US 20110145615A1

(19) United States

(12) Patent Application Publication

Rychlik et al.

(10) Pub. No.: US 2011/0145615 A1

(43) Pub. Date: Jun. 16, 2011

(54) SYSTEM AND METHOD FOR CONTROLLING CENTRAL PROCESSING UNIT POWER BASED ON INFERRED WORKLOAD PARALLELISM

(76) Inventors: **Bohuslav Rychlik**, San Diego, CA (US); **Robert A. Glenn**, Boulder, CO (US); **Ali Iranli**, San Diego, CA (US); **Brian J. Salsbery**, Boulder, CO (US); **Sumit Sur**, Boulder, CO (US); **Steven S. Thomson**, San Diego, CA (US)

(21) Appl. No.: **12/944,140**

(22) Filed: **Nov. 11, 2010**

Related U.S. Application Data

(60) Provisional application No. 61/286,953, filed on Dec. 16, 2009.

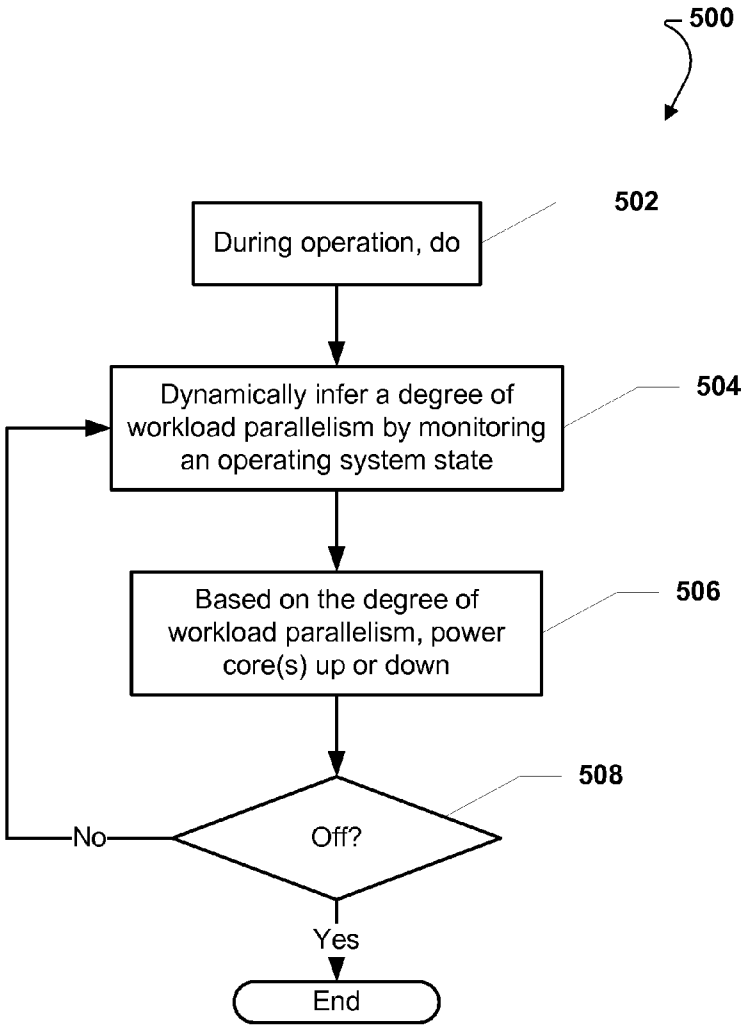
Publication Classification

(51) **Int. Cl.**
G06F 1/32 (2006.01)

(52) **U.S. Cl.** **713/323**

(57) **ABSTRACT**

A method of dynamically controlling power within a multi-core CPU is disclosed and may include receiving a degree of parallelism in a workload of a zeroth core and determining whether the degree of parallelism in the workload of the zeroth core is equal to a first wake condition. Further, the method may include determining a time duration for which the first wake condition is met when the degree of parallelism in the workload of the zeroth core is equal to the first wake condition and determining whether the time duration is equal to a first confirm wake condition. The method may also include invoking an operating system to power up a first core when the time duration is equal to the first confirm wake condition.



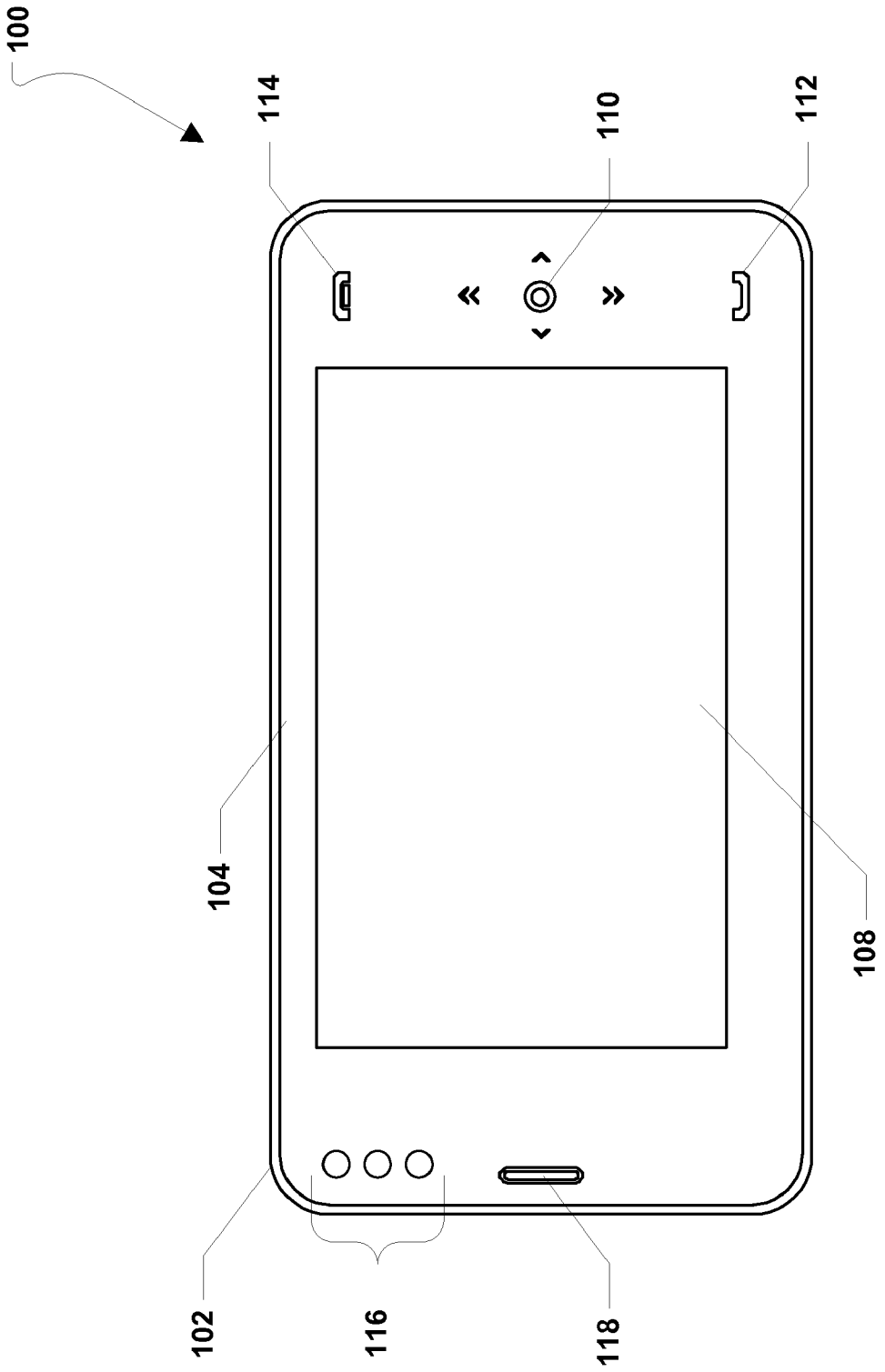


FIG. 1

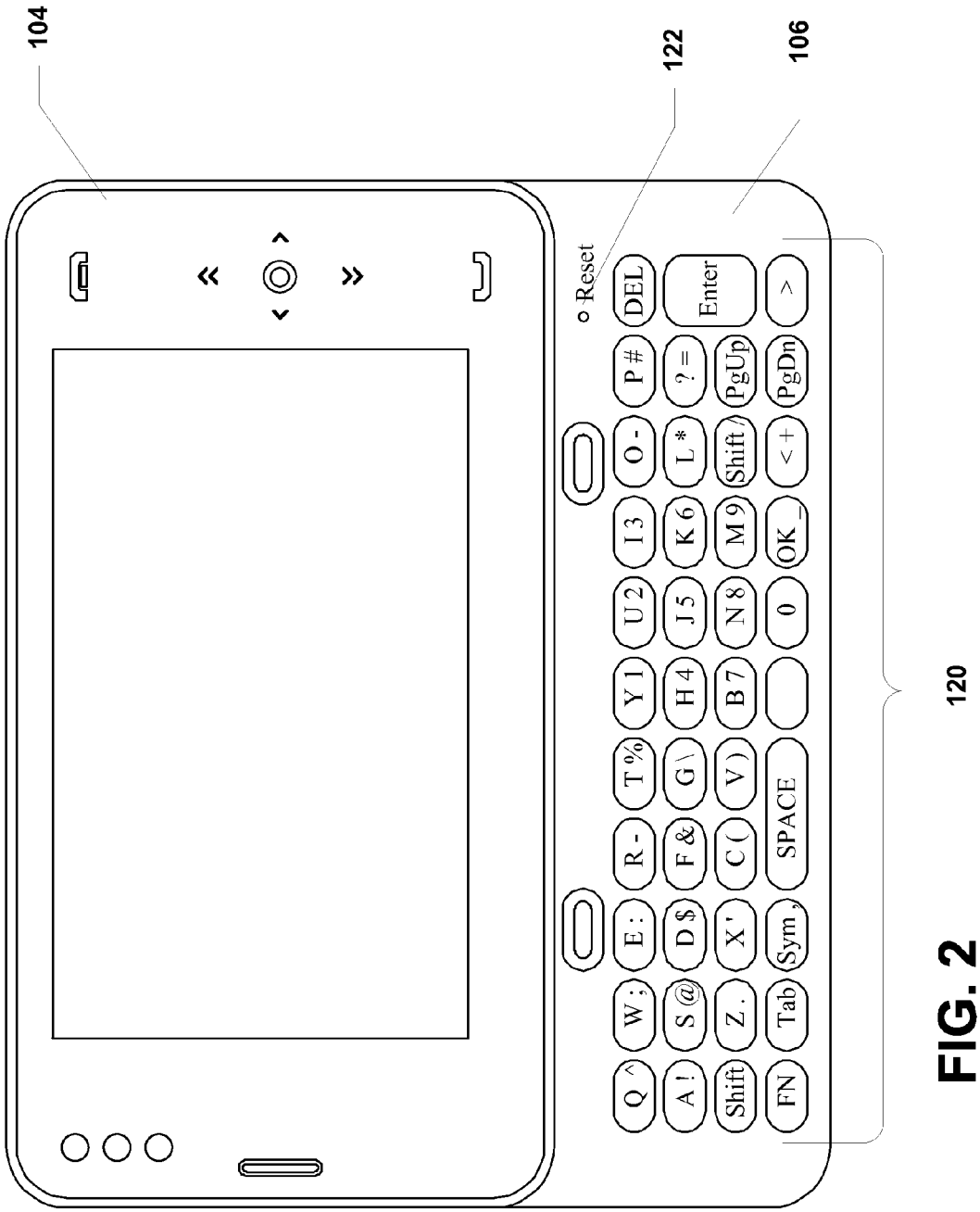

104
122
106
120
Reset
Q ^
W ;
E :
R -
T %
Y 1
U 2
I 3
O -
P #
DEL
A !
S @
D $
F &
G \
H 4
J 5
K 6
L *
? =
Enter
Shift
Z .
X '
C (
V)
B 7
N 8
M 9
Shift /
PgUp
FN
Tab
Sym ,
SPACE
0
OK _
< +
PgDn
>

FIG. 2

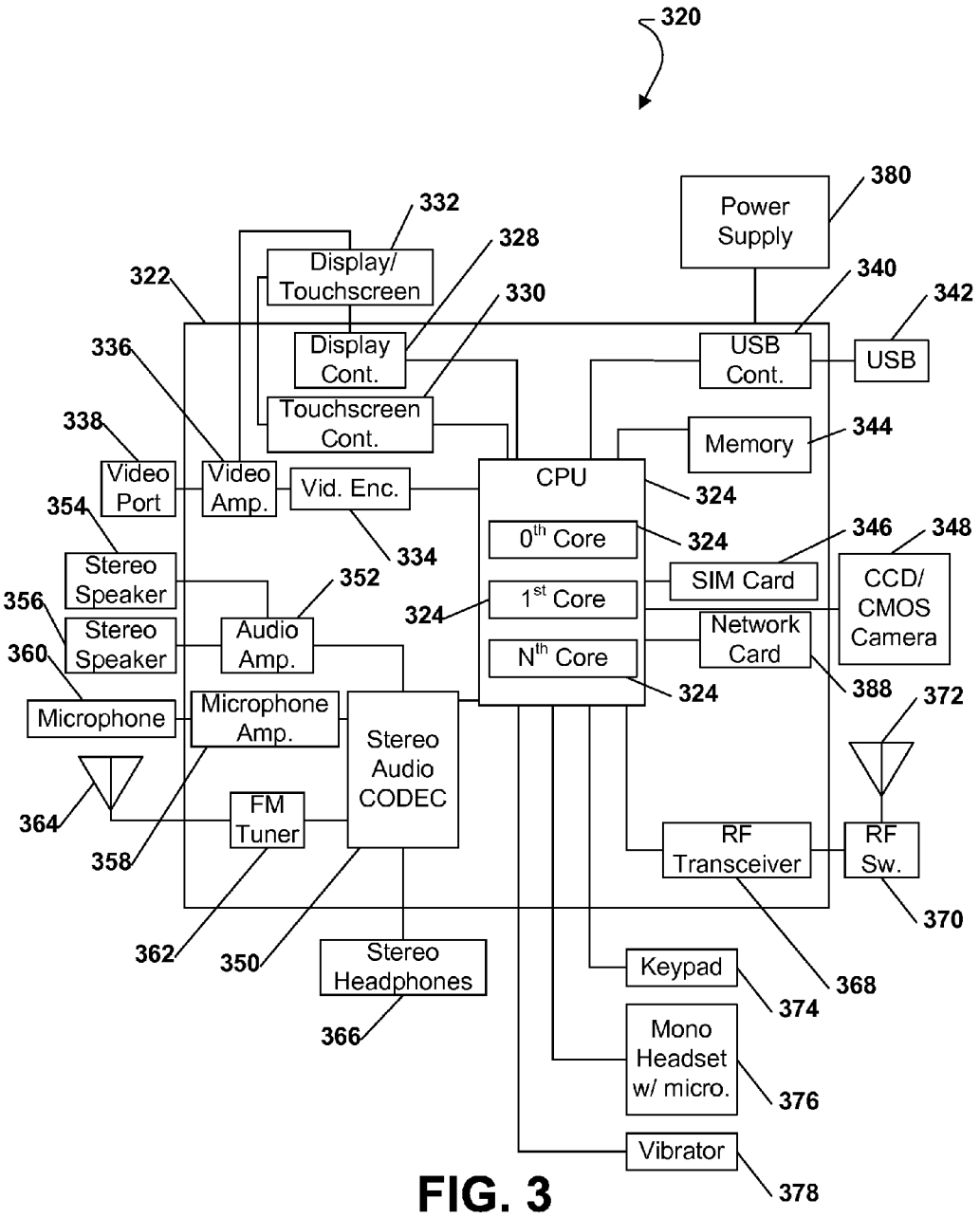

320
Power Supply
380
332
328
Display/ Touchscreen
322
330
340
342
336
Display Cont.
USB Cont.
USB
Touchscreen Cont.
338
Memory
344
Video Port
Video Amp.
Vid. Enc.
CPU
324
354
0th Core
324
346
348
334
Stereo Speaker
352
SIM Card
CCD/ CMOS Camera
356
1st Core
324
Stereo Speaker
Audio Amp.
Network Card
360
Nth Core
388
Microphone
Microphone Amp.
Stereo Audio CODEC
324
372
364
FM Tuner
RF Transceiver
RF Sw.
358
370
362
350
Stereo Headphones
Keypad
368
374
366
Mono Headset w/ micro.
376
Vibrator
378

FIG. 3

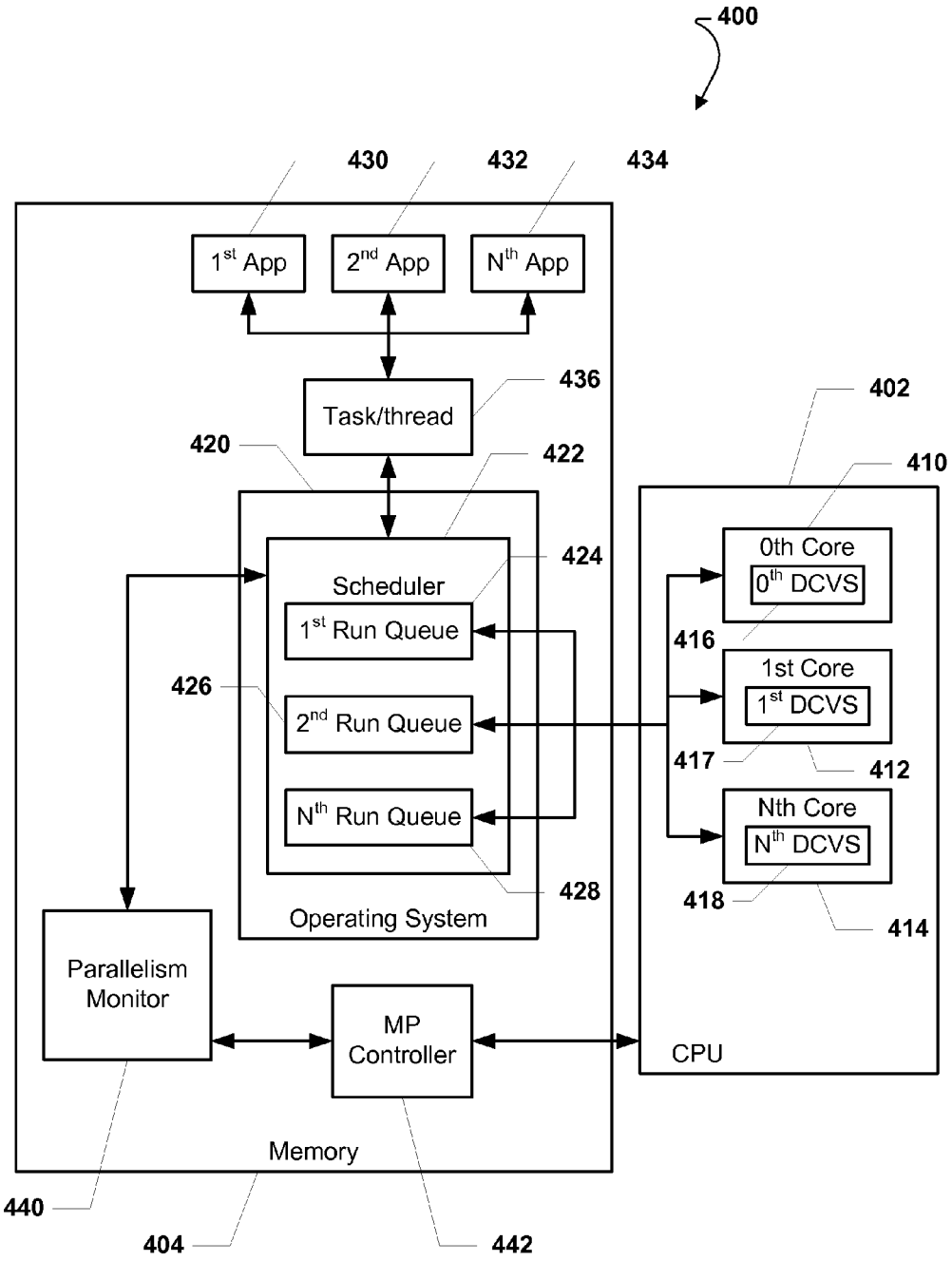

400
430
432
434
1st App
2nd App
Nth App
Task/thread
436
402
420
422
410
Scheduler
424
0th Core
0th DCVS
1st Run Queue
416
426
1st Core
1st DCVS
2nd Run Queue
417
412
Nth Core
Nth DCVS
Nth Run Queue
428
418
414
Operating System
Parallelism Monitor
MP Controller
CPU
Memory
440
404
442

FIG. 4

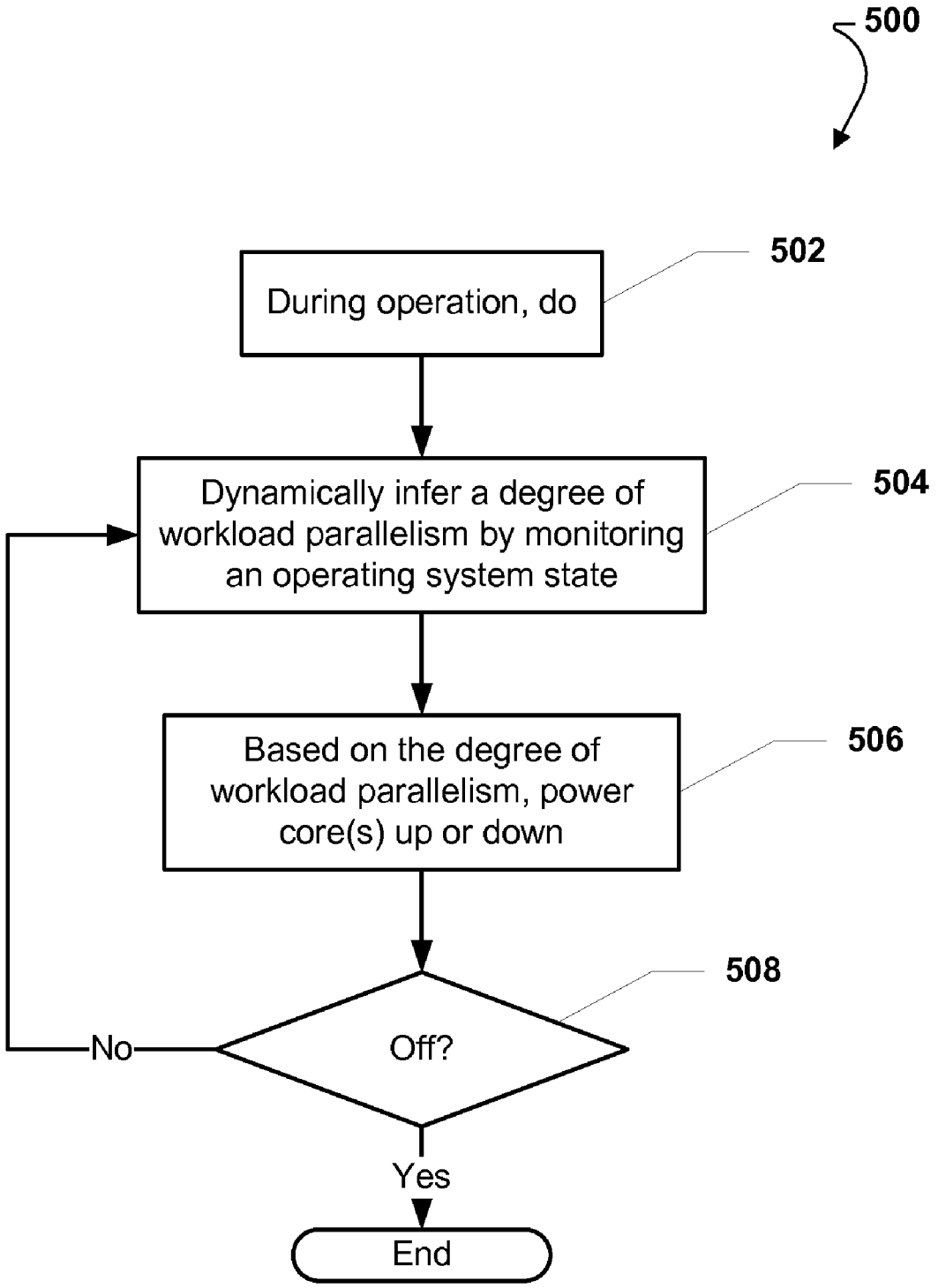

500
502
During operation, do
504
Dynamically infer a degree of workload parallelism by monitoring an operating system state
506
Based on the degree of workload parallelism, power core(s) up or down
508
Off?
No
Yes
End

FIG. 5

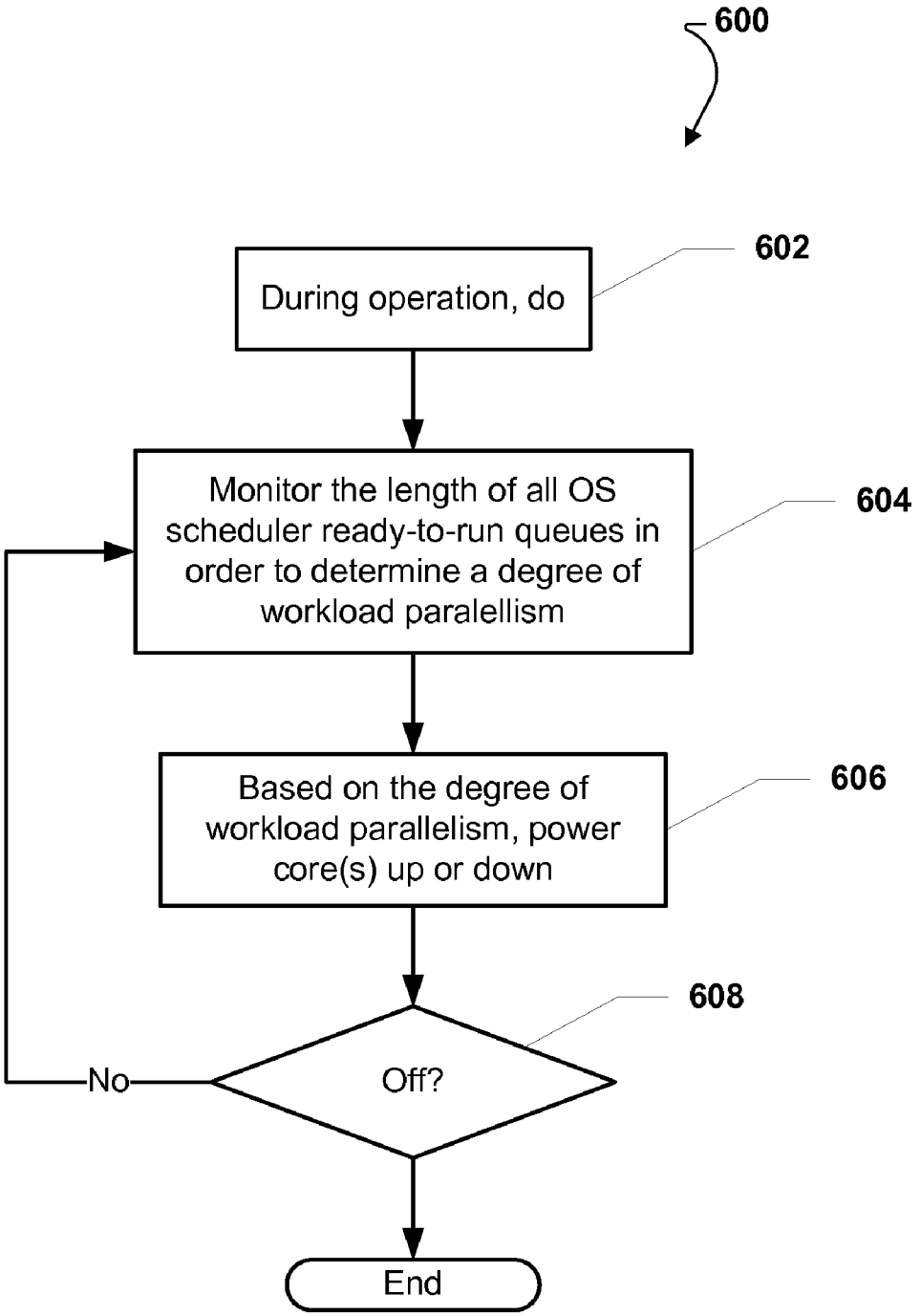

600
During operation, do
602
Monitor the length of all OS scheduler ready-to-run queues in order to determine a degree of workload paralellism
604
Based on the degree of workload parallelism, power core(s) up or down
606
Off?
608
No
End

FIG. 6

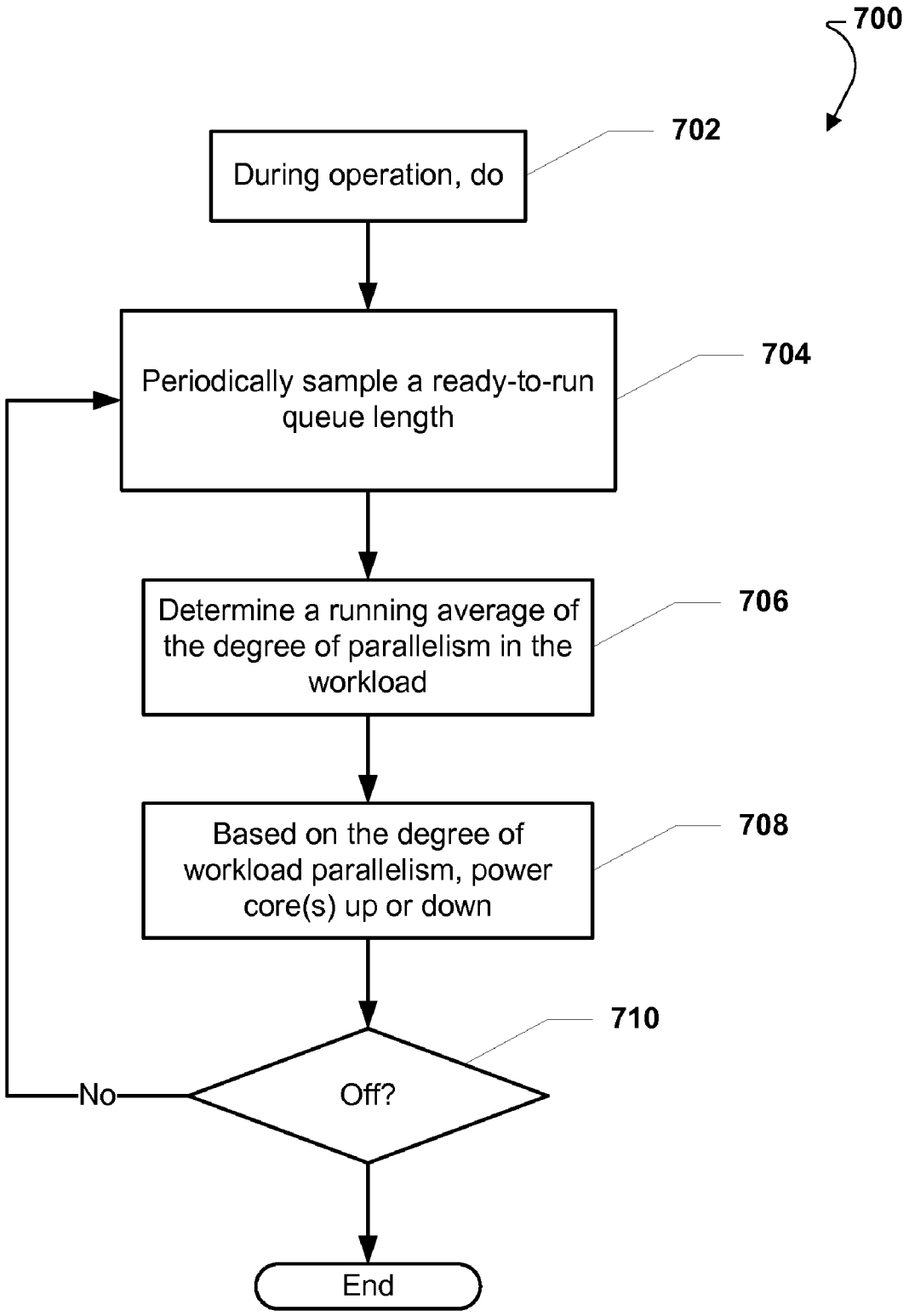

700
702
During operation, do
704
Periodically sample a ready-to-run queue length
706
Determine a running average of the degree of parallelism in the workload
708
Based on the degree of workload parallelism, power core(s) up or down
710
Off?
No
End

FIG. 7

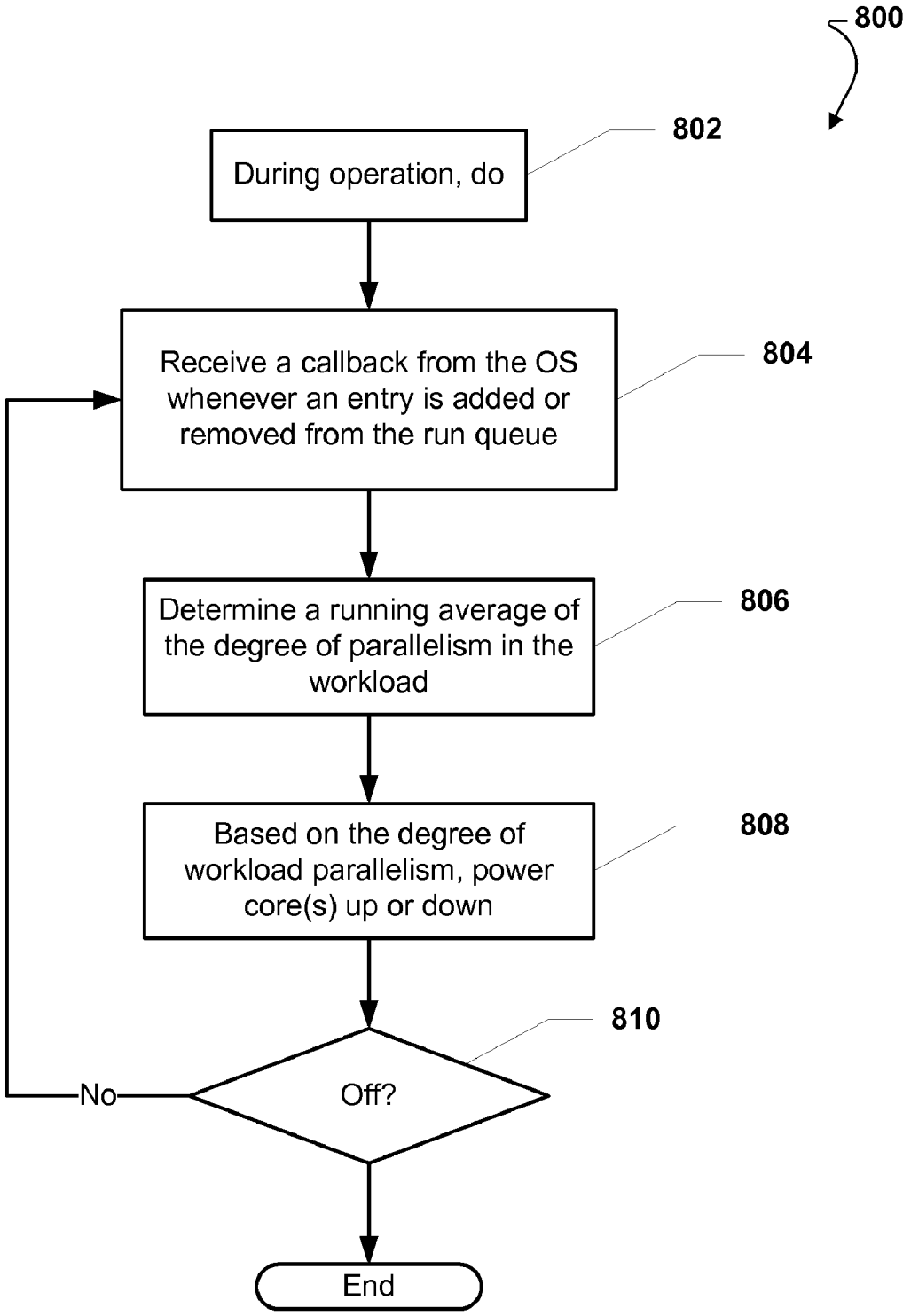

800
802
During operation, do
804
Receive a callback from the OS whenever an entry is added or removed from the run queue
806
Determine a running average of the degree of parallelism in the workload
808
Based on the degree of workload parallelism, power core(s) up or down
810
Off?
No
End

FIG. 8

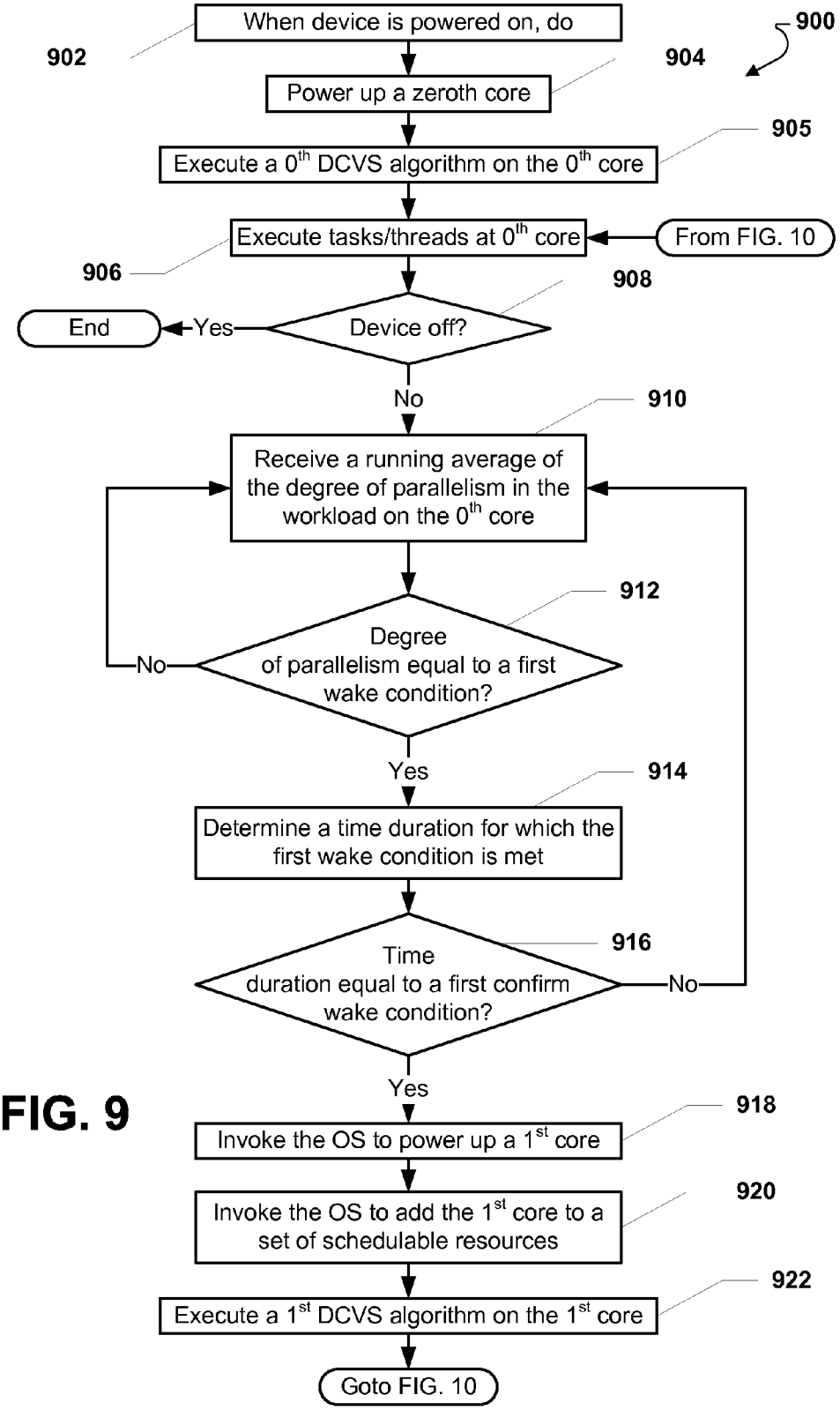

900
902
When device is powered on, do
904
Power up a zeroth core
905
Execute a 0th DCVS algorithm on the 0th core
906
Execute tasks/threads at 0th core
From FIG. 10
908
Device off?
Yes
End
No
910
Receive a running average of the degree of parallelism in the workload on the 0th core
912
Degree of parallelism equal to a first wake condition?
No
Yes
914
Determine a time duration for which the first wake condition is met
916
Time duration equal to a first confirm wake condition?
No
Yes
918
Invoke the OS to power up a 1st core
920
Invoke the OS to add the 1st core to a set of schedulable resources
922
Execute a 1st DCVS algorithm on the 1st core
Goto FIG. 10

FIG. 9

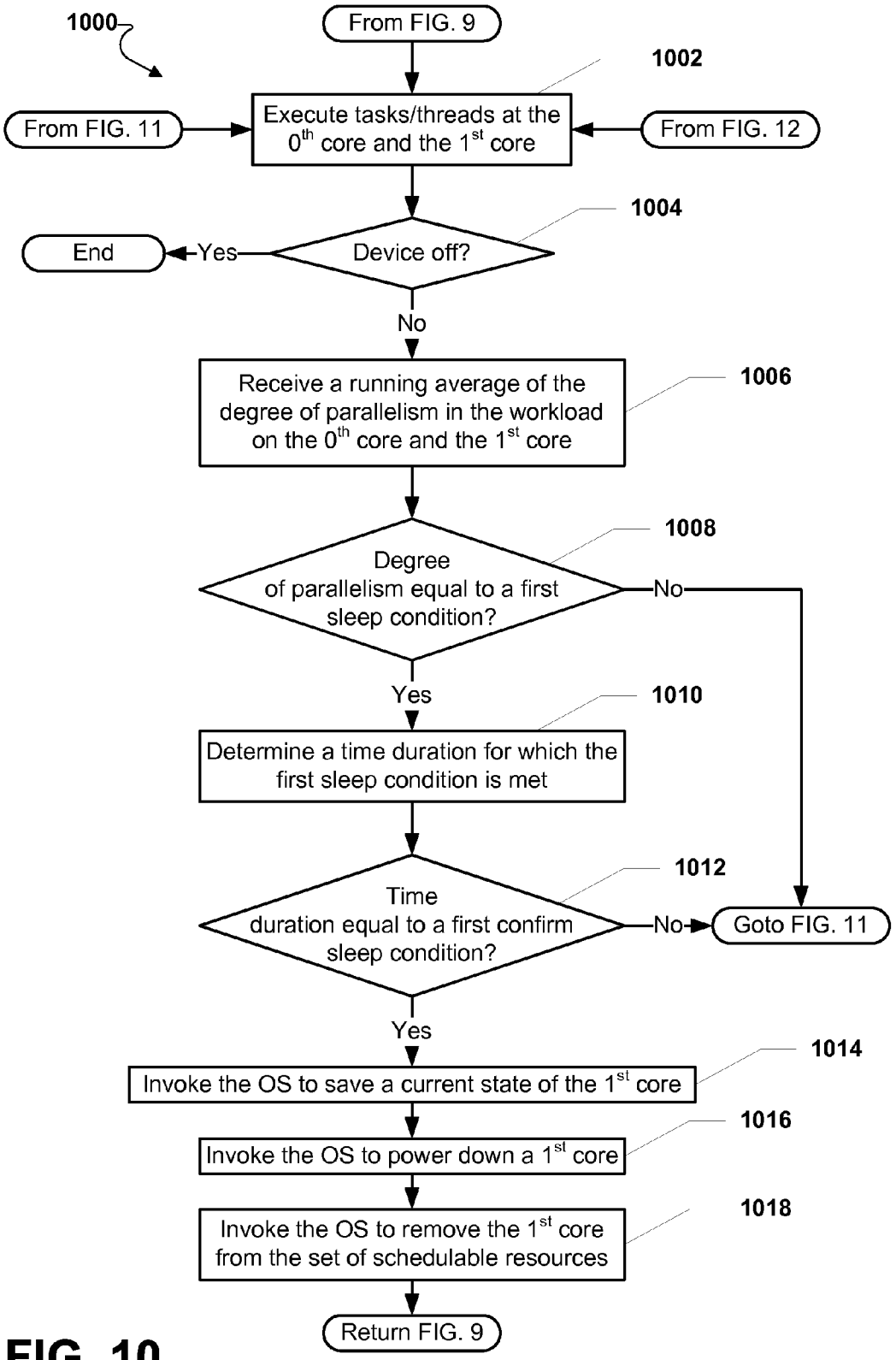

1000
From FIG. 9
1002
From FIG. 11
Execute tasks/threads at the 0th core and the 1st core
From FIG. 12
1004
End
Yes
Device off?
No
Receive a running average of the degree of parallelism in the workload on the 0th core and the 1st core
1006
1008
Degree of parallelism equal to a first sleep condition?
No
Yes
1010
Determine a time duration for which the first sleep condition is met
1012
Time duration equal to a first confirm sleep condition?
No
Goto FIG. 11
Yes
1014
Invoke the OS to save a current state of the 1st core
1016
Invoke the OS to power down a 1st core
1018
Invoke the OS to remove the 1st core from the set of schedulable resources
Return FIG. 9

FIG. 10

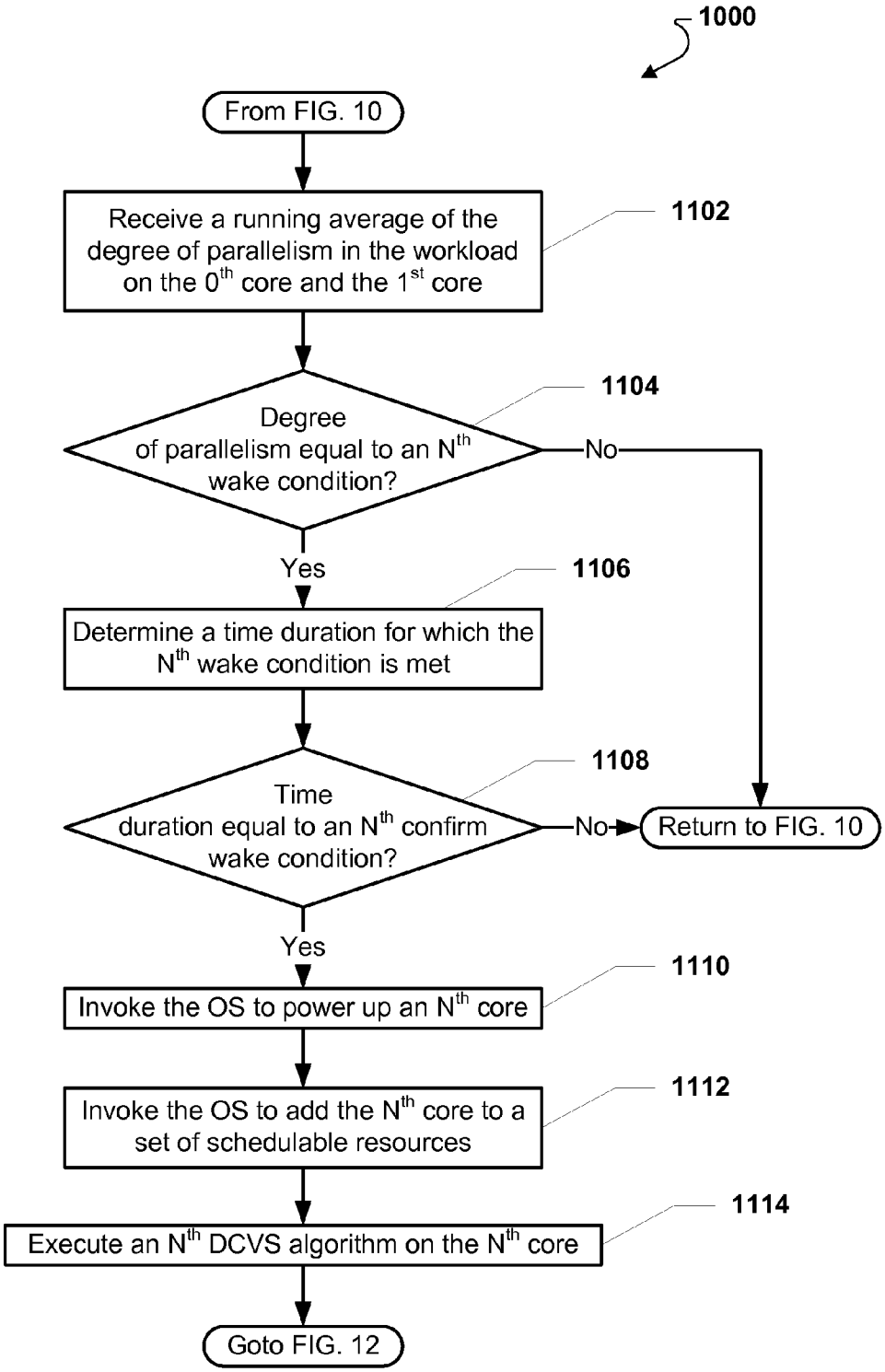

1000
From FIG. 10
Receive a running average of the degree of parallelism in the workload on the 0th core and the 1st core
1102
Degree of parallelism equal to an Nth wake condition?
1104
No
Yes
Determine a time duration for which the Nth wake condition is met
1106
Time duration equal to an Nth confirm wake condition?
1108
No
Return to FIG. 10
Yes
Invoke the OS to power up an Nth core
1110
Invoke the OS to add the Nth core to a set of schedulable resources
1112
Execute an Nth DCVS algorithm on the Nth core
1114
Goto FIG. 12

FIG. 11

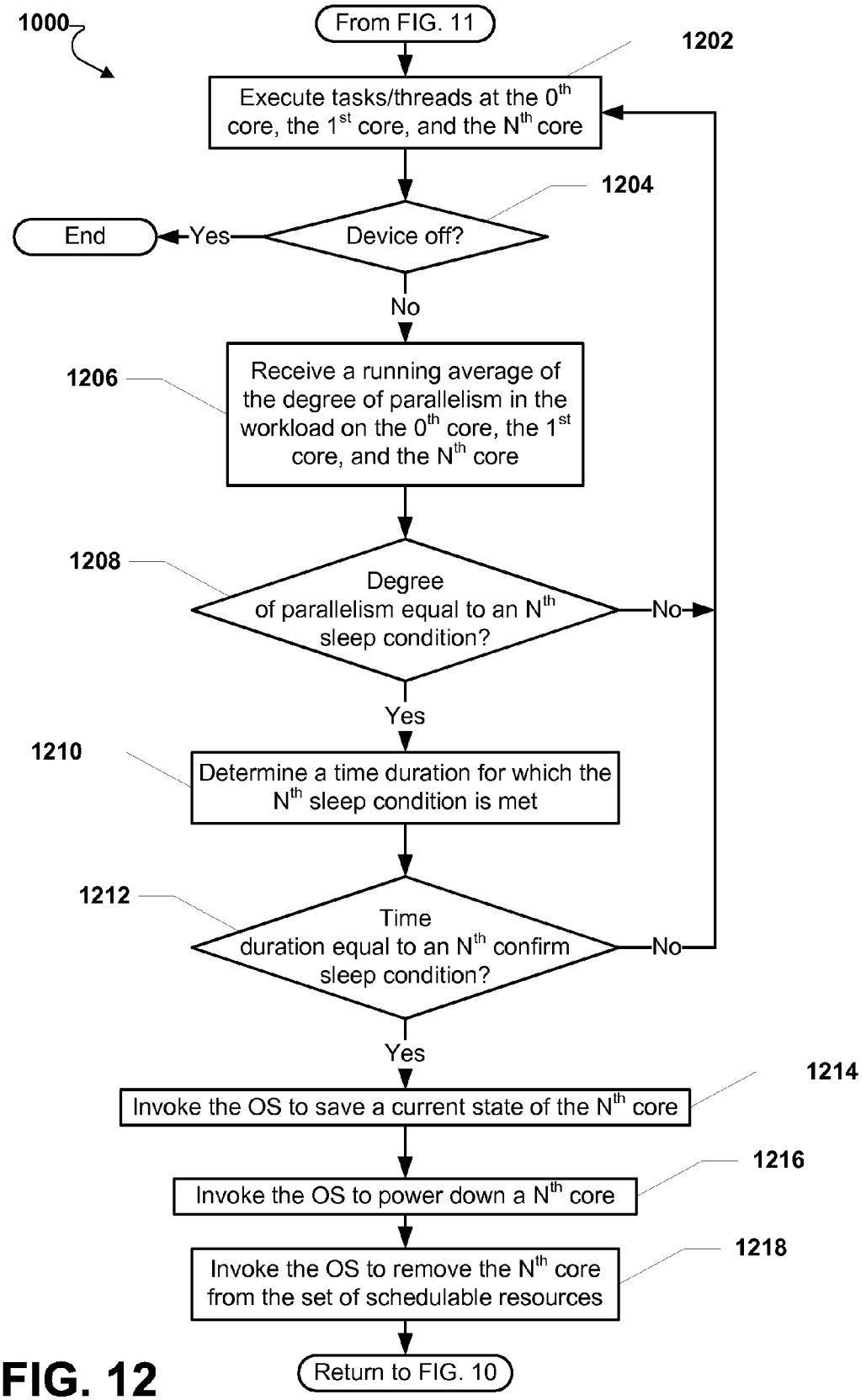

1000
From FIG. 11
1202
Execute tasks/threads at the 0th core, the 1st core, and the Nth core
1204
Device off?
Yes
End
No
1206
Receive a running average of the degree of parallelism in the workload on the 0th core, the 1st core, and the Nth core
1208
Degree of parallelism equal to an Nth sleep condition?
No
Yes
1210
Determine a time duration for which the Nth sleep condition is met
1212
Time duration equal to an Nth confirm sleep condition?
No
Yes
1214
Invoke the OS to save a current state of the Nth core
1216
Invoke the OS to power down a Nth core
1218
Invoke the OS to remove the Nth core from the set of schedulable resources
Return to FIG. 10

FIG. 12

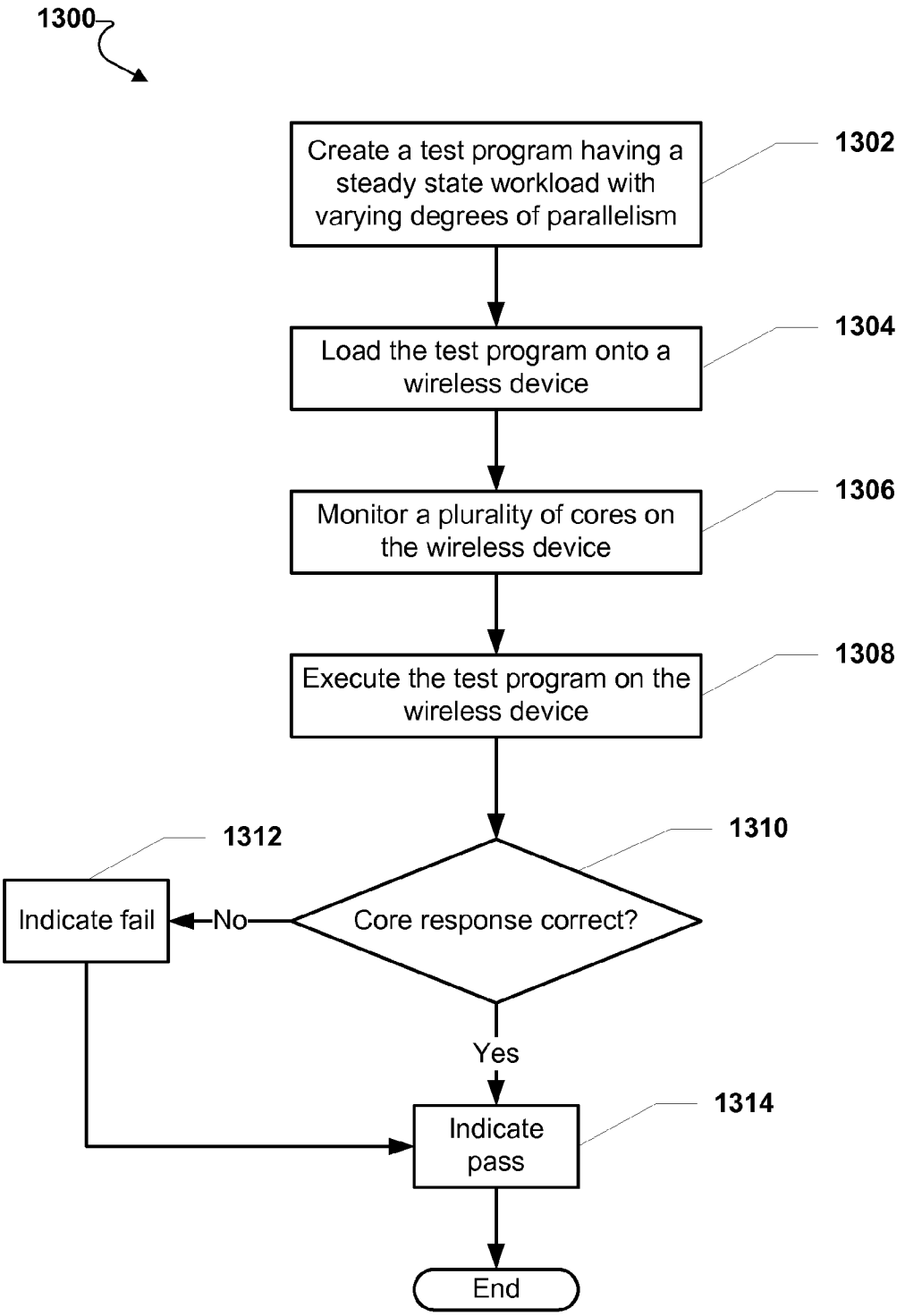

1300
Create a test program having a steady state workload with varying degrees of parallelism
1302
Load the test program onto a wireless device
1304
Monitor a plurality of cores on the wireless device
1306
Execute the test program on the wireless device
1308
Core response correct?
1310
No
Indicate fail
1312
Yes
Indicate pass
1314
End

FIG. 13

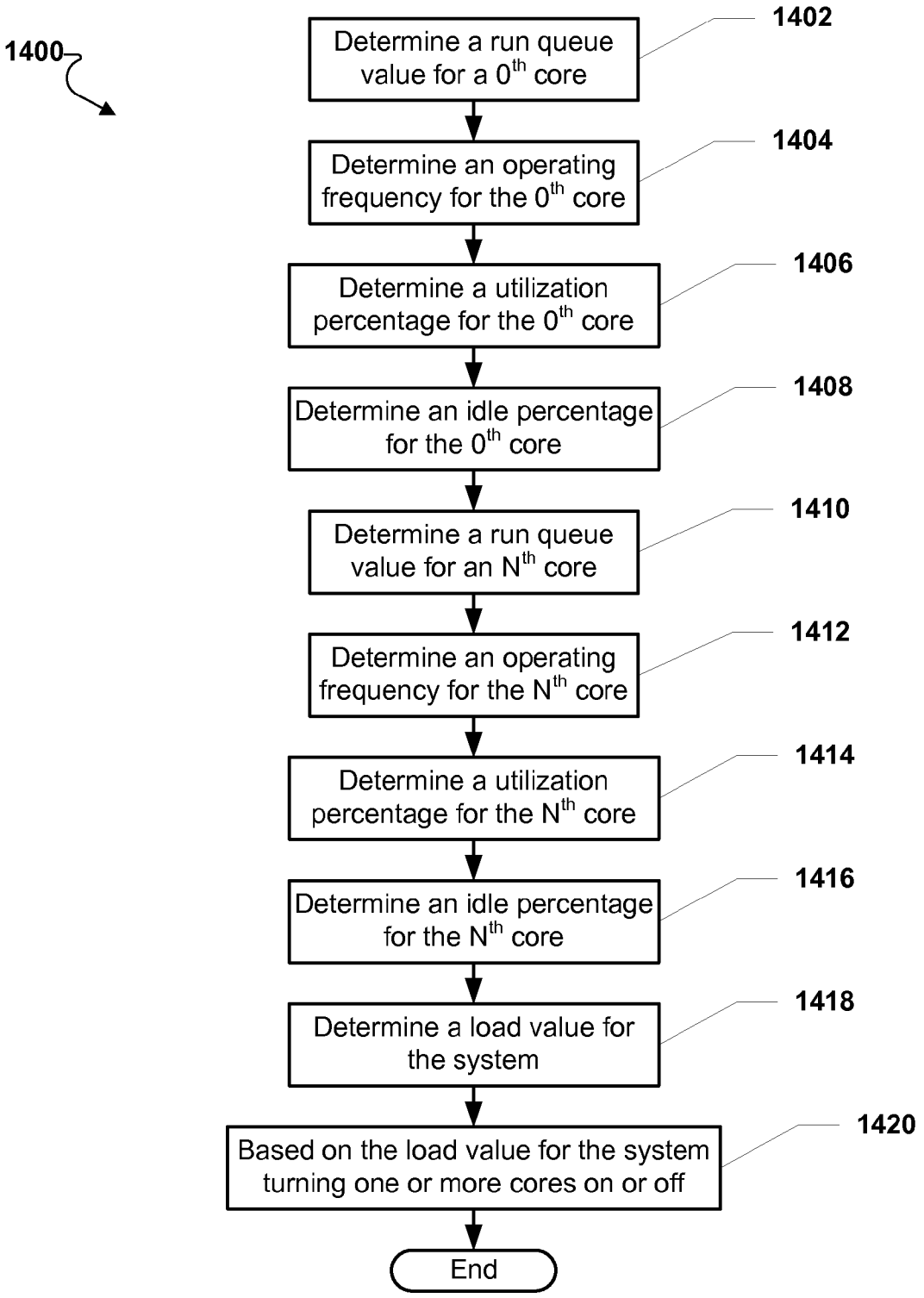

1400
Determine a run queue value for a 0th core
1402
Determine an operating frequency for the 0th core
1404
Determine a utilization percentage for the 0th core
1406
Determine an idle percentage for the 0th core
1408
Determine a run queue value for an Nth core
1410
Determine an operating frequency for the Nth core
1412
Determine a utilization percentage for the Nth core
1414
Determine an idle percentage for the Nth core
1416
Determine a load value for the system
1418
Based on the load value for the system turning one or more cores on or off
1420
End

FIG. 14

SYSTEM AND METHOD FOR CONTROLLING CENTRAL PROCESSING UNIT POWER BASED ON INFERRED WORKLOAD PARALLELISM

RELATED APPLICATIONS

[0001] The present application claims priority to U.S. Provisional Patent Application Ser. No. 61/286,953, entitled SYSTEM AND METHOD OF DYNAMICALLY CONTROLLING A PLURALITY OF CORES IN A MULTICORE CENTRAL PROCESSING UNIT, filed on Dec. 16, 2009, the contents of which are fully incorporated by reference.

CROSS-REFERENCED APPLICATIONS

[0002] The present application is related to, and incorporates by reference, U.S. patent application Ser. No. _____, entitled SYSTEM AND METHOD FOR CONTROLLING CENTRAL PROCESSING UNIT POWER IN A VIRTUALIZED SYSTEM, by Rychlik et al., filed concurrently (Attorney Docket Number 100329U1). The present application is related to, and incorporates by reference, U.S. patent application Ser. No. _____, entitled SYSTEM AND METHOD FOR ASYNCHRONOUSLY AND INDEPENDENTLY CONTROLLING CORE CLOCKS IN A MULTICORE CENTRAL PROCESSING UNIT, by Rychlik et al., filed concurrently (Attorney Docket Number 100330U1). The present application is related to, and incorporates by reference, U.S. patent application Ser. No. _____, entitled SYSTEM AND METHOD FOR CONTROLLING CENTRAL PROCESSING UNIT POWER WITH REDUCED FREQUENCY OSCILLATIONS, by Thomson et al., filed concurrently (Attorney Docket Number 100339U1). The present application is related to, and incorporates by reference, U.S. patent application Ser. No. _____, entitled SYSTEM AND METHOD FOR CONTROLLING CENTRAL PROCESSING UNIT POWER WITH GUARANTEED TRANSIENT DEADLINES, by Thomson et al., filed concurrently (Attorney Docket Number 100340U1). The present application is related to, and incorporates by reference, U.S. patent application Ser. No. _____, entitled SYSTEM AND METHOD FOR CONTROLLING CENTRAL PROCESSING UNIT POWER WITH GUARANTEED STEADY STATE DEADLINES, by Thomson et al., filed concurrently (Attorney Docket Number 100341U1). The present application is related to, and incorporates by reference, U.S. patent application Ser. No. _____, entitled SYSTEM AND METHOD FOR DYNAMICALLY CONTROLLING A PLURALITY OF CORES IN A MULTICORE CENTRAL PROCESSING UNIT BASED ON TEMPERATURE, by Sur et al., filed concurrently (Attorney Docket Number 100344U1).

DESCRIPTION OF THE RELATED ART

[0003] Portable computing devices (PDs) are ubiquitous. These devices may include cellular telephones, portable digital assistants (PDAs), portable game consoles, palmtop computers, and other portable electronic devices. In addition to the primary function of these devices, many include peripheral functions. For example, a cellular telephone may include the primary function of making cellular telephone calls and the peripheral functions of a still camera, a video camera, global positioning system (GPS) navigation, web browsing, sending and receiving emails, sending and receiving text messages, push-to-talk capabilities, etc. As the functionality of such a device increases, the processing power required to support such functionality also increases. Further, as the computing power increases, there exists a greater need to effectively manage the processor, or processors, that provide the computing power.

[0004] Accordingly, what is needed is an improved method of controlling power within a multicore CPU.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] In the figures, like reference numerals refer to like parts throughout the various views unless otherwise indicated.

[0006] FIG. **1** is a front plan view of a first aspect of a portable computing device (PCD) in a closed position;

[0007] FIG. **2** is a front plan view of the first aspect of a PCD in an open position;

[0008] FIG. **3** is a block diagram of a second aspect of a PCD;

[0009] FIG. **4** is a block diagram of a processing system;

[0010] FIG. **5** is a flowchart illustrating a first aspect of a method of dynamically controlling power within a multicore CPU;

[0011] FIG. **6** is a flowchart illustrating a second aspect of a method of dynamically controlling power within a multicore CPU;

[0012] FIG. **7** is a flowchart illustrating a third aspect of a method of dynamically controlling power within a multicore CPU;

[0013] FIG. **8** is a flowchart illustrating a fourth aspect of a method of dynamically controlling power within a multicore CPU;

[0014] FIG. **9** is a flowchart illustrating a first portion of a fifth aspect of a method of dynamically controlling power within a multicore CPU;

[0015] FIG. **10** is a flowchart illustrating a second portion of a fifth aspect of a method of dynamically controlling power within a multicore CPU;

[0016] FIG. **11** is a flowchart illustrating a third portion of a fifth aspect of a method of dynamically controlling power within a multicore CPU;

[0017] FIG. **12** is a flowchart illustrating a fourth portion of a fifth aspect of a method of dynamically controlling power within a multicore CPU;

[0018] FIG. **13** is a flowchart illustrating a method of testing a multicore CPU; and

[0019] FIG. **14** is a flowchart illustrating a sixth aspect of a method of dynamically controlling power within a multicore CPU.

DETAILED DESCRIPTION

[0020] 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.

[0021] 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.

[0022] 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.

[0023] 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).

[0024] Referring initially to FIG. **1** and FIG. **2**, an exemplary portable computing device (PCD) is shown and is generally designated **100**. As shown, the PCD **100** may include a housing **102**. The housing **102** may include an upper housing portion **104** and a lower housing portion **106**. FIG. **1** shows that the upper housing portion **104** may include a display **108**. In a particular aspect, the display **108** may be a touch screen display. The upper housing portion **104** may also include a trackball input device **110**. Further, as shown in FIG. **1**, the upper housing portion **104** may include a power on button **112** and a power off button **114**. As shown in FIG. **1**, the upper housing portion **104** of the PCD **100** may include a plurality of indicator lights **116** and a speaker **118**. Each indicator light **116** may be a light emitting diode (LED).

[0025] In a particular aspect, as depicted in FIG. **2**, the upper housing portion **104** is movable relative to the lower housing portion **106**. Specifically, the upper housing portion **104** may be slidable relative to the lower housing portion **106**. As shown in FIG. **2**, the lower housing portion **106** may include a multi-button keyboard **120**. In a particular aspect, the multi-button keyboard **120** may be a standard QWERTY keyboard. The multi-button keyboard **120** may be revealed when the upper housing portion **104** is moved relative to the lower housing portion **106**. FIG. **2** further illustrates that the PCD **100** may include a reset button **122** on the lower housing portion **106**.

[0026] Referring to FIG. **3**, an exemplary, non-limiting aspect of a portable computing device (PCD) is shown and is generally designated **320**. As shown, the PCD **320** includes an on-chip system **322** that includes a multicore CPU **324**. The multicore CPU **324** may include a zeroth core **325**, a first core **326**, and an Nth core **327**.

[0027] As illustrated in FIG. **3**, a display controller **328** and a touch screen controller **330** are coupled to the multicore CPU **324**. In turn, a touch screen display **332** external to the on-chip system **322** is coupled to the display controller **328** and the touch screen controller **330**.

[0028] FIG. **3** further indicates that a video encoder **334**, e.g., a phase alternating line (PAL) encoder, a sequential couleur a memoire (SECAM) encoder, or a national television system(s) committee (NTSC) encoder, is coupled to the multicore CPU **324**. Further, a video amplifier **336** is coupled to the video encoder **334** and the touch screen display **332**. Also, a video port **338** is coupled to the video amplifier **336**. As depicted in FIG. **3**, a universal serial bus (USB) controller **340** is coupled to the multicore CPU **324**. Also, a USB port **342** is coupled to the USB controller **340**. A memory **344** and a subscriber identity module (SIM) card **346** may also be coupled to the multicore CPU **324**. Further, as shown in FIG. **3**, a digital camera **348** may be coupled to the multicore CPU **324**. In an exemplary aspect, the digital camera **348** is a charge-coupled device (CCD) camera or a complementary metal-oxide semiconductor (CMOS) camera.

[0029] As further illustrated in FIG. **3**, a stereo audio CODEC **350** may be coupled to the multicore CPU **324**. Moreover, an audio amplifier **352** may coupled to the stereo audio CODEC **350**. In an exemplary aspect, a first stereo speaker **354** and a second stereo speaker **356** are coupled to the audio amplifier **352**. FIG. **3** shows that a microphone amplifier **358** may be also coupled to the stereo audio CODEC **350**. Additionally, a microphone **360** may be coupled to the microphone amplifier **358**. In a particular aspect, a frequency modulation (FM) radio tuner **362** may be coupled to the stereo audio CODEC **350**. Also, an FM antenna **364** is coupled to the FM radio tuner **362**. Further, stereo headphones **366** may be coupled to the stereo audio CODEC **350**.

[0030] FIG. **3** further indicates that a radio frequency (RF) transceiver **368** may be coupled to the multicore CPU **324**. An RF switch **370** may be coupled to the RF transceiver **368** and an RF antenna **372**. As shown in FIG. **3**, a keypad **374** may be coupled to the multicore CPU **324**. Also, a mono headset with a microphone **376** may be coupled to the multicore CPU **324**. Further, a vibrator device **378** may be coupled to the multicore CPU **324**. FIG. **3** also shows that a power supply **380** may be coupled to the on-chip system **322**. In a particular aspect, the power supply **380** is a direct current (DC) power supply that provides power to the various components of the PCD **320** that require power. Further, in a particular aspect, the power supply is 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.

[0031] FIG. **3** further indicates that the PCD **320** may also include a network card **388** that may be used to access a data network, e.g., a local area network, a personal area network, or any other network. The network card **388** may be a Bluetooth network card, a WiFi network card, a personal area network (PAN) card, a personal area network ultra-low-power technology (PeANUT) network card, or any other network card well known in the art. Further, the network card **388** may be incorporated into a chip, i.e., the network card **388** may be a full solution in a chip, and may not be a separate network card **388**.

[0032] As depicted in FIG. **3**, the touch screen display **332**, the video port **338**, the USB port **342**, the camera **348**, the first stereo speaker **354**, the second stereo speaker **356**, the microphone **360**, the FM antenna **364**, the stereo headphones **366**, the RF switch **370**, the RF antenna **372**, the keypad **374**, the mono headset **376**, the vibrator **378**, and the power supply **380** are external to the on-chip system **322**.

[0033] In a particular aspect, one or more of the method steps described herein may be stored in the memory **344** as computer program instructions. These instructions may be executed by the multicore CPU **324** in order to perform the methods described herein. Further, the multicore CPU **324**, the memory **344**, or a combination thereof may serve as a means for executing one or more of the method steps described herein in order to control power to each CPU, or core, within the multicore CPU **324**.

[0034] Referring to FIG. **4**, a processing system is shown and is generally designated **500**. In a particular aspect, the processing system **500** may be incorporated into the PCD **320** described above in conjunction with FIG. **3**. As shown, the processing system **500** may include a multicore central processing unit (CPU) **402** and a memory **404** connected to the multicore CPU **402**. The multicore CPU **402** may include a zeroth core **410**, a first core **412**, and an Nth core **414**. The zeroth core **410** may include a zeroth dynamic clock and voltage scaling (DCVS) algorithm **416** executing thereon. The first core **412** may include a first DCVS algorithm **417** executing thereon. Further, the Nth core **414** may include an Nth DCVS algorithm **418** executing thereon. In a particular aspect, each DCVS algorithm **416**, **417**, **418** may be independently executed on a respective core **412**, **414**, **416**.

[0035] Moreover, as illustrated, the memory **404** may include an operating system **420** stored thereon. The operating system **420** may include a scheduler **422** and the scheduler **422** may include a first run queue **424**, a second run queue **426**, and an Nth run queue **428**. The memory **404** may also include a first application **430**, a second application **432**, and an Nth application **434** stored thereon.

[0036] In a particular aspect, the applications **430**, **432**, **434** may send one or more tasks **436** to the operating system **420** to be processed at the cores **410**, **412**, **414** within the multicore CPU **402**. The tasks **436** may be processed, or executed, as single tasks, threads, or a combination thereof. Further, the scheduler **422** may schedule the tasks, threads, or a combination thereof for execution within the multicore CPU **402**. Additionally, the scheduler **422** may place the tasks, threads, or a combination thereof in the run queues **424**, **426**, **428**. The cores **410**, **412**, **414** may retrieve the tasks, threads, or a combination thereof from the run queues **424**, **426**, **428** as instructed, e.g., by the operating system **420** for processing, or execution, of those task and threads at the cores **410**, **412**, **414**.

[0037] FIG. **4** also shows that the memory **404** may include a parallelism monitor **440** and a multicore processor (MP) controller **442** stored thereon. The parallelism monitor **440** may be connected to the operating system **420** and the MP controller **442**. Specifically, the parallelism monitor **440** may be connected to the scheduler **422** within the operating system **420**. As described herein, the parallelism monitor **440** may monitor the workload on the cores **410**, **412**, **414** and the MP controller **442** may control the power to the cores **410**, **412**, **414** as described below. In a particular aspect, by executing one or more of the method steps, e.g., as computer program instructions, described herein, the parallelism monitor **440**, the MP controller **442**, or a combination thereof may serve a means for dynamically controlling the power to the cores **410**, **412**, **414** within the multicore CPU **402**.

[0038] In a particular dual-core aspect, during operating, the MP controller **442** may receive an input from the parallelism monitor **440**. The input may be a total system load. Moreover, the input may be a running average of the degree of parallelism in the workload. Based on the input, the MP controller **442** may determine whether a single core or two cores should be powered on. Further, the MP controller **442** may output a control signal to the multicore CPU **402**. The control signal may indicate whether to turn additional cores on or off. In the dual-core example, the MP controller **442** may include four threshold values for controlling the decision to power the cores on and off. The four threshold values may include a number of ready-to-run threads in the OS scheduler queue to trigger a core wake, N_w; a time duration for which N_w has been exceed to confirm a core wake, Tw; a number of ready-to-run threads in the OS scheduler to trigger a core sleep, N_s; and a time duration for which Ns has been exceeded to confirm a core sleep, T_s.

[0039] Beginning with a single core active, e.g., the zeroth core **410**, when the running average of the degree of parallelism in the workload on the zeroth core **410** meets or exceeds N_w for a duration of at least T_w, the MP controller **442** may wake up a second core, e.g., the first core **412**. Conversely, when both cores, e.g., the zeroth core **410** and the first core **412**, are active and when the degree of parallelism in the workload falls below N_s for at least a duration of T_s, the MP controller **442** may decide to put the second core, e.g., the first core **412**, to sleep.

[0040] In a particular aspect, sustained threshold parallelism over the time T_w implies that the single core is saturated. Further, the cores may be started at the most power efficient voltage-frequency (VF) operating point. In particular aspect, two cores operating at an optimal VF offer more Dhrystone million instructions per second (DMIPS) that a single core operating at a maximum VF. In a dual-core aspect, dual, independent DCVS algorithms may adapt to asymmetric workloads and in some cases, heterogeneous cores. Further, in a dual-core aspect, the two cores should remain active during multi-tasking workloads in order to avoid a performance penalty that is doubled. Also, when the parallelism falls below N_s for the prescribed time T_s, the second core should be powered off and not placed in standby. In a particular aspect, placing the second core in standby may increase power leakage and also may reduce performance.

[0041] The optimal values of the parameters N_w, T_w, N_s, and T_s may depend on the exact power consumption characteristics of the system **400**. However, in one aspect, the values may be as follows:

[0042] N_w=1.2,

[0043] T_w=40 milliseconds (ms),

[0044] N_s=0.8, and

[0045] T_s=80 ms.

[0046] In this particular aspect, N_w=1.2 may ensure a sustained parallelism before the second core is awakened. N_s=0.8 may ensure a sustained absence of parallelism before the second core is put asleep. T_s=80 ms is based on a power collapse threshold of the system, 400 ms. T_w=40 ms is half of T_s to improve multicore responsiveness.

[0047] FIG. **5** illustrates a first aspect of a method of controlling power within a multi-core processor. The method is generally designated **500**. The method **500** commences at block **502** with a do loop in which during operation of a device having a multi-core processor, the succeeding steps may be performed. At block **504**, a power controller may dynamically infer a degree of workload parallelism within the CPUs, or cores, e.g., by monitoring an operating system state. Moving to block **506**, at least partially based on the degree of workload parallelism, the power controller may power core(s) up

or down. In other words, the power controller may turn the cores on or off based on the workload.

[0048] At decision **508**, the power controller may determine whether the device is powered off. If the device is powered off, the method may end. Otherwise, if the device remains powered on, the method **500** may return to block **504** and the method **500** may continue as described.

[0049] Referring now to FIG. **6**, a second aspect of a method of controlling power within a multi-core processor is shown and is generally designated **600**. The method **600** commences at block **602** with a do loop in which during operation of a device having a multi-core processor, the succeeding steps may be performed. At block **604**, a controller, e.g., a parallelism monitor, may monitor the length of all operating system (OS) scheduler ready-to-run queues in order to determine a degree of workload parallelism within the CPUs, or cores. In a particular aspect, the parallelism monitor may be a software program residing in a memory of the device. Further, in a particular aspect, the scheduler ready-to-run queue is a list of current tasks of threads that are available for scheduling on one or more CPUs. Some multi-core systems may only have a single ready-to-run queue. Other multicore systems may have multiple ready-to-run queues. Regardless, of the number of ready-to-run queues, at any instant in time, the total number of tasks, threads, or a combination thereof waiting on these queues, plus a number of tasks, threads, or a combination thereof actually running, may be an approximation for the degree of parallelism in the workload.

[0050] Moving to block **606**, at least partially based on the degree of workload parallelism, the parallelism monitor may power core(s) up or down. In other words, the parallelism monitor may turn the cores on or off based on the workload.

[0051] At decision **608**, the parallelism monitor may determine whether the device is powered off. If the device is powered off, the method may end. Otherwise, if the device remains powered on, the method **600** may return to block **604** and the method **600** may continue as described.

[0052] Referring to FIG. **7**, a third aspect of a method of controlling power within a multi-core processor is shown and is generally designated **700**. The method **700** commences at block **702** with a do loop in which during operation of a device having a multi-core processor, the succeeding steps may be performed. At block **704**, a parallelism monitor may periodically sample a ready-to-run queue length. For example, the parallelism monitor may sample the ready-to-run queue length every millisecond (1 ms). At block **706**, the parallelism monitor may determine a running average of the degree of parallelism in the workload. Moving to block **708**, at least partially based on the degree of workload parallelism, the parallelism monitor may power core(s) up or down. In other words, the parallelism monitor may turn the cores on or off based on the workload.

[0053] At decision **710**, the parallelism monitor may determine whether the device is powered off. If the device is powered off, the method may end. Otherwise, if the device remains powered on, the method **700** may return to block **704** and the method **700** may continue as described.

[0054] FIG. **8** depicts a fourth aspect of a method of controlling power within a multi-core processor is shown. The method is generally designated **800** and the method **800** may commence at block **802** with a do loop in which during operation of a device having a multi-core processor, the succeeding steps may be performed. At block **804**, a parallelism monitor may receive a callback from the operating system (OS) whenever an entry is added or removed from the OS scheduler run queue. Further, at block **806**, the parallelism monitor may determine a running average of the degree of parallelism in the workload of the CPUs, or cores.

[0055] Moving to block **808**, at least partially based on the degree of workload parallelism, the parallelism monitor may power core(s) up or down. In other words, the parallelism monitor may turn the cores on or off based on the workload. At decision **810**, the parallelism monitor may determine whether the device is powered off. If the device is powered off, the method may end. Otherwise, if the device remains powered on, the method **800** may return to block **804** and the method **800** may continue as described.

[0056] Referring now to FIG. **9** through FIG. **12**, a fifth aspect of a method of controlling power within a multi-core processor is shown and is generally designated **900**. The method **900** commences at block **902** with a do loop in which when a device having a multi-core processor is powered on, the following steps may be performed. At block **904**, a zeroth core may be powered up, i.e., energized. At block **905**, a zeroth DCVS algorithm may be executed locally on the zeroth core. Further, at block **906**, one or more tasks, or threads, may be executed at the zeroth core.

[0057] Moving to decision **908**, a multicore processor (MP) controller may determine whether the device is powered off. If so, the method **900** may end. Otherwise, if the device remains powered on, the method **900** may move to block **910** and the MP controller may receive a running average of the degree of parallelism in the workload on the zeroth core from a parallelism monitor. In a particular aspect, at any instant in time, the total number of tasks, threads, or a combination thereof, waiting on the ready-to-run queues of an operating system (OS) plus the number of tasks actually running may be an approximation for the degree of parallelism in the workload on the core.

[0058] At decision **912**, the MP controller may determine whether the degree of parallelism is equal to a first wake condition. In a particular aspect, the first wake condition may be a threshold value associated with the degree of parallelism in the workload on the core. For example, the threshold value may be a predetermined number of ready-to-run threads in the OS scheduler queues and if the parallelism is greater than or equal to this threshold value, the first wake condition may be met.

[0059] Returning to decision **912**, if the degree of parallelism is not equal to a first wake condition, the method **900** may return to block **910** and the method **900** may continue as described herein. Otherwise, if the degree of parallelism is equal to a first wake condition, the method **900** may move to block **914** and the MP controller may determine a time duration for which the first wake condition is met. At decision **916**, the MP controller may determine whether the time duration is equal to a first confirm wake condition. In a particular aspect, the first confirm wake condition may be a threshold time value and if the time duration for which the first wake condition is greater than or equal to the threshold value, the first confirm wake condition may be met.

[0060] Returning to decision **916**, if the time duration for which the first wake condition is met is not equal to a first confirm wake condition, the method **900** may return to block **910** and the method **900** may continue as described herein. Conversely, if the first confirm wake condition is met, the method **900** may move to block **918** and the MP controller

may invoke the OS to power up a first core so that two cores, i.e., the zeroth core and the first core, are running and executing threads and tasks. At block **920**, the MP controller may invoke the OS to add the first core to a set of schedulable resources available to the OS. Further, at block **922**, a first DCVS algorithm may be executed locally on the first core. Thereafter, the method **900** may proceed to block **1002** of FIG. **10**.

[0061] Moving now to block **1002** of FIG. **10**, one or more tasks, threads, or a combination thereof may be executed at the zeroth core and the first core. At decision **1004**, a MP controller may determine whether the device is powered off. If so, the method **900** may end. Otherwise, if the device remains powered on, the method **900** may move to block **1006** and the MP controller may receive a running average of the degree of parallelism in the workload on the zeroth core and the first core from the parallelism monitor. In a particular aspect, at any instant in time, the total number of tasks, threads, or a combination thereof, waiting on the ready-to-run queues of an operating system (OS) plus the number of tasks actually running may be an approximation for the degree of parallelism in the workload on the cores.

[0062] At decision **1008**, the MP controller may determine whether the degree of parallelism is equal to a first sleep condition. In a particular aspect, the first sleep condition may be a threshold value associated with the degree of parallelism in the workload on the core. For example, the threshold value may be a minimum number of ready-to-run threads in the OS scheduler queues and if the parallelism is less than or equal to this threshold value, the first sleep condition may be met.

[0063] Returning to decision **1008**, if the degree of parallelism is not equal to the first sleep condition, the method **900** may proceed to block **1102** of FIG. **11** and the method **900** may continue as described herein. Otherwise, if the degree of parallelism is equal to the first sleep condition, the method **900** may move to block **1010** and the MP controller may determine a time duration for which the first sleep condition is met. At decision **1012**, the MP controller may determine whether the time duration is equal to a first confirm sleep condition. In a particular aspect, the first confirm sleep condition may be a threshold time value and if the time duration for which the first sleep condition is greater than or equal to the threshold value, the first confirm sleep condition may be met.

[0064] Returning to decision **1012**, if the time duration for which the first sleep condition is met is not equal to a first confirm sleep condition, the method **900** may proceed to block **1102** of FIG. **11** and the method **900** may continue as described herein. Conversely, if the first confirm sleep condition is met, the method **900** may move to block **1014** and the MP controller may invoke the OS to save a current state of the first core. At block **1016**, the MP controller may invoke the OS to power down the first core so that one core, i.e., the zeroth core, is running and executing threads and tasks. Further, at block **1018**, the MP controller may invoke the OS to remove the first core from the set of schedulable resources available to the OS. Thereafter, the method **900** may return to block **906** of FIG. **9** and the method **900** may continue as described herein.

[0065] Referring now to FIG. **11**, at block **1102**, the MP controller may receive a running average of the degree of parallelism in the workload on the zeroth core and the first core from the parallelism monitor. In a particular aspect, at any instant in time, the total number of tasks, threads, or a combination thereof, waiting on the ready-to-run queues of an operating system (OS) plus the number of tasks actually running may be an approximation for the degree of parallelism in the workload on the cores. At decision **1104**, the MP controller may determine whether the degree of parallelism is equal to an Nth wake condition. In a particular aspect, the Nth wake condition may be a threshold value associated with the degree of parallelism in the workload on the cores. For example, the threshold value may be a maximum number of ready-to-run threads in the OS scheduler queues and if the parallelism is greater than or equal to this threshold value, the Nth wake condition may be met.

[0066] Returning to decision **1104**, if the degree of parallelism is not equal to the Nth wake condition, the method **900** may return to block **1002** of FIG. **10** and the method **900** may continue as described herein. Otherwise, if the degree of parallelism is equal to the Nth wake condition, the method **900** may move to block **1106** and the MP controller may determine a time duration for which the Nth wake condition is met. At decision **1108**, the MP controller may determine whether the time duration is equal to an Nth confirm wake condition. In a particular aspect, the Nth confirm wake condition may be a threshold time value and if the time duration for which the Nth wake condition is greater than or equal to the threshold value, the Nth confirm wake condition may be met.

[0067] Returning to decision **1108**, if the time duration for which the Nth wake condition is met is not equal to the Nth confirm wake condition, the method **900** may return to block **1002** of FIG. **10** and the method **900** may continue as described herein. Conversely, if the Nth confirm wake condition is met, the method **900** may move to block **1110** and the MP controller may invoke the OS to power up an Nth core so that N cores, i.e., the zeroth core, the first core, and the Nth core, are running and executing threads and tasks. At block **1112**, the MP controller may invoke the OS to add the Nth core to a set of schedulable resources available to the OS. Further, at block **1114**, an Nth DCVS algorithm may be executed locally on the Nth core. Thereafter, the method **900** may proceed to block **1202** of FIG. **12**.

[0068] At block **1202** of FIG. **12**, one or more tasks, threads, or a combination thereof may be executed at the zeroth core, the first core, and the Nth core. At decision **1204**, a MP controller may determine whether the device is powered off. If so, the method **900** may end. Otherwise, if the device remains powered on, the method **900** may move to block **1206** and the MP controller may receive a running average of the degree of parallelism in the workload on the zeroth core, the first core, and the Nth from the parallelism monitor. In a particular aspect, at any instant in time, the total number of tasks, threads, or a combination thereof, waiting on the ready-to-run queues of an operating system (OS) plus the number of tasks actually running may be an approximation for the degree of parallelism in the workload on the cores.

[0069] At decision **1208**, the MP controller may determine whether the degree of parallelism is equal to an Nth sleep condition. In a particular aspect, the Nth sleep condition may be a threshold value associated with the degree of parallelism in the workload on the core. For example, the threshold value may be a minimum number of ready-to-run threads in the OS scheduler queues and if the parallelism is less than or equal to this threshold value, the Nth sleep condition may be met.

[0070] Returning to decision **1208**, if the degree of parallelism is not equal to the Nth sleep condition, the method **900**

may proceed to block **1202** and the method **900** may continue as described herein. Otherwise, if the degree of parallelism is equal to the Nth sleep condition, the method **900** may move to block **1210** and the MP controller may determine a time duration for which the Nth sleep condition is met. At decision **1212**, the MP controller may determine whether the time duration is equal to an Nth confirm sleep condition. In a particular aspect, the Nth confirm sleep condition may be a threshold time value and if the time duration for which the Nth sleep condition is greater than or equal to the threshold value, the Nth confirm sleep condition may be met.

[0071] Returning to decision **1212**, if the time duration for which the Nth sleep condition is met is not equal to the Nth confirm sleep condition, the method **900** may return to block **1202** and the method **900** may continue as described herein. Conversely, if the Nth confirm sleep condition is met, the method **900** may move to block **1214** and the MP controller may invoke the OS to save a current state of the first core. At block **1216**, the MP controller may invoke the OS to power down the Nth core so that N minus one cores, e.g., the zeroth core, and the first core (if N is 2 and a second core is powered down), is running and executing threads and tasks. Further, at block **1218**, the MP controller may invoke the OS to remove the Nth core from the set of schedulable resources available to the OS. Thereafter, the method **900** may return to block **1002** of FIG. **10** and the method **900** may continue as described herein.

[0072] Referring now to FIG. **13**, a method of testing a multicore processor is shown and is generally designated **1300**. As shown, the method **1300** may begin at block **1302** where a test program may be created. The test program may include a steady state workload with varying degrees of parallelism which would cause a plurality of cores to power up and down depending on the degree of parallelism, as described herein.

[0073] Moving to block **1304**, the test program may be loaded onto a wireless device which includes a multicore processor. At block **1306**, a plurality of cores may be monitored on the wireless device. For example, the core up/down state for each core may be monitored. The up/down state may be monitored by monitoring the power on each core, by monitoring the total power consumed, the speed at which workloads are completed, or a combination thereof.

[0074] At block **1308**, the test program may be executed at the wireless device. Moreover, at decision **1310**, it may be determined whether the cores within the wireless device respond correctly to the execution of the test program. In other words, it may be determined whether the cores are powering up and/or down correctly in response to the execution of the test program. At decision **1310**, if the cores do not respond correctly to the execution of the test program, the method **1300** may move to block **1312** and a fail result may be indicated. The method **1300** may then end.

[0075] At decision **1310**, if the cores do respond correctly to the execution of the test program, the method may proceed to block **1314** and a pass result may be indicated. Thereafter, the method **1300** may end.

[0076] FIG. **14** illustrates a sixth aspect of a method of dynamically controlling power within a multicore CPU. The method is generally designated **1400**. Beginning at block **1402**, a controller may determine a run queue value for a zeroth core. The run queue value may indicate the workload for the zeroth core. At block **1404**, the controller may determine an operating frequency for the zeroth core. Moreover, at block **1406**, the controller may determine a utilization percentage for the zeroth core. In a particular aspect, the utilization percentage for the zeroth core may be the current operating frequency divided by the maximum operating frequency for the zeroth core. At block **1408**, the controller may determine an idle percentage for the zeroth core.

[0077] Moving to block **1410**, the controller may determine a run queue value for an Nth core. At block **1412**, the controller may determine an operating frequency for the Nth core. Further, at block **1414**, the controller may determine a utilization percentage for the Nth core. At block **1416**, the controller may determine an idle percentage for the Nth core.

[0078] Moving to block **1418**, the controller may determine a load value for the system. The load value may be determined based on the run queue value for the zeroth core, the run queue value for the Nth core, the operating frequency of the zeroth core, the operating frequency of the Nth core, the utilization percentage of the zeroth core, the utilization percentage of the Nth core, the idle percentage for the zeroth core, the idle percentage for the Nth core, or any combination thereof.

[0079] In a particular aspect, the load value may be determined from the following formula:

$$LV=RQ_0*((100\%-IP_0)*(UP_0))+RQ_N*((100\%-IP_N)*(UP_N))$$

[0080] where,

[0081] LV=a load value,

[0082] RQ_0=a run queue value for the zeroth core,

[0083] IP_0=a idle percentage for the zeroth core,

[0084] UP_0=a utilization percentage for the zeroth core,

[0085] RQ_N=a run queue value for the Nth core,

[0086] IP_N=a idle percentage for the Nth core, and

[0087] UP_N=a utilization percentage for the Nth core.

[0088] At block **1420**, one or more cores may be turned on or off based on the load value. It may be determined whether the load value meets a predetermined condition in order to determine whether to turn the one or more cores on or off. For example, the load value may be compared to a threshold and if the load value exceeds the threshold one or more cores may be turned on or off. For example, if there were two cores operating and the load value indicated that the two cores operating at or near capacity, a third core may be powered on and the workload may be spread between the three cores. Further, as the load value decreases below another threshold indicating that having a third core operational is wasting power, the third core may be powered off and the workload may be spread across the two remaining cores. In another aspect, the load value may indicate an actual number of cores that should be running in order to provide the most efficient operation of the system.

[0089] In a particular aspect, the controller may maintain historic data regarding the operation of the system and using the historic data, the controller may implement a filter, e.g., an IIR filter, the controller may implement a moving average function, or the controller may implement a combination thereof. An exemplary moving average function is shown below:

$$\begin{aligned} N &= f(\text{run } Q0, \text{run } Q1, N1, N2, N3) \\ &= 2 \text{ if } ((\text{run } Q0 + \text{run } Q1) + N1 + N2 + N3)/4 > 1, \text{ and} \\ &= 1 \text{ if } ((\text{run } Q0 + \text{run } Q1) + N1 + N2 + N3)/4 <= 1, \end{aligned}$$

[0090] where,

[0091] runQ0=a run queue value for a zeroth core;

[0092] runQ1=a run queue value for a first core; and

[0093] N1, N2, N3=the values of N for the last three historical periods.

[0094] It is to be understood that the method steps described herein need not necessarily be performed in the order as described. 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 method steps. Moreover, the methods described herein are described as executable on a portable computing device (PCD). The PCD may be a mobile telephone device, a portable digital assistant device, a smartbook computing device, a netbook computing device, a laptop computing device, a desktop computing device, or a combination thereof.

[0095] With the configuration described herein, the system and method disclosed may reduce power consumption within a multicore CPU by powering additional CPU cores when there is likelihood of actual speedup due to workload parallelism. While the degree of parallelism may not be known a-priori, the degree of parallelism may be inferred by observing an operating system scheduler run queue length. Based on the inferred degree of parallelism, additional cores may be power up or down as needed.

[0096] Unlike traditional systems, which power all CPU cores symmetrically, i.e., all on or all off, the present system and method may asymmetrically power CPU cores on and off. In other words, one core may be powered on while another is powered off. Further, as the workload increases and has sufficient parallelism, the second core may be powered on.

[0097] Further, the system and methods disclosed herein may dynamically adapt the number of powered cores to the actual dynamic parallelism offered in the workload. This may substantially power savings. Further, the system and method does not require developers to communicate the parallelism in their workloads in any special way. Applications may not need to be modified in any way and applications may be deployed in a typical manner. Moreover, the present system and methods are compatible with existing applications and devices.

[0098] In a particular aspect, the input to a MP controller is the number of ready-to-run threads in the scheduler queue. It may be appreciated that other inputs that indicate parallelism in the workload may be used. However, a core(s) idle/busy indicator, which does not indicate parallelism, may not be not a sufficient input to the MP controller. For example, a CPU core may be one hundred percent (100%) busy, but running a single-threaded workload. In that case, powering up the second core does not help the performance of the single-thread at all. In fact, it may hurt the performance by introducing overhead necessary to power up the second core. Worse, powering a second core for a single-threaded workload may introduce wasteful leakage power consumption from the second core.

[0099] Accordingly, in a particular aspect, it may be undesirable to power up the second core without assurance that there is sufficient parallel work for it to do. In an instantaneous sense, parallel work is only possible when there are two or more threads which are ready to run in the OS scheduler queues.

[0100] Under normal operation, in a dual-core example, the system may start out with a DCVS algorithm operating on zeroth CPU core, while a first CPU core remains dormant. In this mode, the system may respond to instantaneous transient loads by ramping up the frequency of the zeroth CPU core. The system ma remain in the single-core mode for as long as there are insufficient ready-to-run threads.

[0101] However, once the ready-to-run threshold N_w has been exceeded for a certain amount of time, the zeroth CPU core may be implicitly saturated and there is also clearly a parallel workload available for the first CPU core. A multicore processor (MP) controller may invoke the OS to activate the first CPU core and add the first CPU core to the OS's set of schedulable resources. The MP controller may also starts out the first CPU core at the optimal voltage-frequency point, i.e., the highest frequency at the lowest voltage, and activates the DCVS local to the first CPU core. The MP controller may optionally also reset the frequency of the zeroth CPU core to the optimal voltage-frequency point. Otherwise, the MP controller may leave the frequency of the zeroth CPU core at the current setting.

[0102] Both cores may now operate with individually executing DCVS algorithms. With sustained parallel workloads, it may be likely that both cores will eventually attain maximum frequency. However, there may be very brief lulls in load which may allow the cores to partially retreat from the maximum frequency. However, such lulls are not possible without also reducing the number of threads that are ready to run to 0.

[0103] Once enough such lulls have occurred, causing the average number of ready-to-run threads to drop below the N_s threshold for a long enough duration, the MP controller may invokes the OS to save any appropriate state from the first CPU core and remove it from its schedulable set. Subsequently, the first CPU core may be safely power collapsed. With the first CPU core power collapsed, the system may return to a single-core operating mode.

[0104] 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 on or transmitted over as one or more instructions or code on a computer program product such as a machine readable medium, i.e., a computer-readable medium. Computer-readable media includes both computer storage media and communication media including any medium that facilitates transfer of a computer program from one place to another. A storage media may be any available 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. Also, any connection is properly termed a computer-readable medium. For example, if the software is transmitted from a website, server, or other remote source using a coaxial cable, fiber optic cable, twisted pair, digital subscriber line (DSL), or wireless technologies such as infrared, radio, and microwave, then the coaxial cable, fiber optic cable, twisted pair, DSL, or wireless technologies such as infrared, radio, and microwave are included in the definition of medium. 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.

[0105] 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.

What is claimed is:

1. A method of dynamically controlling power within a multicore CPU, the method comprising:

receiving a degree of parallelism in a workload of a zeroth core; and

determining whether the degree of parallelism in the workload of the zeroth core is equal to a first wake condition.

2. The method of claim **1**, wherein the degree of parallelism comprises a total number of tasks, threads, or a combination thereof waiting on one or more ready-to-run queues of a operating system scheduler, a number of tasks, threads, or a combination thereof actually running on the zeroth core, or a combination thereof.

3. The method of claim **1**, further comprising:

determining a time duration for which the first wake condition is met when the degree of parallelism in the workload of the zeroth core is equal to the first wake condition; and

determining whether the time duration is equal to a first confirm wake condition.

4. The method of claim **3**, further comprising:

invoking an operating system to power up a first core when the time duration is equal to the first confirm wake condition.

5. The method of claim **4**, further comprising:

invoking the operating system to add the first core to a set of schedulable resources.

6. The method of claim **5**, further comprising:

receiving a degree of parallelism in a workload on the zeroth core and the first core; and

determining whether the degree of parallelism in the workload on the zeroth core and the first core is equal to a first sleep condition.

7. The method of claim **6**, further comprising:

determining a time duration for which the first sleep condition is met when the degree of parallelism in the workload on the zeroth core and the first core is equal to the first sleep condition; and

determining whether the time duration for which the first sleep condition is met is equal to a first confirm sleep condition.

8. The method of claim **7**, further comprising:

invoking the operating system to save a current state of the first core, when the time duration for which the first sleep condition is met is equal to the first sleep condition.

9. The method of claim **8**, further comprising:

invoking the operating system to power down the first core.

10. The method of claim **9**, further comprising:

invoking the operating system to remove the first core from the set of schedulable resources.

11. A wireless device, comprising:

means for receiving a degree of parallelism in a workload of a zeroth core; and

means for determining whether the degree of parallelism in the workload of the zeroth core is equal to a first wake condition.

12. The wireless device of claim **11**, wherein the degree of parallelism comprises a total number of tasks, threads, or a combination thereof waiting on one or more ready-to-run queues of a operating system scheduler, a number of tasks, threads, or a combination thereof actually running on the zeroth core, or a combination thereof.

13. The wireless device of claim **11**, further comprising:

means for determining a time duration for which the first wake condition is met when the degree of parallelism in the workload of the zeroth core is equal to the first wake condition; and

means for determining whether the time duration is equal to a first confirm wake condition.

14. The wireless device of claim **13**, further comprising:

means for invoking an operating system to power up a first core when the time duration is equal to the first confirm wake condition.

15. The wireless device of claim **14**, further comprising:

means for invoking the operating system to add the first core to a set of schedulable resources.

16. The wireless device of claim **15**, further comprising:

means for receiving a degree of parallelism in a workload on the zeroth core and the first core; and

means for determining whether the degree of parallelism in the workload on the zeroth core and the first core is equal to a first sleep condition.

17. The wireless device of claim **16**, further comprising:

means for determining a time duration for which the first sleep condition is met when the degree of parallelism in the workload on the zeroth core and the first core is equal to the first sleep condition; and

means for determining whether the time duration for which the first sleep condition is met is equal to a first confirm sleep condition.

18. The wireless device of claim **17**, further comprising:

means for invoking the operating system to save a current state of the first core, when the time duration for which the first sleep condition is met is equal to the first sleep condition.

19. The wireless device of claim **18**, further comprising:

means for invoking the operating system to power down the first core.

20. The wireless device of claim **19**, further comprising:

means for invoking the operating system to remove the first core from the set of schedulable resources.

21. A wireless device, comprising:

a processor, wherein the processor is operable to:

receive a degree of parallelism in a workload of a zeroth core; and

determine whether the degree of parallelism in the workload of the zeroth core is equal to a first wake condition.

22. The wireless device of claim **21**, wherein the degree of parallelism comprises a total number of tasks, threads, or a combination thereof waiting on one or more ready-to-run queues of a operating system scheduler, a number of tasks, threads, or a combination thereof actually running on the zeroth core, or a combination thereof.

23. The wireless device of claim **21**, wherein the processor is further operable to:

determine a time duration for which the first wake condition is met when the degree of parallelism in the workload of the zeroth core is equal to the first wake condition; and

determine whether the time duration is equal to a first confirm wake condition.

24. The wireless device of claim **23**, wherein the processor is further operable to:

invoke an operating system to power up a first core when the time duration is equal to the first confirm wake condition.

25. The wireless device of claim **24**, wherein the processor is further operable to:

invoke the operating system to add the first core to a set of schedulable resources.

26. The wireless device of claim **25**, wherein the processor is further operable to:

receive a degree of parallelism in a workload on the zeroth core and the first core; and

determine whether the degree of parallelism in the workload on the zeroth core and the first core is equal to a first sleep condition.

27. The wireless device of claim **26**, wherein the processor is further operable to:

determine a time duration for which the first sleep condition is met when the degree of parallelism in the workload on the zeroth core and the first core is equal to the first sleep condition; and

determine whether the time duration for which the first sleep condition is met is equal to a first confirm sleep condition.

28. The wireless device of claim **27**, wherein the processor is further operable to:

invoke the operating system to save a current state of the first core, when the time duration for which the first sleep condition is met is equal to the first sleep condition.

29. The wireless device of claim **28**, wherein the processor is further operable to:

invoke the operating system to power down the first core.

30. The wireless device of claim **29**, wherein the processor is further operable to:

invoke the operating system to remove the first core from the set of schedulable resources.

31. A memory medium, comprising:

at least one instruction for receiving a degree of parallelism in a workload of a zeroth core; and

at least one instruction for determining whether the degree of parallelism in the workload of the zeroth core is equal to a first wake condition.

32. The memory medium of claim **31**, wherein the degree of parallelism comprises a total number of tasks, threads, or a combination thereof waiting on one or more ready-to-run queues of a operating system scheduler, a number of tasks, threads, or a combination thereof actually running on the zeroth core, or a combination thereof.

33. The memory medium of claim **31**, further comprising:

at least one instruction for determining a time duration for which the first wake condition is met when the degree of parallelism in the workload of the zeroth core is equal to the first wake condition; and

at least one instruction for determining whether the time duration is equal to a first confirm wake condition.

34. The memory medium of claim **33**, further comprising:

at least one instruction for invoking an operating system to power up a first core when the time duration is equal to the first confirm wake condition.

35. The memory medium of claim **34**, further comprising:

at least one instruction for invoking the operating system to add the first core to a set of schedulable resources.

36. The memory medium of claim **35**, further comprising:

at least one instruction for receiving a degree of parallelism in a workload on the zeroth core and the first core; and

at least one instruction for determining whether the degree of parallelism in the workload on the zeroth core and the first core is equal to a first sleep condition.

37. The memory medium of claim **36**, further comprising:

at least one instruction for determining a time duration for which the first sleep condition is met when the degree of parallelism in the workload on the zeroth core and the first core is equal to the first sleep condition; and

at least one instruction for determining whether the time duration for which the first sleep condition is met is equal to a first confirm sleep condition.

38. The memory medium of claim **37**, further comprising:

at least one instruction for invoking the operating system to save a current state of the first core, when the time duration for which the first sleep condition is met is equal to the first sleep condition.

39. The memory medium of claim **38**, further comprising:

at least one instruction for invoking the operating system to power down the first core.

40. The memory medium of claim **39**, further comprising:

at least one instruction for invoking the operating system to remove the first core from the set of schedulable resources.

* * * * *