



(51) International Patent Classification:

G06F 15/80 (2006.01) G06F 1/32 (2006.01)
G06F 13/24 (2006.01)

(21) International Application Number:

PCT/US201 1/050580

(22) International Filing Date:

6 September 2011 (06.09.2011)

(25) Filing Language:

English

(26) Publication Language:

English

(71) Applicant (for all designated States except US): **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, MS: RNB-4-150, Santa Clara, CA 95052 (US).

(72) Inventors; and

(75) Inventors/Applicants (for US only): **HERDRICH, Andrew, J.** [US/US]; 2111 NE 25th Avenue, Hillsboro, OR 97124 (US). **ILLIKKAL, Rameshkumar, G.** [US/US]; 151 Red Ridge Ct., Folsom, CA 95630 (US). **IYER, Ravishankar** [US/US]; 15934 NW Andalusian Way, Portland, OR 97229 (US). **SRINIVASAN, Sadogopan** [IN/US]; 1470 NE Alex Way #353, Hillsboro, OR 97124 (US). **MOSES, Jaideep** [IN/US]; 13580 NW Stonebridge Drive,

Portland, OR 97229 (US). **MAKINENI, Srihari** [US/US]; 2111 NE 25th Ave., JF2-65, Hillsboro, OR 97124 (US).

(74) Agents: **TROP, Timothy, N.** et al; Trop, Pruner & HU, P.C., 1616 S. Voss Rd., Ste. 750, Houston, Texas 77057-2631 (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, 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, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM,

[Continued on nextpage]

(54) Title: POWER EFFICIENT PROCESSOR ARCHITECTURE

100

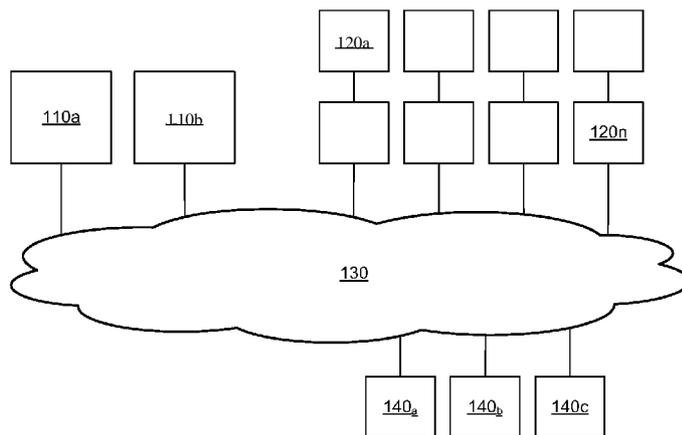


FIG. 1

(57) Abstract: In one embodiment, the present invention includes a method for receiving an interrupt from an accelerator, sending a resume signal directly to a small core responsive to the interrupt and providing a subset of an execution state of the large core to the first small core, and determining whether the small core can handle a request associated with the interrupt, and performing an operation corresponding to the request in the small core if the determination is in the affirmative, and otherwise providing the large core execution state and the resume signal to the large core. Other embodiments are described and claimed.

WO 2013/036222 A1

TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW,

Published:

ML, MR, NE, SN, TD, TG).

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

Declarations under Rule 4.17:

— *of inventorship (Rule 4.17(iv))*

POWER EFFICIENT PROCESSOR ARCHITECTURE

Background

Typically, a processor uses a power saving sleep mode such as in accordance with an Advanced Configuration and Power Interface (ACPI) standard (e.g., Rev. 3.0b, published
5 October 10, 2006) when possible. These so-called C-state core low power states (ACPI C-states) in addition to voltage and frequency scaling (DVFS or ACPI performance state (P-states)) can save power when a core is idle or not fully utilized. However, even in a multi-core processor context, a core is often woken from an efficient sleep state to perform a relatively simple operation, and is then returned to the sleep state. This operation can adversely affect power
10 efficiency, as there is a cost in both latency and power consumption for exiting and returning to low power states. During the state transition power may be consumed in some types of processors without useful work being accomplished, to the detriment of power efficiency.

Examples of operations to be handled upon exiting a low power state include keyboard inputs, timer interrupts, network interrupts and so on. To handle these operations in a power
15 sensitive manner, current operating systems (OSs) change program behavior by processing larger amounts of data at a time, or moving to a tickless OS where there are no periodic timer interrupts, and only sporadic programmed ones. Another strategy is to use timer coalescing, where multiple interrupts are grouped and handled at the same time. But in addition to changing a program's behavior, all of these options raise complexity and still can lead to power inefficient
20 operation. Further, some types of software (e.g., media playback) may make attempts to defeat hardware power efficiency mechanisms by requesting frequent, periodic wakes regardless of how much work needs to be accomplished. Thus, the tickless/timer coalescing strategies can save some power by reducing unnecessary wakes from deep C-states, but they require invasive changes to the OS and may take a significant amount of time to propagate through a computing
25 ecosystem, as such changes are not implemented until a new version of an operating system is distributed.

Brief Description of the Drawings

FIG. 1 is a block diagram of a processor in accordance with one embodiment of the present invention.

30 FIG. 2 is a block diagram of a processor in accordance with another embodiment of the present invention.

FIG. 3 is a flow diagram of resume flow options between cores in accordance with one embodiment of the present invention.

FIG. 4 is a flow diagram of a method in accordance with an embodiment of the present

invention.

FIG. 5 is a flow diagram of a method for transferring execution state in accordance with an embodiment of the present invention.

5 FIG. 6 is a block diagram of a processor in accordance with yet another embodiment of the present invention.

FIG. 7 is a block diagram of a processor in accordance with a still further embodiment of the present invention.

FIG. 8 is a block diagram of a processor in accordance with yet another embodiment of the present invention.

10 FIG. 9 is a timing diagram in accordance with an embodiment of the present invention.

FIG. 10 is a graphical illustration of power savings in accordance with an embodiment of the present invention.

FIG. 11 is a block diagram of a system in accordance with an embodiment of the present invention.

15 Detailed Description

In various embodiments, average power consumption can be reduced in a heterogeneous processor environment. This heterogeneous environment may include large fast cores and smaller more power-efficient cores that are combined for system and power efficiency reasons. Further still, embodiments may provide this power control in a manner that is transparent to an operating system (OS) executing on the processor. However, the scope of the present invention is not limited to heterogeneous environments, and can also be used in homogenous environments (from an OS-transparent but not necessarily hardware-heterogeneous perspective) to reduce average power (e.g., to keep as many cores asleep in a multiprocessor environment as possible). Embodiments may be especially suitable in hardware-accelerated environments such as tablet computer-based and system-on-chip (SoC) architectures where the cores are often asleep.

20 In general, embodiments may provide for power control by steering all wakeup signals to a smaller core rather than a larger core. In this way, it is possible to reduce average power by well over two times when the system is 95% idle. As will be described, in many embodiments this smaller core can be sequestered from the OS. That is, the presence of this smaller core is unknown to the OS, and this core is thus invisible to the OS. As such, embodiments can provide for power efficient processor operation via processor hardware in a manner that is transparent to the OS and applications executing on the processor.

30 Referring now to FIG. 1, shown is a block diagram of a processor in accordance with one embodiment of the present invention. As seen in FIG. 1, processor 100 may be a heterogeneous

processor having a number of large cores, small cores and accelerators. Although described herein in the context of a multi-core processor, understand embodiments are not so limited and in implementations may be within a SoC or other semiconductor-based processing devices. Note that the accelerators can perform work whether the processor cores are powered up or not, based
5 on a queue of input work. In the embodiment of FIG. 1, processor 100 includes a plurality of large cores. In the specific embodiment shown, two such cores 110a and 110b (generally, large cores 110) are shown, although understand that more than two such large cores may be provided. In various implementations, these large cores may be out-of-order processors having a relatively complex pipelined architecture and operating in accordance with a complex instruction set
10 computing (CISC) architecture.

In addition, processor 100 further includes a plurality of small cores 120a-120n (generally, small cores 120). Although 8 such cores are shown in the embodiment of FIG. 1, understand the scope of the present invention is not limited in this aspect. In various embodiments, small cores 120 may be power efficient in-order processors, e.g., to execute instructions according to a CISC
15 or a reduced instruction set computing (RISC) architecture. In some implementations, two or more of these cores may be coupled together in series to perform related processing, e.g., if several large cores are in power-saving states then one or more smaller cores may be active to perform work that would otherwise wake the large cores. In many embodiments, small cores 120 can be transparent to an OS, although in other embodiments the small and large cores may
20 be exposed to the OS, with configuration options available. In general, any core mix between large and small cores can be used in different embodiments. For example, a single small core can be provided per large core, or in other embodiments a single small core may be associated with multiple large cores.

As used herein, the term "large core" may be a processor core that is of a relatively
25 complex design and which may consume a relatively large amount of chip real estate as compared to a "small core," which may be of a lesser complexity design and consume a correspondingly smaller amount of chip real estate. In addition, the smaller cores are more power efficient than the larger cores, as they may have a smaller thermal design power (TDP) than the larger cores. However, understand that the smaller cores may be limited in their
30 processing capabilities as compared to the large cores. For example, these smaller cores may not handle all operations that are possible in the large cores. And in addition, it is possible that the smaller cores can be less efficient in instruction processing. That is, instructions may be performed more rapidly in the large cores than the small cores.

As further seen, both large cores 110 and small cores 120 may be coupled to an
35 interconnect 130. Different implementations of this interconnect structure can be realized in

different embodiments. For example, in some embodiments the interconnect structure can be according to a front side bus (FSB) architecture or an Intel® Quick Path Interconnect (QPI) protocol. In other embodiments, the interconnect structure can be according to a given system fabric.

5 Still referring to FIG. 1, multiple accelerators 140a-140c also may be coupled to interconnect 130. Although the scope of the present invention is not limited in this regard, the accelerators may include media processors such as audio and/or video processors, cryptographic processors, fixed function units and so forth. These accelerators may be designed by the same designers that designed the cores, or can be independent third party intellectual property (IP)
10 blocks incorporated into the processor. In general, dedicated processing tasks can be performed in these accelerators more efficiently than they can be performed on either the large cores or the small cores, whether in terms of performance or power consumption. Although shown with this particular implementation in the embodiment of FIG. 1, understand the scope of the present invention is not limited in this regard. For example, instead of having only two types of cores,
15 namely a large core and a small core, other embodiments may have multiple hierarchies of cores, including at least a large core, a medium core and a small core, with the medium core having a larger chip real estate than the small core but a smaller chip real estate than the large core and corresponding power consumption between that of the large core and the small core. In still other embodiments, the small core can be embedded within a larger core, e.g., as a subset of the
20 logic and structures of the larger core.

Furthermore, while shown in the embodiment of FIG. 1 as including multiple large cores and multiple small cores, it is possible that for certain implementations such as a mobile processor or SoC, only a single large core and a single small core may be provided. Specifically referring now to FIG. 2, shown is a block diagram of a processor in accordance with another
25 embodiment of the present invention in which processor 100" includes a single large core 110 and a single small core 120, along with interconnect 130 and accelerators 140a-c. As mentioned, this implementation may be suitable for mobile applications.

As example power figures for a typical large core, power consumption may be on the order of approximately 6000 milliwatts (mW), while for a medium core power consumption may be on the order of approximately 500 mW, and for a very small core power consumption may be on the order of approximately 15 mW. In an implementation that avoids waking the large core, significant power benefits may be achieved.
30

Embodiments allow the larger, less power-efficient cores to remain in low power sleep states longer than they otherwise would be able to. By steering interrupts and other core waking events to the smaller cores instead of the larger cores, the smaller cores may run longer and wake
35

more often, but this is still more power efficient than waking a large core to perform a trivial task such as data moving. Note that as described below for some operations, the large core may be powered on for execution, as for instance smaller cores might not support vector operations (e.g., AVX operations), complex addressing modes or floating point (FP) operations. In such cases a
5 wake signal could be re-routed from the small core to the large core.

For example, while performing hardware-accelerated 1080p video playback on a processor, over 1000 transitions into and out of core C6 state and nearly 1200 interrupts occur each second. If even a portion of these wake events are re-steered to a smaller core using an embodiment of the present invention, significant power savings can be achieved.

10 FIG. 3 summarizes resume flow options between cores in accordance with one embodiment of the present invention. As seen in FIG. 3, a software domain 210 and a hardware domain 220 are present. In general, software domain 210 corresponds to OS operations with regard to power management, e.g., according to an ACPI implementation. In general, the OS, based on its knowledge of upcoming tasks according to its scheduling mechanism, can select one
15 of multiple C-states to request the processor to enter into a low power mode. For example, an OS can issue an MWAIT call which includes a particular low-power state that is being requested.

In general, C0 corresponds to a normal operating state in which instructions are executed, while states C1-C3 are OS lower power states, each having a different level of power savings
20 and a corresponding different level of latency to return to the C0 state. As seen, depending on an expected workload of the processor, the OS may select a non-idle state, e.g., OS C0 or one of multiple idle states, e.g., OS C-states C1-C3. Each of these idle states can be mapped to a corresponding hardware low power state that is under control of processor hardware. Thus processor hardware can map a given OS C-state to a corresponding hardware C-state, which may
25 provide for greater power savings than that dictated by the OS. In general, lighter C-states (e.g., C1) save less power but have lower resume times than deeper C-states (e.g. C3). In various embodiments, hardware domain 220 and the mapping of OS C-states to processor C-states can be performed by a power control unit (PCU) of the processor, although the scope of the present invention is not limited in this regard. This mapping may be based on a prior history of OS-
30 based power management requests. Also, the decision can be based on a status of the overall system, configuration information and so forth.

In addition, the PCU or other processor logic may be configured to direct all wake events to a smallest available core (which may be an OS invisible core, in various embodiments). As seen in FIG. 3, upon exit from a given hardware-based idle state, control resumes directly to the
35 smallest available core such that the state is transferred to this smallest core. In contrast, in a

conventional hardware/software resumption, control returns only to the large core. Generally an OS selects a C-state based on the expected idle time and resume latency requirements, which the architecture maps to a hardware C-state. Thus as seen in the embodiment of FIG. 3 all resume signals (such as interrupts) are routed to the smallest available core, which determines whether it
5 can handle the resume operation, or instead is to send a wake signal to a larger core to continue. Note that embodiments do not interfere with existing P-states or C-state auto-demotion where the hardware selects a hardware C-state with lower resume latency automatically based on measured experimental efficiency. Note that it is also possible that the PCU or another programmable entity may examine incoming wake events to determine which core (large or small) to route
10 them to.

As described above, in some implementations, the small core itself can be hidden from the OS and application software. For example, a small-large core pair can be abstracted and hidden from application software. In a low power state all cores can be asleep while an accelerator (such as a video decode accelerator) performs a given task such as a decoding task. When the
15 accelerator runs out of data, it directs a wake signal to request additional data that can be from the small core, which wakes and determines that this simple data move operation can be accomplished without waking the large core, thus saving power. If a timer interrupt arrives and the small core wakes up and instead detects that a complex vector operation (like a 256-bit AVX instruction) exists in the instruction stream, the large core may be awakened to handle the
20 complex instruction (and other instructions in this stream) to enable reduced latency. In an alternate implementation a global hardware observation mechanism which can be located in the PCU or another uncore location near the PCU, or as a separate section of hardware logic on the global interconnect, or as an addition to the internal control logic of the small core, can detect that the small core encounters the AVX instruction and may generate an undefined instruction
25 fault, which may cause a shut down of the small core and re-steer the instruction stream to the larger core after waking it. Note that this behavior may extend beyond instructions to configuration or features. If the small core encounters a write to a configuration space that only exists on the large core, for instance, it may request a wake of the large core.

Referring now to FIG. 4, shown is a flow diagram of a method in accordance with an
30 embodiment of the present invention. Note that the method of FIG. 4 may be performed by various agents, depending upon a given implementation. For example, in some embodiments method 300 may be implemented in part by system agent circuitry within a processor such as a power control unit, which can be in a system agent or uncore portion of a processor. In other embodiments, method 300 may be implemented in part by interconnect logic such as power
35 control logic within an interconnect structure that can receive interrupts, e.g., from accelerators

coupled to the interconnect structure and forward the interrupts to a selected location.

As seen in FIG. 4, method 300 may begin by placing both large and small cores in a sleep state (block 310). That is, it is assumed that no active operations are being performed in the cores. As such, they can be placed in a selected low power state to reduce power consumption.

5 Although the cores may not be active, other agents within a processor or SoC such as one or more accelerators may be performing tasks. At block 320, an interrupt may be received from such an accelerator. This interrupt may be sent when the accelerator has completed a task, encountered an error, or when the accelerator needs additional data or other processing is to be performed by another component such as a given core. Control passes next to block 330 where

10 the logic can send a resume signal directly to the small core. That is, the logic may be programmed to always send a resume signal to the small core (or a selected one of multiple such small cores, depending upon system implementation) when both large and small cores are in a low power state. By sending interrupts directly and always to the small core, greater power consumption by the large core can be avoided for the many instances of interrupts for which the

15 small core can handle the requested operation. Note that certain types of filtering or caching mechanisms may be added to block 330 such that certain interrupt sources are always routed to one core or another, as desired to balance performance and power.

Referring still to FIG. 4, control next passes to diamond 340 where it can be determined whether the small core can handle a request associated with the interrupt. Although the scope of

20 the present invention is not limited in this regard, in some embodiments this determination may be done in the small core itself, after it is awoken. Or the logic that performs the method of FIG. 4 can perform the determination (and in which case it is possible for this analysis to be done prior to sending the resume signal to the small core).

As an example, the small core may determine whether it can handle the requested

25 operation based on performance requirements and/or instruction set architecture (ISA) capabilities of the small core. If the small core cannot handle a requested operation because it does not have ISA support, front end logic of the small core can parse a received instruction stream and determine that at least one instruction in the stream is not supported by the small core. Accordingly, the small core may issue an undefined instruction fault. This undefined fault

30 may be sent to the PCU (or another entity), which can analyze the fault and the state of the small core to determine whether the undefined fault is as a result of the small core not having hardware support for handling instruction, or if instead it is a true undefined fault. In the latter case, the undefined fault may be forwarded to an OS for further handling. If the fault is due to the small core not having the appropriate hardware support for handling the instruction, the PCU can cause

35 the execution state transferred to this small core to be transferred to a corresponding large core to

handle the requested instruction(s).

In other embodiments, a transfer of the execution state between small core and large core may occur when it is determined that the small core has been executing for too long a time or with too low a performance level. That is, assume that the small core has been executing for
5 many thousands or millions of processor cycles to perform requested tasks. Because of the more expedient execution available in the large core, it is possible that greater power reductions can occur by transferring the state to the large core to enable the large core to more rapidly conclude the task.

Still referring to FIG. 4 if it is determined that the requested operation can be handled in
10 the small core, control passes to block 350 where the operation is thus performed in the small core. For example, assume that the requested operation is a data move operation, the small core can perform the requested processing and if no other tasks are pending for the small core, it again can be placed into a low power state.

If instead it is determined at diamond 340 that the small core cannot handle the requested
15 operation, e.g., if the operation is a relatively complex operation that the small core is not configured to handle, control instead passes to block 360. There, a wakeup signal can be sent, e.g., directly from the small core to the large core, to cause the large core to be powered up. Accordingly, control passes to block 370 where the requested operation can thus be performed in the large core. Note that although described with this particular set of operations in the
20 embodiment of FIG. 4, understand the scope of the present invention is not limited in this regard.

Thus in various embodiments, a mechanism may be provided to allow hardware interrupts and other wake signals to be routed directly to the small core, without waking the large core. Note that in different implementations, the small core itself or a supervisory agent can determine whether the wake signal and processing can be completed without waking the large core. In
25 representative cases, the smaller core may be much more power efficient than the larger cores, and may as a result support only a subset of the instructions that the large core supports. And, many operations to be performed upon waking from a low power state can be offloaded to a simpler, more power-efficient core to avoid waking a larger more powerful core in heterogeneous environments (where many cores of various sizes are included in a system for
30 performance or power efficiency reasons).

Referring now to FIG. 5, shown is a flow diagram of a method for transferring execution state in accordance with an embodiment of the present invention. As shown in FIG. 5, method
380 may be performed by logic of a PCU, in one embodiment. This logic may be triggered responsive to a request to place a large core into a low power state. Responsive to such request,
35 method 380 may begin at block 382 where the execution state of the large core can be stored in a

temporary storage area. Note that this temporary storage area may be a dedicated state save area associated with the core or it can be within a shared cache such as a last level cache (LLC). Although the scope of the present invention is not limited in this regard, the execution state can include general-purpose registers, status and configuration registers, execution flags and so forth.

5 In addition, at this time additional operations to enable the large core to be placed into a low power state can be performed. Such operations include flushing of the internal caches and other state as well as signaling for shutdown of the given core.

Still referring to FIG. 5, it can then be determined whether the small core has resumed (diamond 384). This resumption may occur as a result of a resume signal received responsive to
10 an interrupt coming from, e.g., an accelerator of the processor. As part of the small core resumption, control passes to block 386 where at least a portion of the large core state can be extracted from the temporary storage area. More specifically, this extracted portion may be that portion of the large core's execution state that is to be used by the small core. As examples, this state portion may include the main register contents, various flags such as certain execution
15 flags, machine status registers and so forth. However, certain state may not be extracted, such as state associated with one or more execution units present in the large core that do not have corresponding execution units in the small core. This extracted portion of the state can then be sent to the small core (block 388), thus enabling the small core to perform whatever operations are appropriate responsive to the given interrupt. Although shown with this particular
20 implementation in the embodiment of FIG. 5, understand the scope of the present invention is not limited in this regard.

Referring now to FIG. 6, shown is a block diagram of a processor in accordance with an embodiment of the present invention. As shown in FIG. 6, processor 400 may be a multicore processor including a first plurality of cores 410i-410n that can be exposed to an OS, and a
25 second plurality of cores 410a-x that are transparent to the OS.

As seen, the various cores may be coupled via an interconnect 415 to a system agent or uncore 420 that includes various components. As seen, the uncore 420 may include a shared cache 430 which may be a last level cache. In addition, the uncore may include an integrated memory controller 440, various interfaces 450a-n, power control unit 455, and an advanced
30 programmable interrupt controller (APIC) 465.

PCU 450 may include various logic to enable power efficient operation in accordance with an embodiment of the present invention. As seen, PCU 450 can include wakeup logic 452 that can perform wakeups as described above. Thus logic 452 can be configured to always wake a small core first. However, this logic can be configured dynamically to not perform such small
35 core direct wakeups in certain circumstances. For example, a system can be dynamically

configured for power saving operations, e.g., when the system is a mobile system running on a battery. In such circumstances, the logic can be configured to always wake the small core. Instead, if the system is a server system, desktop or laptop system that is connected to wall power, embodiments may provide for a user-based selection to select latency and performance
5 over power savings. Thus wakeup logic 452 can be configured in such instances to wake up a large core rather than a small core responsive to an interrupt. Similar wakeups of the large core can be performed when it has been determined that a large number of small core wakeups result in a redirection to a large core.

To further enable power efficient operation, PCU 450 may further include a state transfer
10 logic 454 that can perform transfers of execution state between large and small cores. As discussed above, this logic may be used to take a large core's execution state stored into a temporary storage during a low power state, and extract at least a portion of that state to provide to a small core upon a small core wakeup.

Further still, PCU 450 may include an interrupt history storage 456. Such storage may
15 include a plurality of entries each identifying an interrupt that has occurred during system operation and whether the interrupt was successfully handled by the small core. Then based on this history, when a given interrupt is received, a corresponding entry of this storage can be accessed to determine whether a previous interrupt of the same type was successfully handled by the small core. If so, the PCU can direct the new incoming interrupt to the same small core.
20 Instead if it is determined based on this history that this type of interrupt was not successfully handled by small core (or with unsatisfactorily low performance), the interrupt can instead be sent to a large core.

Still referring to FIG. 6, PCU 450 may further include an undefined handling logic 458. Such logic may receive undefined faults issued by a small core. Based on this logic, information
25 in the small core can be accessed. Then it can be determined whether the undefined fault is as a result of a lack of support for the instruction in the small core or for another reason. Responsive to this determination, the logic can either cause the small core's state to be merged with the remaining part of the large core execution state (stored in a temporary storage area) and thereafter sent to the large core for handling of the interrupt, or send the undefined fault to an OS
30 for further handling. When it is determined that a small core cannot handle the interrupt, the portion of the execution state provided to the small core is thus taken from the small core and saved back to the temporary storage location and accordingly, the small core can be powered down. This merged state along with the remaining execution state of the large core can then be provided back to the large core to enable the large core to handle an interrupt that the small core
35 could not handle. Note also that an entry in interrupt history storage 456 can be written

responsive to such mishandling by the small core. Although shown with this particular logic in the embodiment of FIG. 6, understand the scope of the present invention is not limited in this regard. For example, the various logics of PCU 450 can be implemented in a single logic block in other embodiments.

5 APIC 465 may receive various interrupts, e.g., issued from accelerators and direct the interrupts as appropriate to a given one or more cores. In some embodiments, to maintain the small cores as hidden to the OS, APIC 465 may dynamically remap incoming interrupts, each of which may include an APIC identifier associated with it, from an APIC ID associated with a large core to an APIC ID associated with a small core.

10 With further reference to FIG. 6, processor 400 may communicate with a system memory 460, e.g., via a memory bus. In addition, by interfaces 450, connection can be made to various off-chip components such as peripheral devices, mass storage and so forth. While shown with this particular implementation in the embodiment of FIG. 6, the scope of the present invention is not limited in this regard.

15 Note that various architectures are possible to enable different coupling or integration of the large and small cores. As examples, the degree of coupling between these disparate cores can depend on a variety of engineering optimization parameters related to die area, power, performance and responsiveness.

Referring now to FIG. 7, shown is a block diagram of a processor in accordance with another embodiment of the present invention. As shown in FIG. 7, processor 500 may be a true heterogeneous processor including a large core 510 and a small core 520. As seen, each processor may be associated with its own private cache memory hierarchy, namely cache memories 515 and 525 which may include both level 1 and level 2 cache memories. In turn, the cores may be coupled together via a ring interconnect 530. Multiple accelerators 540a and 540b and a LLC, namely an L3 cache 550 which may be a shared cache are also coupled to the ring interconnect. In this implementation, execution state between the two cores may be transferred via ring interconnect 530. As described above, the execution state of the large core 500 can be stored in cache 550 prior to entry into a given low power state. Then upon wakeup of small core 520, at least a subset of this execution state can be provided to the small core to ready the core for execution of an operation that triggered its wakeup. Thus in the embodiment of FIG. 7, the cores are loosely coupled via this ring interconnect. Although shown for ease of illustration with a single large core and a single small core, understand the scope of the present invention is not limited in this regard. Using an implementation such as that of FIG. 7, any state or communication to be exchanged can be handled either via the ring architecture (which may also be a bus or fabric architecture). Or, in other embodiments this communication may be via a

dedicated bus between the two cores (not shown in FIG. 7).

Referring now to FIG. 8, shown is a block diagram of a processor in accordance with yet another embodiment of the present invention. As shown in FIG. 8, processor 500 may be a hybrid heterogeneous processor in which there is tight coupling or integration between the large and small cores. Specifically as seen in FIG. 8 large core 510 and small core 520 may share a shared cache memory 518, which in various embodiments may include both level 1 and level 2 caches. As such, execution state can be transferred from one of the cores to the other via this cache memory, thus avoiding the latency of communication via ring interconnect 530. Note that this arrangement allows for lower power due to reduced data movement overheads and faster communication between the cores, but may not be as flexible.

It should be noted that FIGS. 7 and 8 only illustrate two possible implementations (and only show limited numbers of cores). More implementation varieties are possible, including different arrangements of cores, a combination of the two schemes, more than two types of cores, etc. It is also possible that in a variant of FIG. 8 the two cores may share some components like execution units, an instruction pointer or a register file.

As discussed, embodiments can be completely transparent and invisible to the operating system, and thus no software changes and only minimal increases in resume time from C-states can be achieved. In other embodiments, the presence and availability of small cores can be exposed to the OS to thus enable the OS to make decisions whether to provide an interrupt to a small core or a large core. Furthermore, embodiments may provide mechanisms in system software such as a basic input output system (BIOS) to expose the large and small cores to the OS, or to configure whether the small cores are exposed or not. Embodiments may increase apparent resume times from C-states, but this is acceptable as current platforms vary in resume latencies, and currently no useful work is done during the time a core's state is being restored. The ratio of how different small and large cores are may vary from insignificant differences to major microarchitectural structural differences. According to various embodiments, the most primary differentiators between the heterogeneous cores may be the die area and power consumed by the cores.

In some implementations, a control mechanism may be provided such that if it is detected that the large core is woken most of the time upon resume, waking of the small core first may be bypassed, and the large core can be directly woken, at least for a predetermined period of time to preserve performance. Note that in some embodiments a mechanism to universally re-steer all interrupts and other wake signals to either the small or large core can be exposed to software, both system and user-level software, depending on the power and performance requirements of the application and system. As one such example, a user-level instruction may be provided to

perform the steering of wakeup operations to a specified core. Such instruction may be a variant of an MWAIT-like instruction.

In some embodiments, an accelerator can send a hint to the PCU or other management agent with an interrupt to indicate that the requested operation is a relatively simple operation
5 such that it can be handled effectively in the small core. This accelerator-provided hint may be used by the PCU to automatically direct incoming interrupts to the small core for handling.

Referring now to FIG. 9, shown is a timing diagram illustrating operations occurring in a large core 710 and a small core 720 in accordance with an embodiment of the present invention. As seen, a longer sleep duration for large core 710 can be enabled by allowing a device interrupt
10 to be provided to small core 720 directly, and determining in the small core whether it can handle the interrupt. If so, large core 710 can remain in a sleep state and the interrupt handled on small core 720.

Referring now to FIG. 10, shown is a graphical illustration of power savings in accordance with an embodiment of the present invention. As shown in FIG. 10, in a conventional system
15 that has transitions from an active CO state to a deep low power state, e.g., a C6 state, core power consumption of a large core can vary from a relatively high level, e.g. 500 mW during every entry into the CO states to a zero power consumption level in the C6 state (middle view). Instead in an embodiment of the present invention (bottom view), wakeups into a CO state can be directed away from the large core and to a small core and thus, rather than the 500 mW power
20 consumption level, the small cores can handle CO states at a much lower power level, e.g., 10 mW in the embodiment of FIG. 10.

Embodiments may be implemented in many different system types. Referring now to FIG. 11, shown is a block diagram of a system in accordance with an embodiment of the present invention. As shown in FIG. 11, multiprocessor system 600 is a point-to-point interconnect
25 system, and includes a first processor 670 and a second processor 680 coupled via a point-to-point interconnect 650. As shown in FIG. 11, each of processors 670 and 680 may be multicore processors, including first and second processor cores (i.e., processor cores 674a and 674b and processor cores 684a and 684b), although potentially many more cores may be present in the processors. More specifically, each of the processors can include a mix of large, small (and
30 possibly medium) cores, accelerators and so forth, in addition to logic to direct wakeups to the smallest available core, when at least the large cores are in a low power state, as described herein.

Still referring to FIG. 11, first processor 670 further includes a memory controller hub (MCH) 672 and point-to-point (P-P) interfaces 676 and 678. Similarly, second processor 680
35 includes a MCH 682 and P-P interfaces 686 and 688. As shown in FIG. 11, MCH's 672 and 682

couple the processors to respective memories, namely a memory 632 and a memory 634, which may be portions of system memory (e.g., DRAM) locally attached to the respective processors. First processor 670 and second processor 680 may be coupled to a chipset 690 via P-P interconnects 652 and 654, respectively. As shown in FIG. 11, chipset 690 includes P-P
5 interfaces 694 and 698.

Furthermore, chipset 690 includes an interface 692 to couple chipset 690 with a high performance graphics engine 638, by a P-P interconnect 639. In turn, chipset 690 may be coupled to a first bus 616 via an interface 696. As shown in FIG. 11, various input/output (I/O) devices 614 may be coupled to first bus 616, along with a bus bridge 618 which couples first bus
10 616 to a second bus 620. Various devices may be coupled to second bus 620 including, for example, a keyboard/mouse 622, communication devices 626 and a data storage unit 628 such as a disk drive or other mass storage device which may include code 630, in one embodiment. Further, an audio I/O 624 may be coupled to second bus 620. Embodiments can be incorporated into other types of systems including mobile devices such as a smart cellular telephone, tablet
15 computer, netbook, or so forth.

Embodiments may be implemented in code and may be stored on a non-transitory storage medium having stored thereon instructions which can be used to program a system to perform the instructions. The storage medium may include, but is not limited to, any type of disk including floppy disks, optical disks, solid state drives (SSDs), compact disk read-only memories
20 (CD-ROMs), compact disk rewritables (CD-RWs), and magneto-optical disks, semiconductor devices such as read-only memories (ROMs), random access memories (RAMs) such as dynamic random access memories (DRAMs), static random access memories (SRAMs), erasable programmable read-only memories (EPROMs), flash memories, electrically erasable programmable read-only memories (EEPROMs), magnetic or optical cards, or any other type of
25 media suitable for storing electronic instructions.

While the present invention has been described with respect to a limited number of embodiments, those skilled in the art will appreciate numerous modifications and variations therefrom. It is intended that the appended claims cover all such modifications and variations as fall within the true spirit and scope of this present invention.
30

What is claimed is:

1. An apparatus comprising:

a first core to execute instructions;

a second core to execute instructions, the second core heterogeneous to and smaller than

5 the first core; and

a logic to cause the second core and not the first core to be woken responsive to an interrupt when the first and second cores are in a low power state.

2. The apparatus of claim 1, wherein the logic is to always cause the second core

10 and not the first core to be woken responsive to the interrupt.

3. The apparatus of claim 1, wherein the logic is to provide a subset of an execution state of the first core to the second core responsive to the interrupt.

4. The apparatus of claim 3, wherein the second core is to determine whether the second core can handle the interrupt, and if not, to cause a wakeup signal to be sent the first core.

5. The apparatus of claim 4, wherein responsive to the determination that the second core cannot handle the interrupt, the logic is to obtain the subset of the execution state of the first core from the second core and to merge the execution state subset with a remainder of the execution state of the first core stored in a temporary storage area.

6. The apparatus of claim 1, wherein the apparatus comprises a multicore processor including the first and second cores and a power control unit (PCU), the PCU including the logic, the logic comprising:

a wakeup logic;

a state transfer logic;

an undefined handling logic; and

an interrupt history storage.

7. The apparatus of claim 1, further comprising an accelerator coupled to the logic, the accelerator to perform a task and to send the interrupt to the logic upon completion of the task.

8. The apparatus of claim 7, wherein the second core is to handle the interrupt when

the interrupt comprises a request for a data movement operation.

9. The apparatus of claim 7, wherein the second core is to cause a wakeup signal to
5 be sent to the first core to enable the first core to handle the interrupt when the interrupt
comprises a request for a vector operation.

10. The apparatus of claim 9, wherein the logic is to receive an undefined instruction
10 fault from the second core, determine that the second core cannot handle the vector operation,
obtain an execution state from the second core, merge the execution state with at least a portion
of an execution state of the first core stored in a temporary storage area, and cause the merged
execution state to be sent to the first core.

11. The apparatus of claim 1, wherein the logic is to analyze a plurality of interrupts
15 and if a majority of the plurality of interrupts are to be handled by the first core, the logic is to
not wake the second core responsive to the interrupt and instead wake the first core.

12. A method comprising:
receiving an interrupt from an accelerator in a logic of a processor including a first small
20 core, a first large core and the accelerator, when the first small core and the first large core are in
a low power state;
sending a resume signal directly to the first small core responsive to the interrupt and
providing a subset of an execution state of the first large core to the first small core; and
determining whether the first small core can handle a request associated with the
25 interrupt, and if so performing an operation corresponding to the request in the first small core.

13. The method of claim 12, further comprising if the first small core cannot handle
the request, obtaining the execution state subset from the first small core, merging the execution
state subset with a stored execution state of the first large core, and sending a wakeup signal and
30 the merged execution state to the large core.

14. The method of claim 13, further comprising thereafter performing the operation
corresponding to the request in the first large core.

35 15. The method of claim 12, further comprising receiving the interrupt with a hint to

indicate whether the interrupt should be directed to the first small core or the first large core.

16. The method of claim 12, further comprising accessing an entry of a table based on a type of the interrupt and determining whether to send the resume signal directly to the first
5 small core or the first large core based on the entry.

17. A system comprising:

a multicore processor including a first plurality of cores and a second plurality of cores, the second plurality of cores having lower thermal design power than the first plurality of cores,
10 an accelerator, and a power control unit (PCU), wherein the PCU is to receive an interrupt from the accelerator when the first plurality of cores and the second plurality of cores are in a low power state, send a resume signal directly to a first of the second plurality of cores responsive to the interrupt and provide a subset of an execution state of a first of the first plurality of cores to the first of the second plurality of cores; and

15 a dynamic random access memory (DRAM) coupled to the multicore processor.

18. The system of claim 17, wherein the first plurality of cores are of a heterogeneous design from the second plurality of cores.

19. The system of claim 17, wherein the second plurality of cores are transparent to
20 an operating system (OS).

20. The system of claim 17, wherein the PCU is to access an entry of a table using the interrupt to determine whether to send the resume signal to the first one of the first or second
25 plurality of cores, wherein the PCU is to send the resume signal to the first one of the first plurality of cores when the entry indicates that one of the second plurality of cores incurred an undefined fault responsive to a previous interrupt of the same type as the interrupt.

100

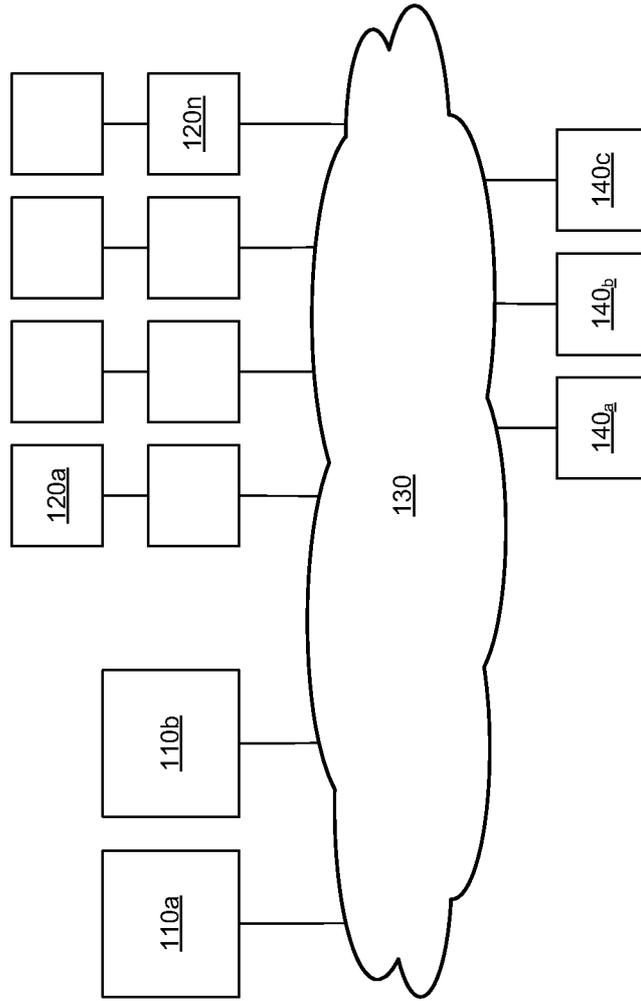


FIG. 1

100

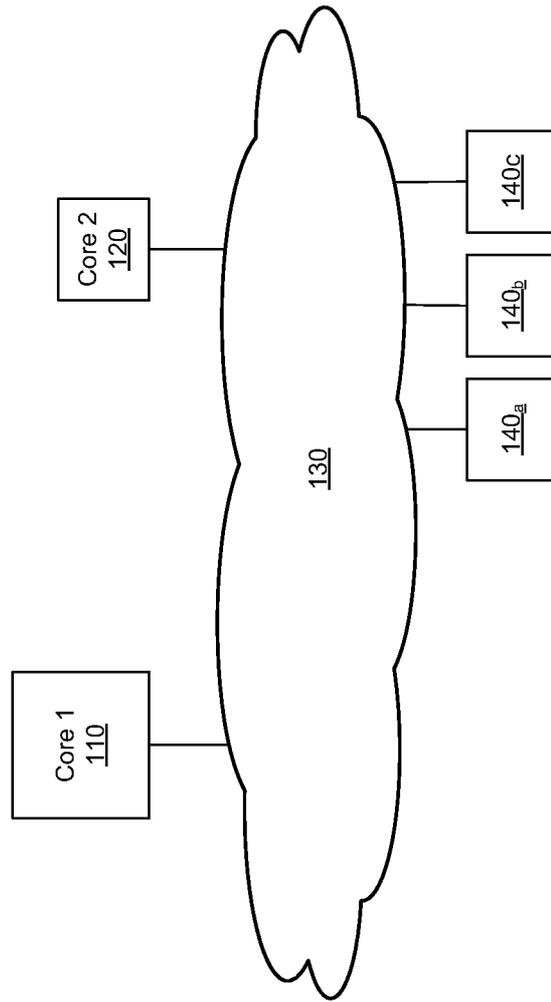


FIG. 2

200

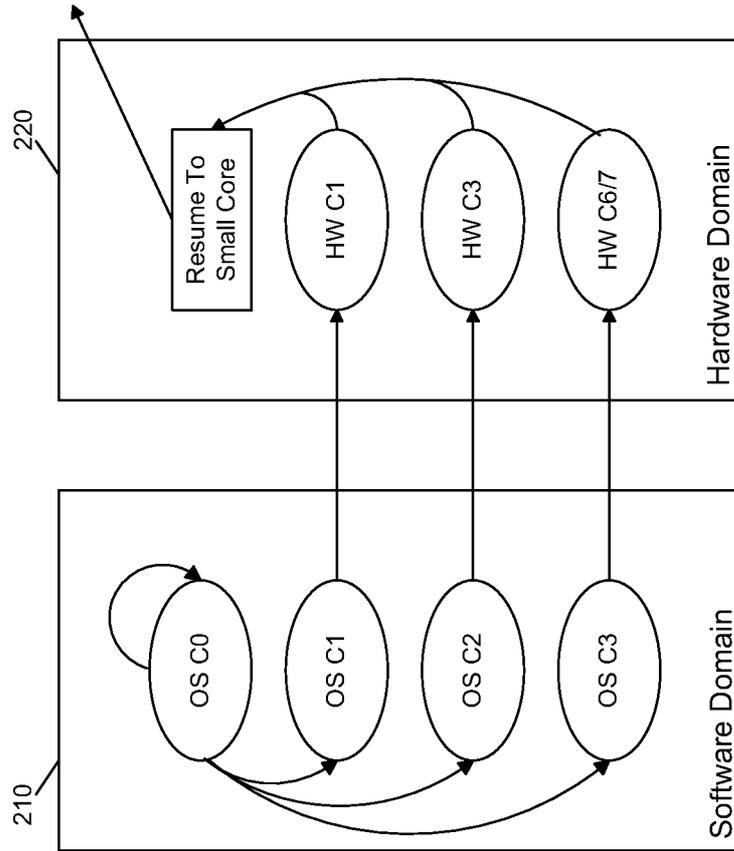


FIG. 3

300

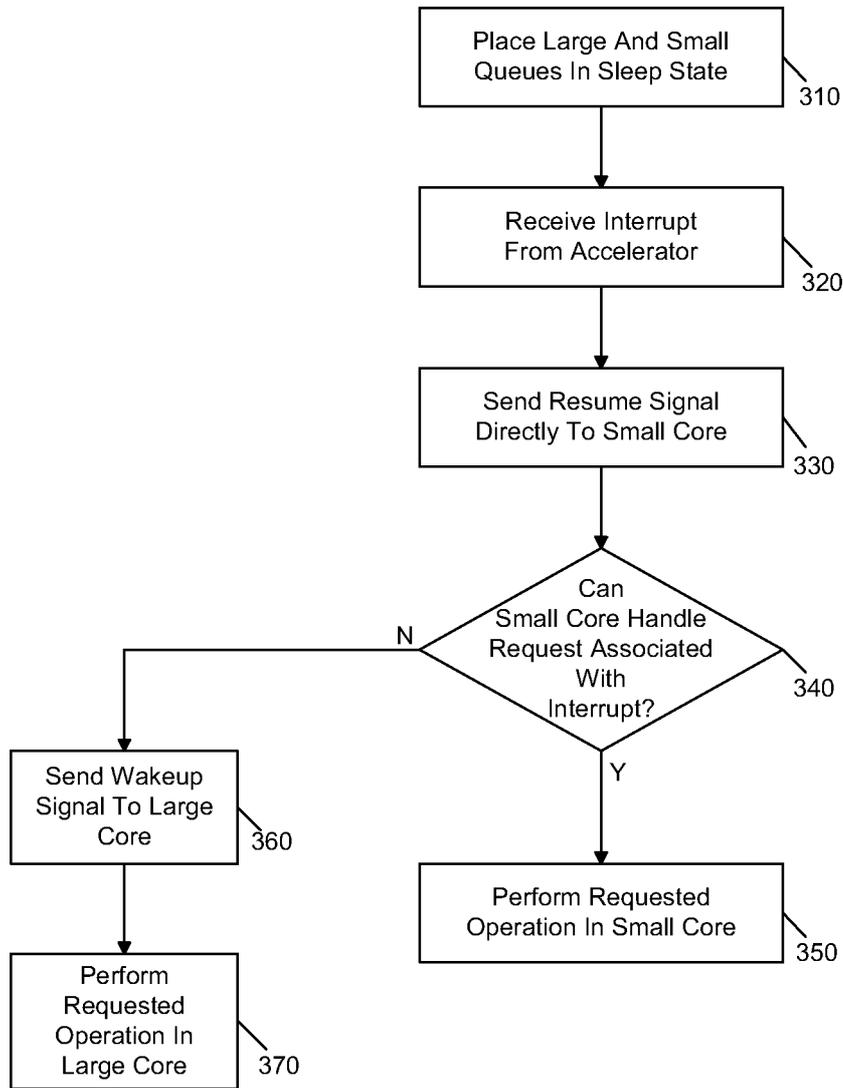


FIG. 4

380

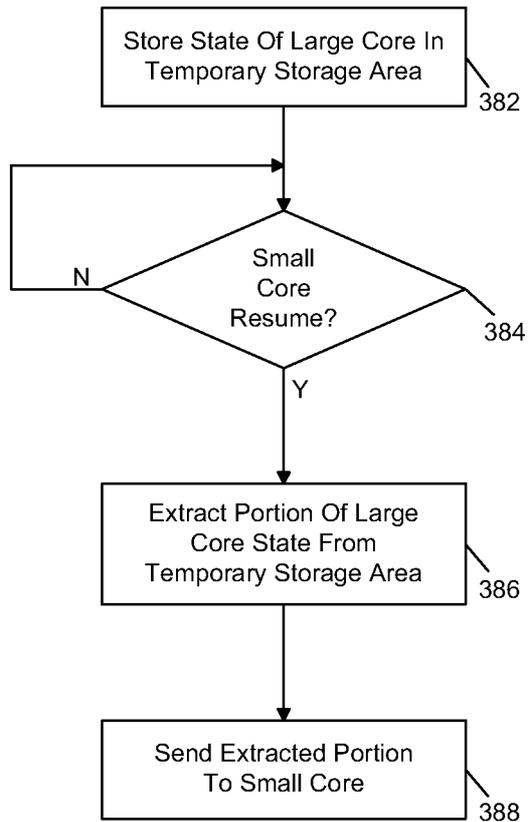


FIG. 5

400

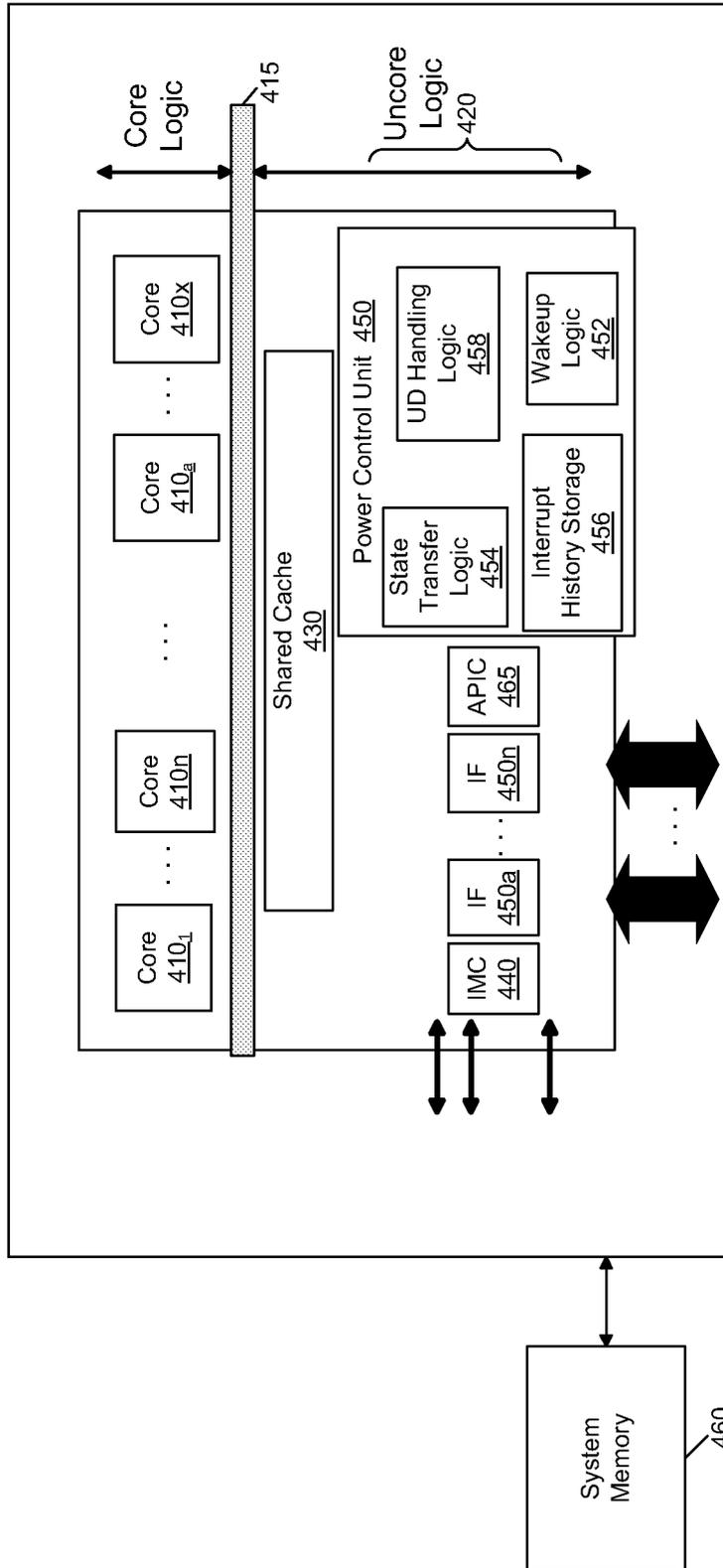


FIG. 6

500

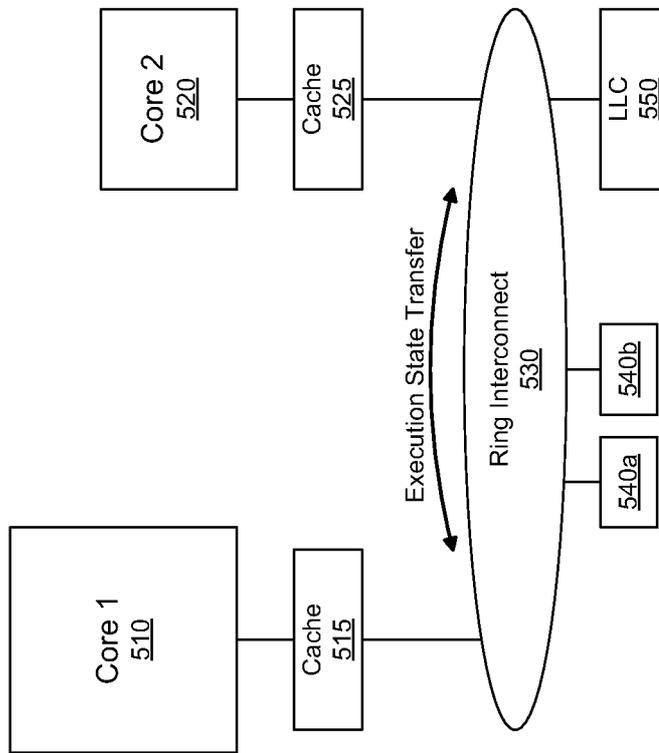


FIG. 7

500

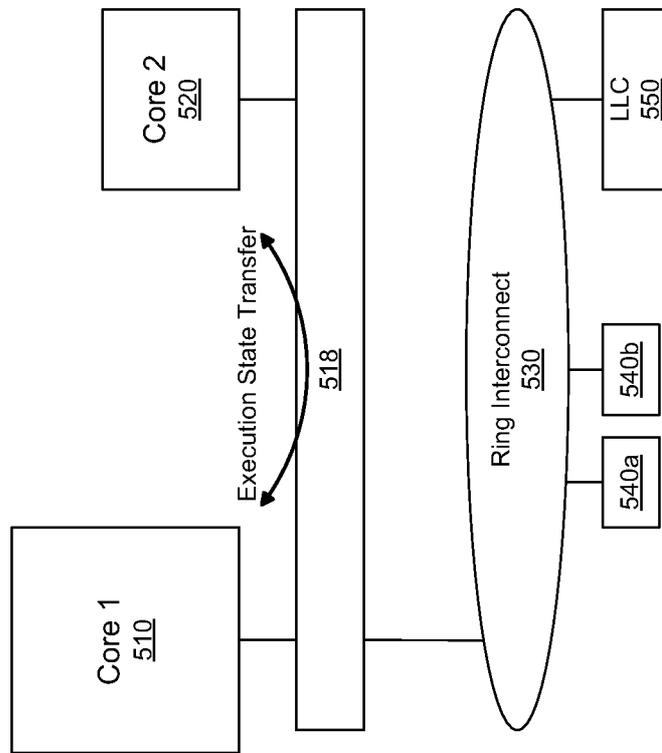


FIG. 8

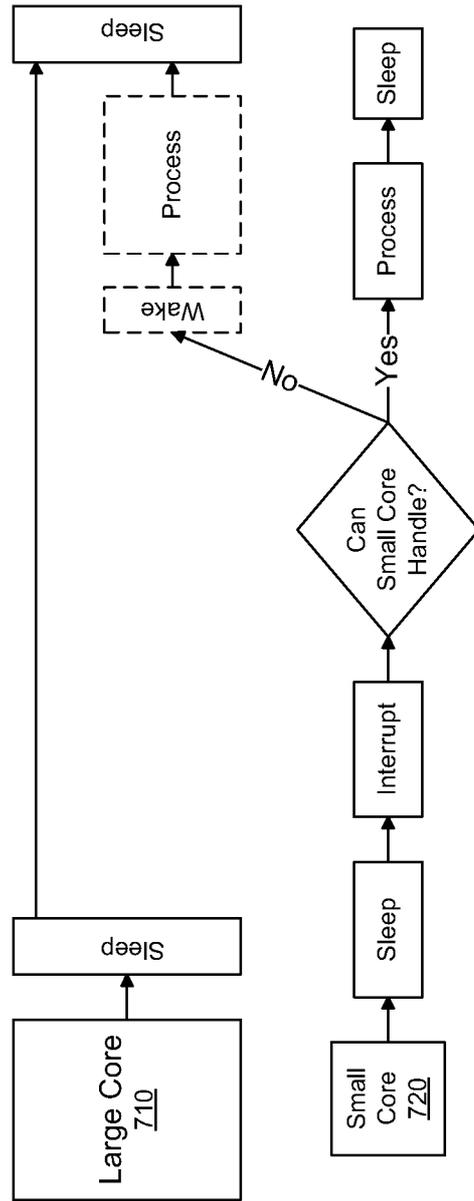


FIG. 9

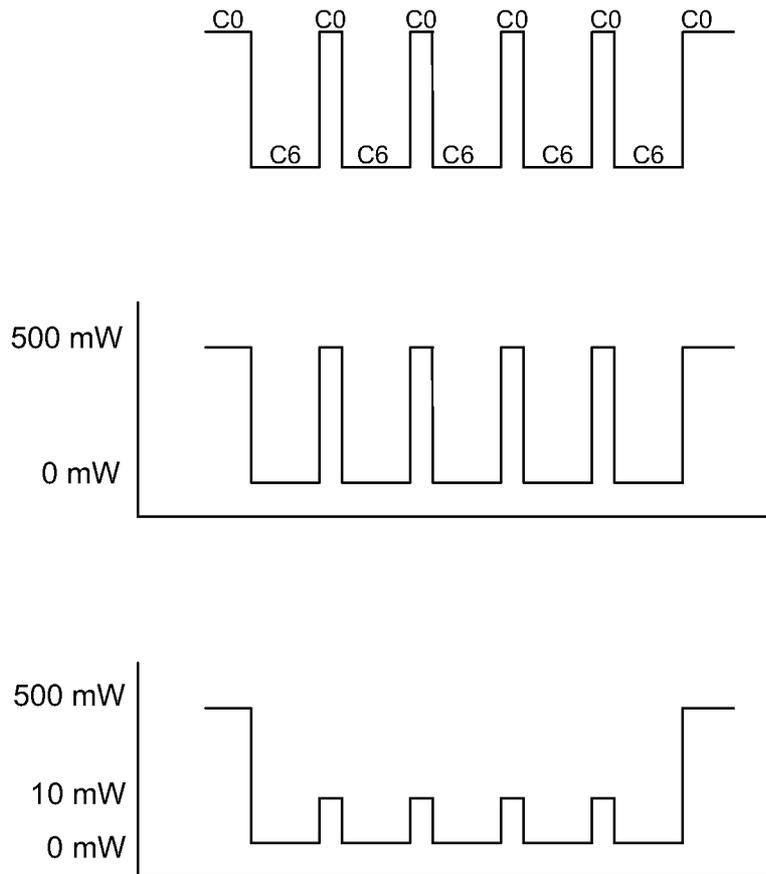


FIG. 10

600

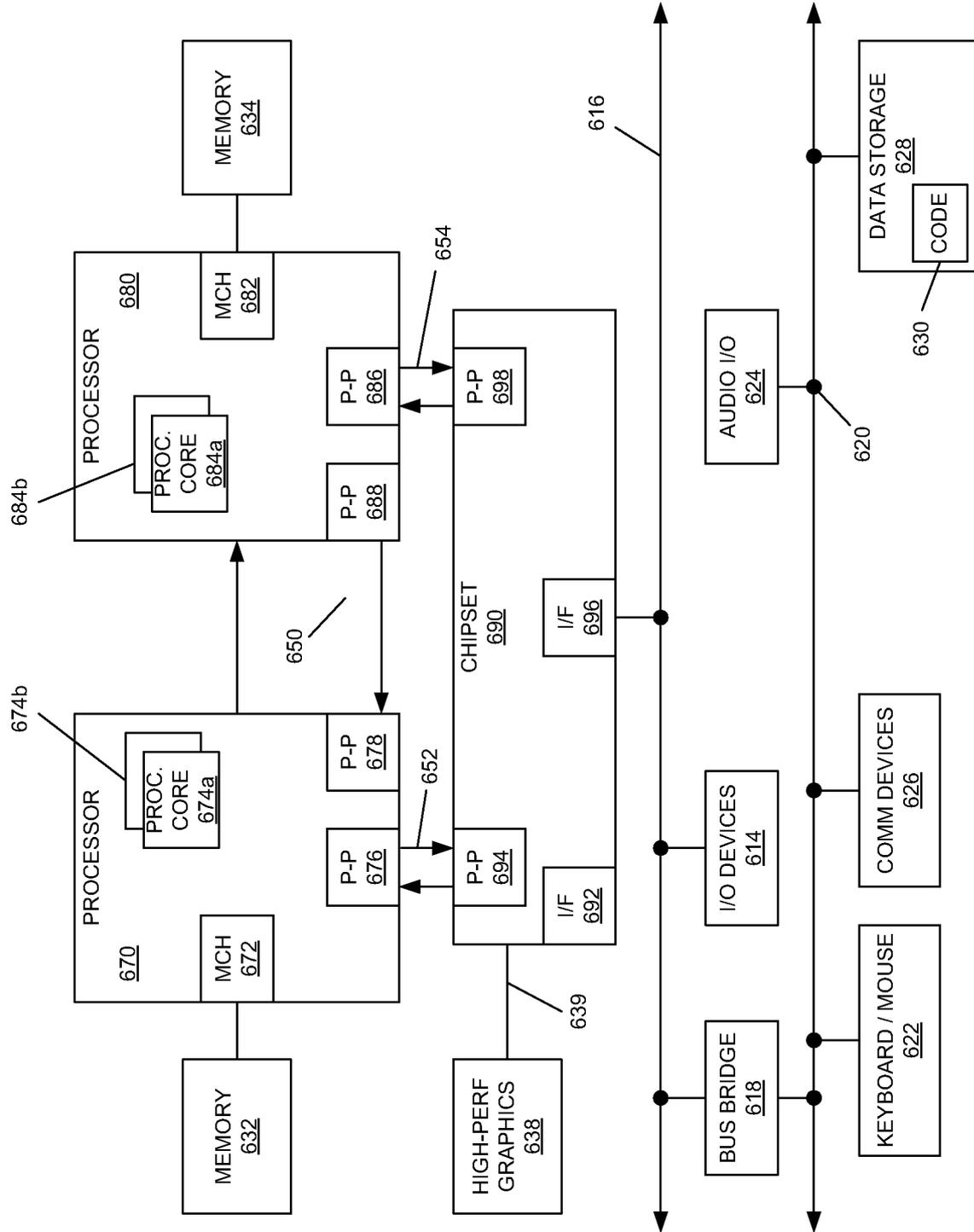


FIG. 11

A. CLASSIFICATION OF SUBJECT MATTER**G06F 15/80(2006.01)i, G06F 13/24(2006.01)I, G06F 1/32(2006.01)I**

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F 15/80; G06F 1/00; G06F 13/24

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

Korean utility models and applications for utility models

Japanese utility models and applications for utility models

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

eKOMPASS(KIPO internal) & Keywords: core, interrupt,

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2009-0319712 A1 (BHARADWAJ PUDIPEDDI et al.) 24 December 2009 See the abstract; claim 1 and figures 1-2.	1-20
A	US 2007-0101174 A1 (AKIFUMI TSUKIMORI et al.) 03 May 2007 See the abstract; claim 1 and figure 7.	1-20
A	US 2007-0266264 A1 (SAUL LEWITES et al.) 15 November 2007 See the abstract; claims 1, 9 and figure 3.	1-20

 Further documents are listed in the continuation of Box C. See patent family annex.

* Special categories of cited documents:

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

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

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of 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

26 APRIL 2012 (26.04.2012)

Date of mailing of the international search report

01 MAY 2012 (01.05.2012)

Name and mailing address of the ISA/KR

Korean Intellectual Property Office
Government Complex-Daejeon, 189 Cheongsu-ro,
Seo-gu, Daejeon 302-701, Republic of Korea

Facsimile No. 82-42-472-7140

Authorized officer

LEE, Young Su

Telephone No. 82-42-481-8456



INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2011/050580

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2009--03 197 12 A 1	24 . 12 ,2009	US 2009-0158068 A1 US 7603504 B2 US 7765352 B2	18 . 06 ,2009 13 . 10 ,2009 27 . 07 ,2010
us 2007--0 101 174 A 1	03 . 05 ,2007	JP 04-62 1113 B2 JP 2007- 122437 A JP 4621113 B2 US 7673 163 B2	05 . 11 ,2010 17 . 05 ,2007 26 . 01 ,201 1 02 . 03 ,2010
us 2007--0266264 A 1	15 . 11 ,2007	EP 1855 182 A2 EP 1892626 A2 EP 1892626 A3 US 2007-0266265 A1 US 2011-202778 A1 US 7647509 B2 US 7797555 B2	14 . 11 ,2007 27 . 02 ,2008 30 . 09 ,2009 15 . 11 ,2007 18 . 08 ,201 1 12 . 01 ,2010 14 . 09 ,2010