer and a second memory buffer of a controller of the portable computing device;

updating a first resource state set and a second resource state set by a processor operating in a first application state storing updates for the first resource state set in the first memory buffer and storing updates for the second resource state set in the second memory buffer, the updates of the first resource state set and second resource state set being based on prior usage of the portable computing device;

accessing contents of the first and second memory buffers through the controller in response to the updates while the processor is operating in the first application state;

issuing a request for the processor operating in the first application state corresponding to the first resource state set to transition from the first application state to a second application state corresponding to the second resource state set;

the processor storing updates for a third resource state set in the first memory buffer after the request to transition is issued but before the controller is enabled to access the updates in the first memory buffer;

the controller reviewing a trigger set to determine if a processor condition matches one or more conditions listed in the trigger set;

if the processor condition matches one or more conditions listed in the trigger set, then the controller switching states of one or more resources from states indicated by the first resource state set to states indicated by the second resource state set; and

accessing the contents of the first memory buffer through the controller in response to the processor transitioning from the first application state to the second application state.

40. The computer program product of Claim 39, wherein:

the updating step comprises the processor issuing an interrupt contemporaneously with the updates to indicate that information has been stored in a memory buffer, and the controller accessing contents of the first and second memory buffers in response to the interrupt; and

the processor storing updates for the third resource state set comprises the processor storing the updates in the first memory buffer without issuing the interrupt contemporaneously with the updates.

41. The computer program product of Claim 39, wherein the controller does not have sufficient memory to simultaneously store the first, second and third resource sets.

42. The computer program product of Claim 39, wherein:

the first resource state set is an active resource state set corresponding to a first active application state of the processor;

the second resource state set is a sleep resource state set corresponding to a sleep application state of the processor; and

the third resource state set is another active resource state set corresponding to a next active application state of the processor.

43. The computer program product of Claim 39, wherein the step of maintaining a first memory buffer and a second memory buffer of a controller of the portable computing device comprises maintaining a resource set for only one processor at a time in each of the first and second memory buffers of a portable computing device comprising a plurality of master processors.

44. The computer program product of Claim 39, wherein the portable computing device comprises at least one of: a cellular telephone, a satellite telephone, a pager, a personal digital assistant (PDA), a smartphone, a navigation device, a smartbook or reader, a media player, and a laptop computer with a wireless connection.

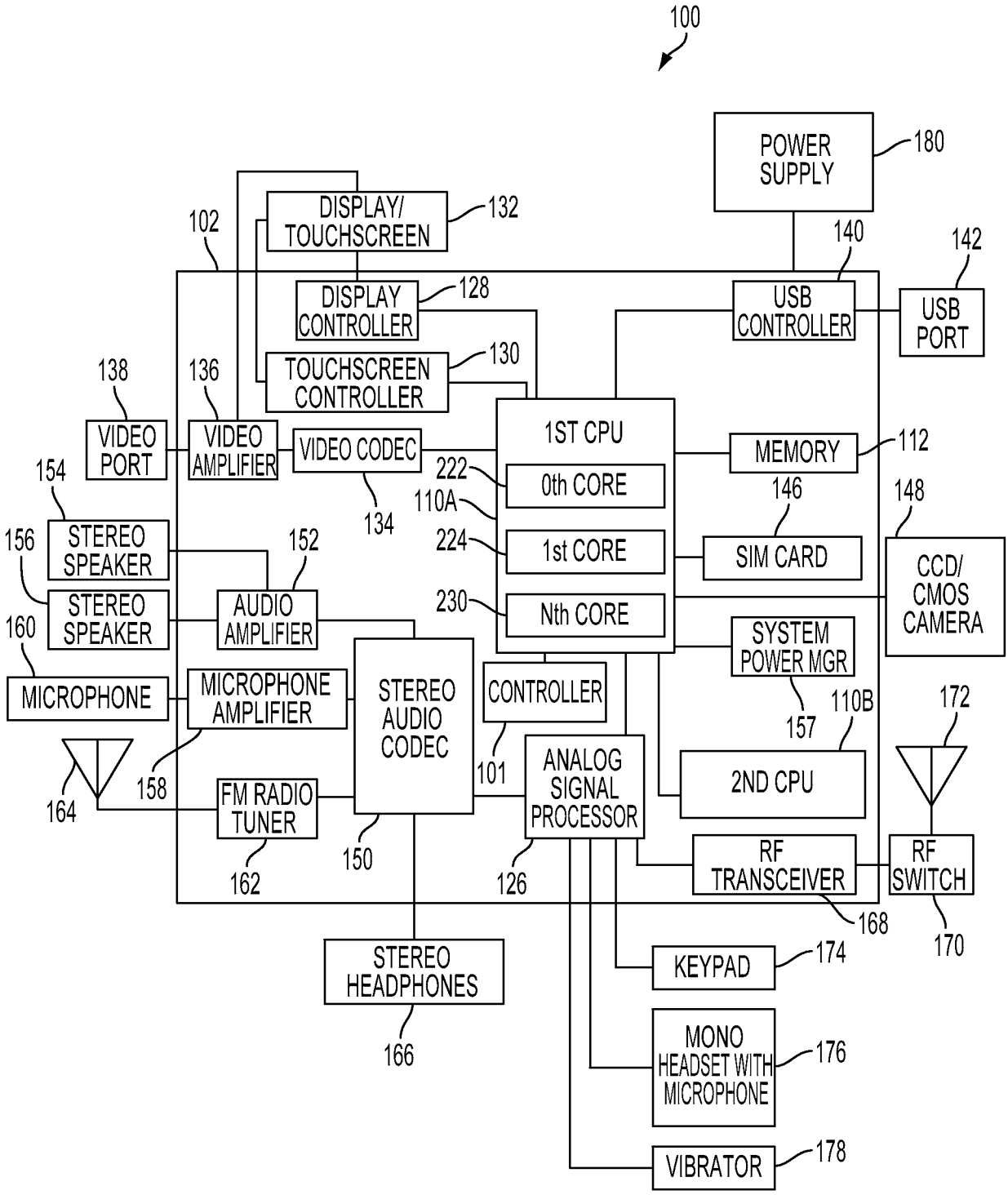
100
POWER SUPPLY
180
DISPLAY/ TOUCHSCREEN
132
102
140
142
DISPLAY CONTROLLER
128
USB CONTROLLER
USB PORT
TOUCHSCREEN CONTROLLER
130
138
136
VIDEO PORT
VIDEO AMPLIFIER
VIDEO CODEC
1ST CPU
MEMORY
112
222
0th CORE
146
148
154
110A
134
224
1st CORE
SIM CARD
CCD/ CMOS CAMERA
156
STEREO SPEAKER
152
230
Nth CORE
STEREO SPEAKER
AUDIO AMPLIFIER
160
SYSTEM POWER MGR
MICROPHONE
MICROPHONE AMPLIFIER
STEREO AUDIO CODEC
CONTROLLER
157
110B
172
101
ANALOG SIGNAL PROCESSOR
2ND CPU
164
158
FM RADIO TUNER
RF TRANSCEIVER
RF SWITCH
150
126
162
168
170
STEREO HEADPHONES
KEYPAD
174
166
MONO HEADSET WITH MICROPHONE
176
VIBRATOR
178

FIG. 1

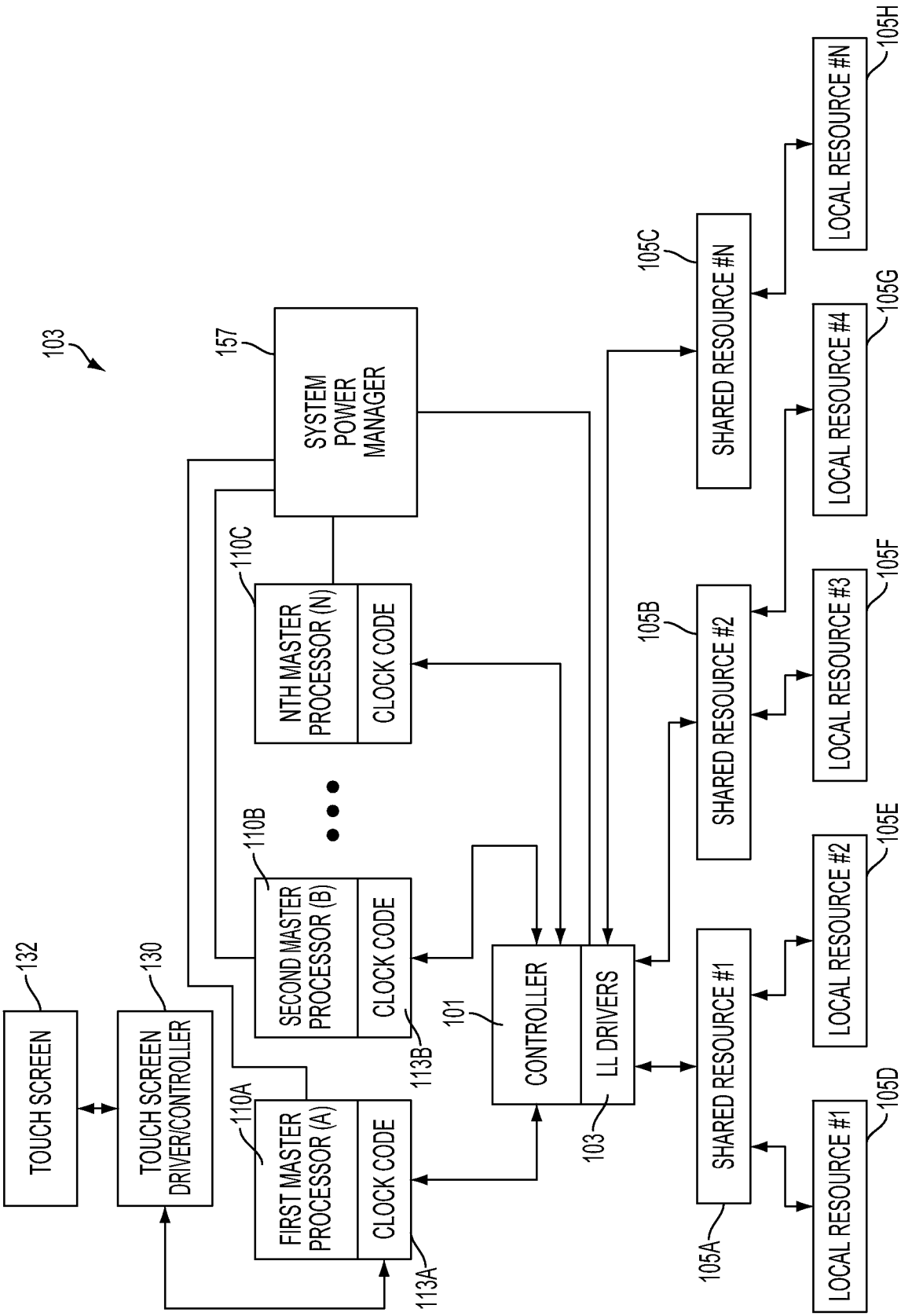
103
157
SYSTEM POWER MANAGER
110C
NTH MASTER PROCESSOR (N)
CLOCK CODE
110B
SECOND MASTER PROCESSOR (B)
CLOCK CODE
113B
110A
FIRST MASTER PROCESSOR (A)
CLOCK CODE
113A
132
TOUCH SCREEN
130
TOUCH SCREEN DRIVER/CONTROLLER
101
CONTROLLER
LL DRIVERS
103
105C
SHARED RESOURCE #N
105B
SHARED RESOURCE #2
SHARED RESOURCE #1
105A
LOCAL RESOURCE #N
105H
LOCAL RESOURCE #4
105G
LOCAL RESOURCE #3
105F
LOCAL RESOURCE #2
105E
LOCAL RESOURCE #1
105D

FIG. 2

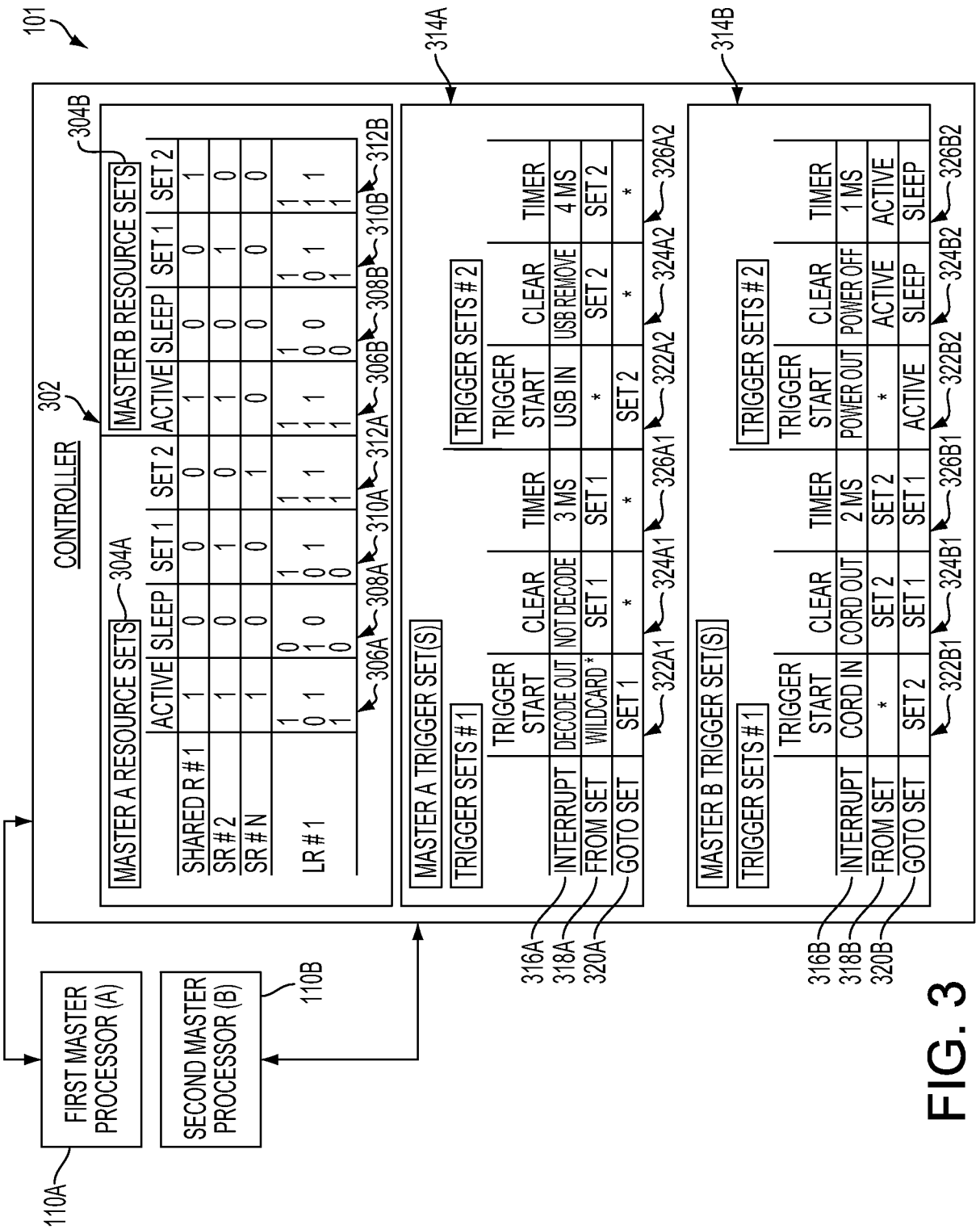
101
CONTROLLER
302
304A
MASTER A RESOURCE SETS
ACTIVE SLEEP SET 1 SET 2
SHARED R # 1
SR # 2
SR # N
LR # 1
306A
308A
310A
312A
304B
MASTER B RESOURCE SETS
ACTIVE SLEEP SET 1 SET 2
306B
308B
310B
312B
314A
MASTER A TRIGGER SET(S)
TRIGGER SETS # 1
TRIGGER START CLEAR TIMER
INTERRUPT DECODE OUT NOT DECODE 3 MS
FROM SET WILDCARD * SET 1 SET 1
GOTO SET SET 1 * *
322A1 324A1 326A1
TRIGGER SETS # 2
TRIGGER START CLEAR TIMER
USB IN USB REMOVE 4 MS
* SET 2 SET 2
SET 2 * *
322A2 324A2 326A2
314B
MASTER B TRIGGER SET(S)
TRIGGER SETS # 1
TRIGGER START CLEAR TIMER
INTERRUPT CORD IN CORD OUT 2 MS
FROM SET * SET 2 SET 2
GOTO SET SET 2 SET 1 SET 1
322B1 324B1 326B1
TRIGGER SETS # 2
TRIGGER START CLEAR TIMER
POWER OUT POWER OFF 1 MS
* ACTIVE ACTIVE
ACTIVE SLEEP SLEEP
322B2 324B2 326B2
316A
318A
320A
316B
318B
320B
FIRST MASTER PROCESSOR (A)
110A
SECOND MASTER PROCESSOR (B)
110B

FIG. 3

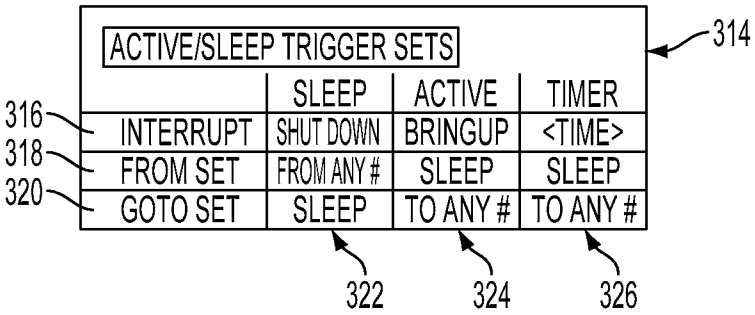
ACTIVE/SLEEP TRIGGER SETS
314
SLEEP
ACTIVE
TIMER
316
INTERRUPT
SHUT DOWN
BRINGUP
<TIME>
318
FROM SET
FROM ANY #
SLEEP
SLEEP
320
GOTO SET
SLEEP
TO ANY #
TO ANY #
322
324
326

FIG. 4

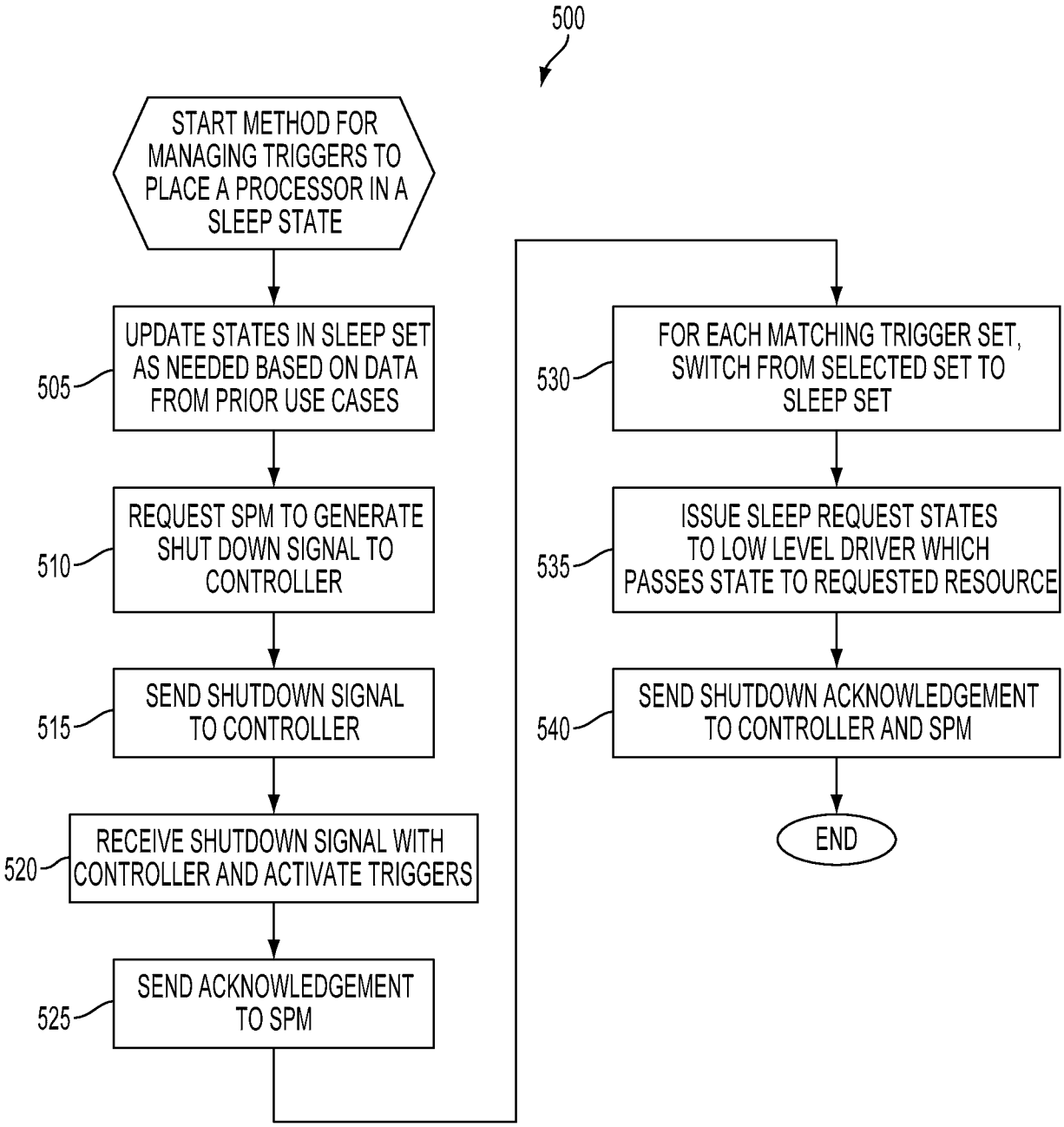
500
START METHOD FOR MANAGING TRIGGERS TO PLACE A PROCESSOR IN A SLEEP STATE
505
UPDATE STATES IN SLEEP SET AS NEEDED BASED ON DATA FROM PRIOR USE CASES
510
REQUEST SPM TO GENERATE SHUT DOWN SIGNAL TO CONTROLLER
515
SEND SHUTDOWN SIGNAL TO CONTROLLER
520
RECEIVE SHUTDOWN SIGNAL WITH CONTROLLER AND ACTIVATE TRIGGERS
525
SEND ACKNOWLEDGEMENT TO SPM
530
FOR EACH MATCHING TRIGGER SET, SWITCH FROM SELECTED SET TO SLEEP SET
535
ISSUE SLEEP REQUEST STATES TO LOW LEVEL DRIVER WHICH PASSES STATE TO REQUESTED RESOURCE
540
SEND SHUTDOWN ACKNOWLEDGEMENT TO CONTROLLER AND SPM
END

FIG. 5

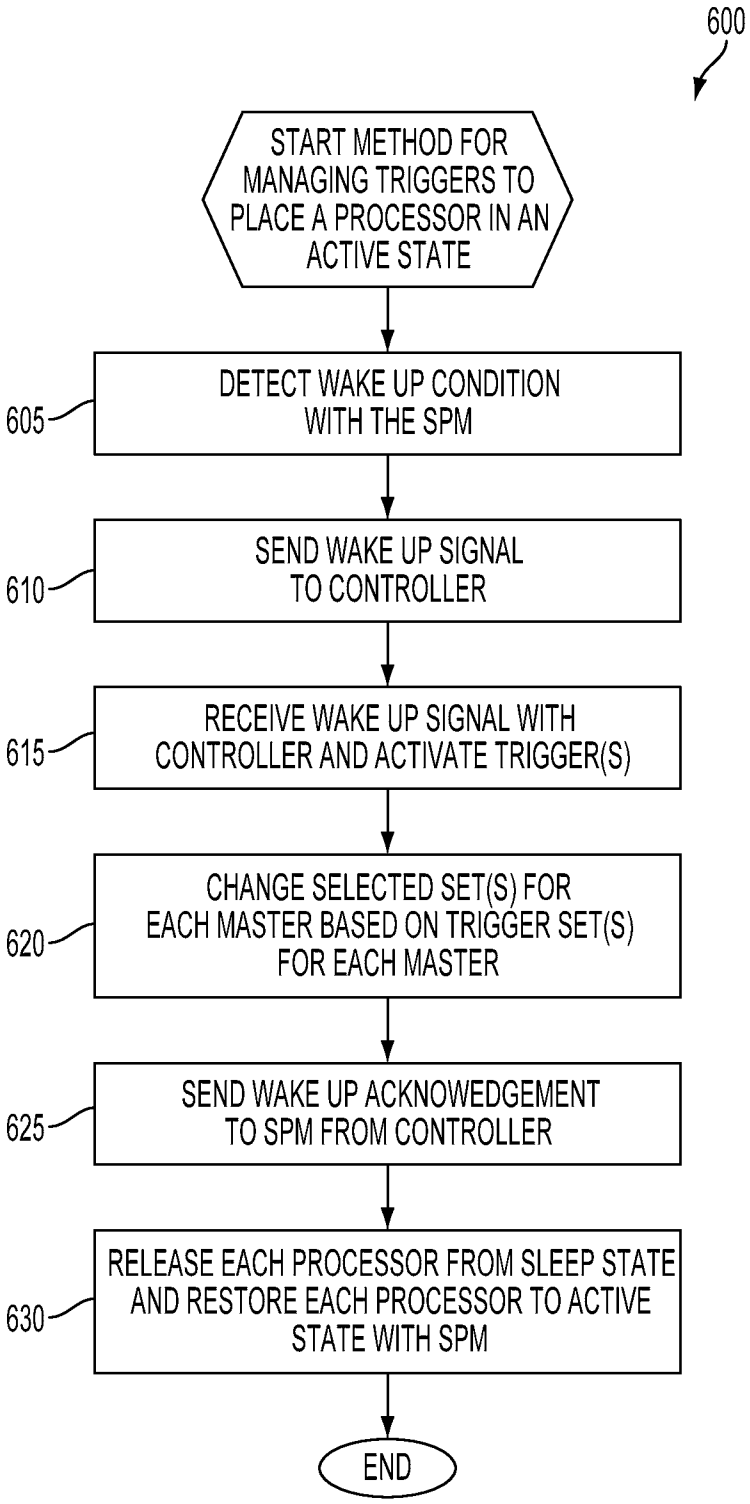
600
START METHOD FOR MANAGING TRIGGERS TO PLACE A PROCESSOR IN AN ACTIVE STATE
605
DETECT WAKE UP CONDITION WITH THE SPM
610
SEND WAKE UP SIGNAL TO CONTROLLER
615
RECEIVE WAKE UP SIGNAL WITH CONTROLLER AND ACTIVATE TRIGGER(S)
620
CHANGE SELECTED SET(S) FOR EACH MASTER BASED ON TRIGGER SET(S) FOR EACH MASTER
625
SEND WAKE UP ACKNOWEDGEMENT TO SPM FROM CONTROLLER
630
RELEASE EACH PROCESSOR FROM SLEEP STATE AND RESTORE EACH PROCESSOR TO ACTIVE STATE WITH SPM
END

FIG. 6

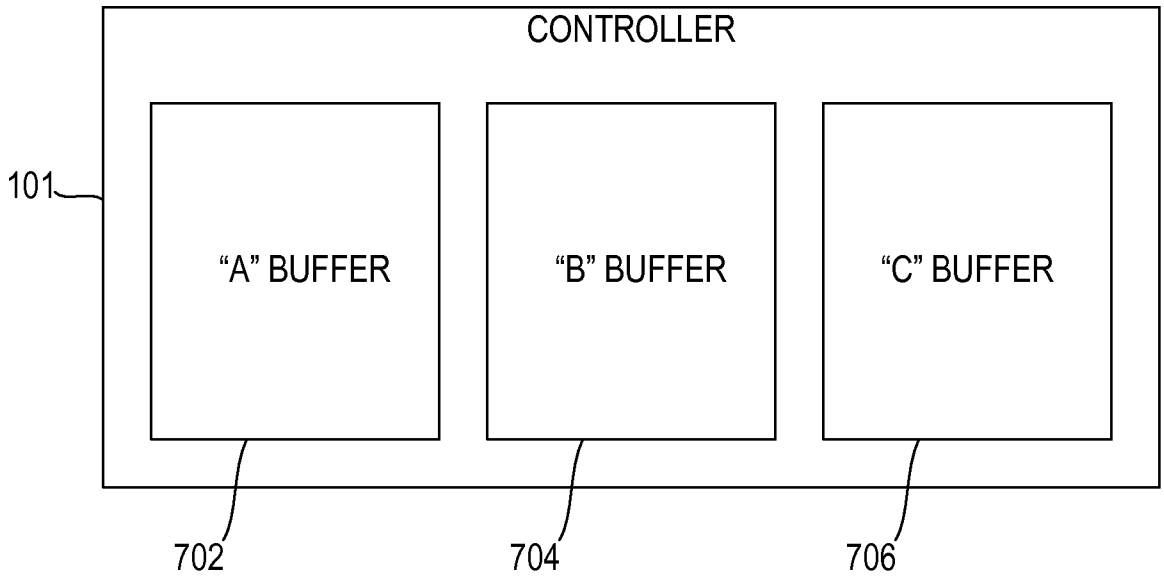

FIG. 7

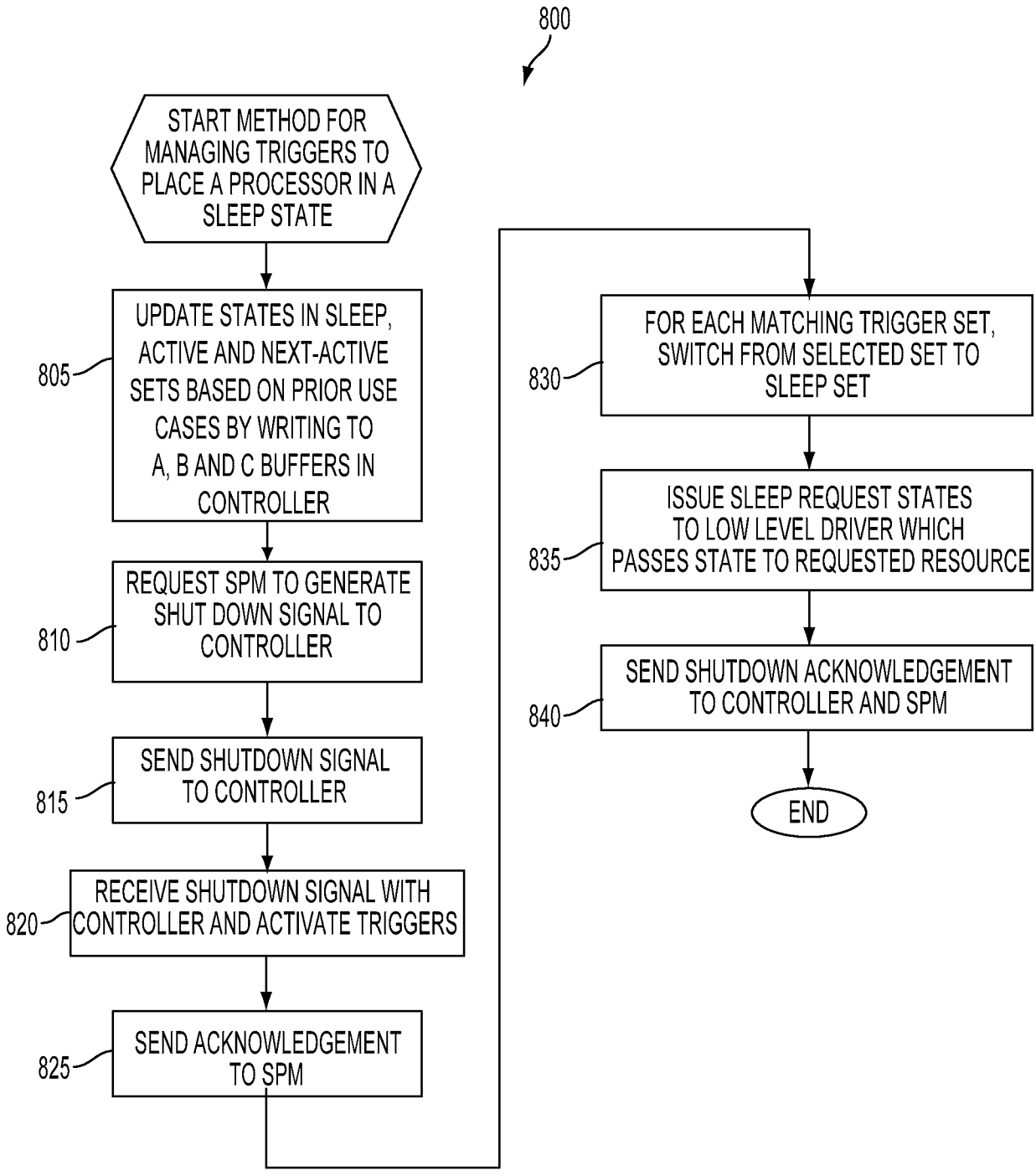
800
START METHOD FOR MANAGING TRIGGERS TO PLACE A PROCESSOR IN A SLEEP STATE
805
UPDATE STATES IN SLEEP, ACTIVE AND NEXT-ACTIVE SETS BASED ON PRIOR USE CASES BY WRITING TO A, B AND C BUFFERS IN CONTROLLER
810
REQUEST SPM TO GENERATE SHUT DOWN SIGNAL TO CONTROLLER
815
SEND SHUTDOWN SIGNAL TO CONTROLLER
820
RECEIVE SHUTDOWN SIGNAL WITH CONTROLLER AND ACTIVATE TRIGGERS
825
SEND ACKNOWLEDGEMENT TO SPM
830
FOR EACH MATCHING TRIGGER SET, SWITCH FROM SELECTED SET TO SLEEP SET
835
ISSUE SLEEP REQUEST STATES TO LOW LEVEL DRIVER WHICH PASSES STATE TO REQUESTED RESOURCE
840
SEND SHUTDOWN ACKNOWLEDGEMENT TO CONTROLLER AND SPM
END

FIG. 8

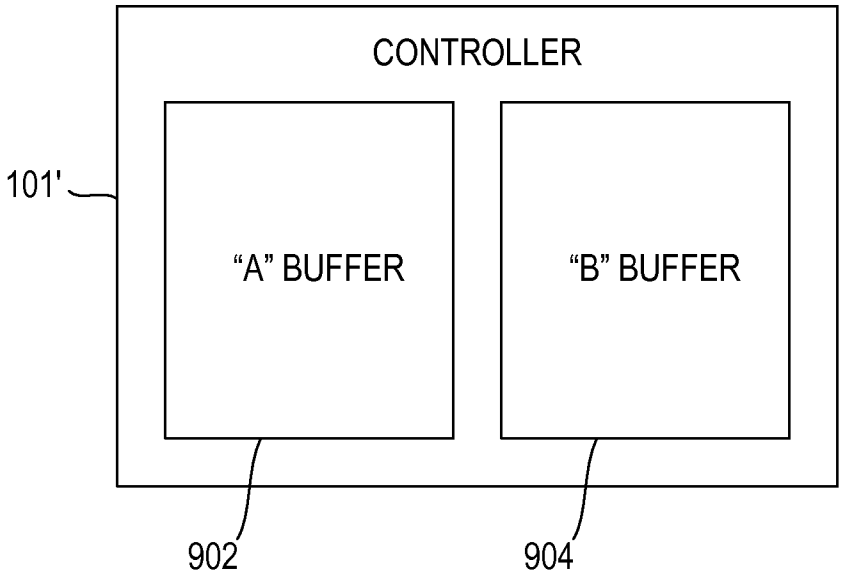

FIG. 9

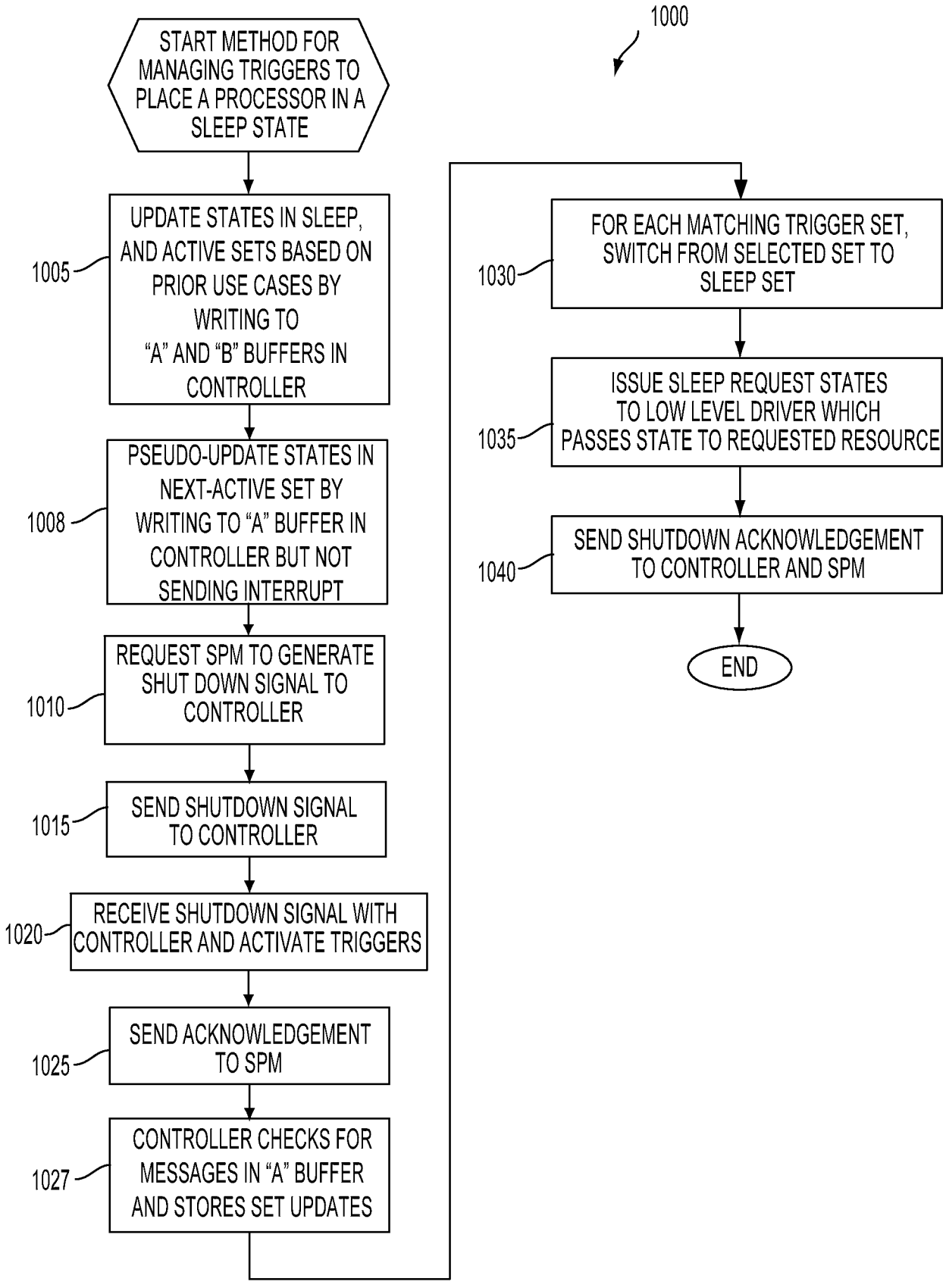
1000
START METHOD FOR MANAGING TRIGGERS TO PLACE A PROCESSOR IN A SLEEP STATE
1005
UPDATE STATES IN SLEEP, AND ACTIVE SETS BASED ON PRIOR USE CASES BY WRITING TO "A" AND "B" BUFFERS IN CONTROLLER
1008
PSEUDO-UPDATE STATES IN NEXT-ACTIVE SET BY WRITING TO "A" BUFFER IN CONTROLLER BUT NOT SENDING INTERRUPT
1010
REQUEST SPM TO GENERATE SHUT DOWN SIGNAL TO CONTROLLER
1015
SEND SHUTDOWN SIGNAL TO CONTROLLER
1020
RECEIVE SHUTDOWN SIGNAL WITH CONTROLLER AND ACTIVATE TRIGGERS
1025
SEND ACKNOWLEDGEMENT TO SPM
1027
CONTROLLER CHECKS FOR MESSAGES IN "A" BUFFER AND STORES SET UPDATES
1030
FOR EACH MATCHING TRIGGER SET, SWITCH FROM SELECTED SET TO SLEEP SET
1035
ISSUE SLEEP REQUEST STATES TO LOW LEVEL DRIVER WHICH PASSES STATE TO REQUESTED RESOURCE
1040
SEND SHUTDOWN ACKNOWLEDGEMENT TO CONTROLLER AND SPM
END

FIG. 10

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2011/062934

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F1/32 G06F9/50
ADD.

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 H04W

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 practical, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2009/049314 A1 (TAHA ALI [US] ET AL) 19 February 2009 (2009-02-19) paragraph [0022] - paragraph [0024] paragraph [0050] - paragraph [0073]	1-44
X	US 2010/316099 A1 (SUGITA TAKEHIRO [JP] ET AL) 16 December 2010 (2010-12-16) paragraph [0160] - paragraph [0163] paragraph [0214] - paragraph [0232]	1-44
A	US 2009/307519 A1 (HYATT EDWARD CRAIG [US]) 10 December 2009 (2009-12-10) the whole document	1-44

-/--

[X] Further documents are listed in the continuation of Box C.

[X] 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
17 April 2012	25/04/2012

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Dewyn, Torkild

1

Form PCT/ISA/210 (second sheet) (April 2005)

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	BENINI L ET AL: "A SURVEY OF DESIGN TECHNIQUES FOR SYSTEM-LEVEL DYNAMIC POWER MANAGEMENT", 1 June 2000 (2000-06-01), IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, IEEE SERVICE CENTER, PISCATAWAY, NJ, USA, PAGE(S) 299, XP008057349, ISSN: 1063-8210 the whole document -----	1-44

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2011/062934

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2009049314 A1	19-02-2009	NONE	
US 2010316099 A1	16-12-2010	CN 101925164 A	22-12-2010
		EP 2265061 A2	22-12-2010
		JP 2011003965 A	06-01-2011
		KR 20100135180 A	24-12-2010
		US 2010316099 A1	16-12-2010
US 2009307519 A1	10-12-2009	US 2009307519 A1	10-12-2009
		WO 2009148472 A2	10-12-2009