



- (51) **International Patent Classification:**
G06F 1/26 (2006.01) *G06F 1/08* (2006.01)
- (21) **International Application Number:**
PCT/US2013/062024
- (22) **International Filing Date:**
26 September 2013 (26.09.2013)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (30) **Priority Data:**
61/735,944 11 December 2012 (11.12.2012) US
13/913,307 7 June 2013 (07.06.2013) US
- (71) **Applicant:** APPLE INC. [US/US]; 1 Infinite Loop, Cupertino, California 95014 (US).
- (72) **Inventors:** DORSEY, John G.; 1 Infinite Loop, Cupertino, California 95014 (US). ISMAIL, James S.; 1 Infinite Loop, Cupertino, California 95014 (US). COX, Keith; 1 Infinite Loop, Cupertino, California 95014 (US). KA-POOR, Gaurav; 1 Infinite Loop, Cupertino, California 95014 (US).
- (74) **Agents:** FERRAZANO, Michael J. et al.; Womble Carlyle Sandridge & Rice LLP, 10050 North Wolfe Road, Suite 260, Cupertino, California 95014 (US).

(81) **Designated States** (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, 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, 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, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) **Designated States** (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

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

(54) **Title:** CLOSED LOOP CPU PERFORMANCE CONTROL

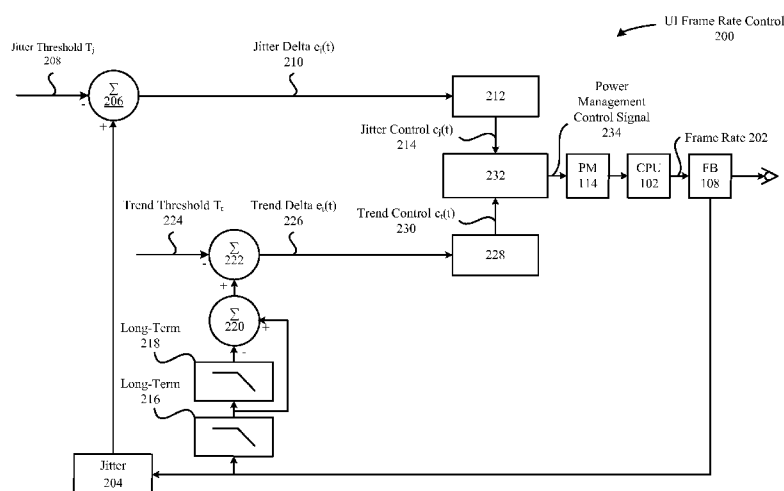


FIG. 2A

(57) **Abstract:** The invention provides a technique for targeted scaling of the voltage and/or frequency of a processor included in a computing device. One embodiment involves scaling the voltage/frequency of the processor based on the number of frames per second being input to a frame buffer in order to reduce or eliminate choppiness in animations shown on a display of the computing device. Another embodiment of the invention involves scaling the voltage/frequency of the processor based on a utilization rate of the GPU in order to reduce or eliminate any bottleneck caused by slow issuance of instructions from the CPU to the GPU. Yet another embodiment of the invention involves scaling the voltage/frequency of the CPU based on specific types of instructions being executed by the CPU. Further embodiments include scaling the voltage and/or frequency of a CPU when the CPU executes workloads that have characteristics of traditional desktop/laptop computer applications.

CLOSED LOOP CPU PERFORMANCE CONTROL

TECHNICAL FIELD

[0001] The present invention relates generally to power management in a mobile
5 computing device. More particularly, the present invention relates to power
management techniques that correlate to central processor unit (CPU) activity,
memory controller (MC) activity, graphics processing unit (GPU) activity, and user
interface (UI) frame rate activity within the mobile computing device.

BACKGROUND

10 [0002] Conventional computing devices (e.g., desktop computers and laptop
computers) typically implement one or more algorithms directed to controlling the
operating clock frequency and voltage of processors included therein, such as a CPU
and a GPU. These algorithms are directed to monitoring the CPU/GPU for workloads
that take more than a threshold amount of time to complete. Consider, for example, a
15 time-intensive image processing workload that takes several minutes for a CPU/GPU
to execute when the CPU/GPU are in a low-performance operating mode. In this
example, the algorithms detect that the workload meets certain criteria (e.g., the
threshold amount of time has passed or processor duty factor has exceeded a
threshold) and cause the CPU/GPU to switch from a low-performance operating
20 mode to a mid-performance or a high-performance operating mode so that the
workload is completed sooner. These algorithms enable conventional computing
devices to reduce power for short, bursty workloads while providing high
performance for long-running compute tasks.

[0003] Notably, recent years have shown a proliferation in the usage of mobile
25 computing devices with performance characteristics, energy constraints and
interactive user interfaces that are different from those of desktop/laptop computers,
which affect the types of workloads users execute on mobile devices. More
specifically, unlike traditional long-running pure-compute tasks, mobile applications
instead emphasize interactive performance for visual scenarios such as web browsing,
30 gaming and photography. Consequently, the aforementioned algorithms—which are
directed to identifying and responding to complex, time-intensive workloads—are not
as effective when implemented in mobile devices since the algorithms cannot
accurately determine when the operating mode of the CPU/GPU should be modified.

SUMMARY

[0004] This paper describes various embodiments that relate to the operation of CPU performance control algorithms within a mobile computing device. In contrast to conventional approaches, these performance control algorithms can operate based on feedback received from various components included in the mobile computing device, such as a frame buffer, a GPU and a memory controller. For example, instead of focusing solely on the amount of time a workload spends executing on the CPU, the techniques presented herein measure the smoothness of UI animations presented on a display of the mobile computing device, the utilization rates of the GPU or memory interfaces, and the types of instructions being executed by the CPU.

[0005] One embodiment of the invention sets forth a method for updating an operating mode of a processor. The method includes the steps of monitoring a cycle-to-cycle jitter associated with a rate by which a user interface (UI) is animated, and, further, adjusting an operating mode of the processor based on the cycle-to-cycle jitter. Adjusting the operating mode of the processor comprises adjusting the voltage and/or frequency at which the processor is operating. Moreover, monitoring the cycle-to-cycle jitter comprises analyzing a rate of change in a number of frames per second (NFPS) being input to a frame buffer associated with the processor. Further, monitoring the cycle-to-cycle jitter comprises establishing: a jitter control signal based on short-term sampling of the NFPS being input to the frame buffer, and a trend control signal based on long-term sampling of the NFPS being input to the frame buffer.

[0006] Another embodiment of the invention sets forth a method for optimizing operations of a CPU in a mobile computing device having the CPU configured to issue instructions to a GPU. The method includes the steps of determining that a utilization rate of the GPU is exceeding a threshold level, determining that the CPU is operating in a sub-optimal operating mode, and causing the CPU to enter into an optimal operating mode where the CPU generates instructions for execution by the GPU at a faster rate. Causing the CPU to enter into the optimal operating mode includes establishing a control signal by a control signal generator. Causing the CPU to enter into the optimal operating mode includes adjusting the voltage and/or frequency at which the CPU is operating. The method can further include the steps of determining that the utilization rate of the GPU is no longer exceeding the threshold level, and causing the CPU to return to a more energy-efficient operating mode.

- [0007]** A third embodiment of the invention sets forth a method for updating an operating mode of a CPU. The method includes the steps of determining that the CPU is tasked with executing instructions that are associated with a high instruction-per-cycle density, and causing the CPU to enter into a high-performance operating mode to cause an increase in the rate at which the CPU executes the instructions. The instructions can comprise integer arithmetic instructions, vector floating point (VFP) arithmetic instructions, single-instruction multiple-data (SIMD) arithmetic instructions, and load-store instructions. The method can further include the steps of establishing: an integer arithmetic control signal based on the rate at which integer arithmetic instructions are being executed by the CPU, a VFP control signal based on the rate at which VFP arithmetic instructions are being executed by the CPU, a SIMD control signal based on the rate at which SIMD arithmetic instructions are being executed by the CPU, and a load-store control signal based on the rate at which load-store instructions are being executed by the CPU.
- [0008]** Yet another embodiment of the invention sets forth a method for optimizing operations of a CPU in a mobile computing device having the CPU configured to perform transactions with a memory controller that manages access to a dynamic random-access memory (DRAM) and a flash memory. The method includes the steps of determining that the data throughputs of memory controller exchanges with one or more agents are exceeding threshold levels, determining that the CPU is operating in a sub-optimal operating mode, and causing the CPU to enter into an optimal operating mode where the CPU performs transactions with the memory controller at a faster rate. The agents can include the CPU itself or a flash memory subsystem.
- [0009]** An additional embodiment of the invention sets forth a method for updating an operating mode of a CPU while executing workloads that have characteristics of traditional desktop/laptop computer applications. The method includes the steps of determining that a utilization rate of the CPU is exceeding a threshold level, determining that the interactive user interface is updating below a threshold rate, and causing the CPU to enter into a high-performance operating mode to allow the mobile device to complete the task more quickly.
- [0010]** Other embodiments include a non-transitory computer readable medium storing instructions that, when executed by a processor, cause the processor to carry out any of the method steps described above. Further embodiments include a system

that includes at least a processor and a memory storing instructions that, when executed by the processor, cause the processor to carry out any of the method steps described above. Yet other embodiments include a system having a management co-processor separate from the main CPU capable of, either in cooperation with or in place of the main CPU, carrying out any of the method steps described above.

[0011] Other aspects and advantages of the invention will become apparent from the following detailed description taken in conjunction with the accompanying drawings which illustrate, by way of example, the principles of the described embodiments.

10 **BRIEF DESCRIPTION OF THE DRAWINGS**

[0012] The included drawings are for illustrative purposes and serve only to provide examples of possible structures and arrangements for the disclosed inventive apparatuses and methods for providing portable computing devices. These drawings in no way limit any changes in form and detail that may be made to the invention by one skilled in the art without departing from the spirit and scope of the invention. The embodiments will be readily understood by the following detailed description in conjunction with the accompanying drawings, wherein like reference numerals designate like structural elements.

[0013] Figure 1 illustrates a block diagram of a mobile computing device configured to implement embodiments of the invention.

[0014] Figure 2A illustrates a conceptual diagram of an embodiment directed to scaling the voltage and/or frequency of a CPU based on the NFPS being supplied to a frame buffer.

[0015] Figure 2B illustrates a method for updating an operating mode of a CPU based on monitoring a cycle-to-cycle jitter associated with a rate by which a user interface (UI) is refreshed, according to one embodiment of the invention.

[0016] Figures 2C-2E illustrate a method for scaling the voltage and/or frequency of a CPU based on the NFPS being input to a frame buffer, according to one embodiment of the invention.

30 [0017] Figure 3A illustrates a conceptual diagram of an embodiment directed to scaling the voltage and/or frequency of a CPU based on a utilization rate of a GPU.

[0018] Figure 3B illustrates a method for entering a CPU into an optimal operating mode based on a utilization rate of a GPU, according to one embodiment of the invention.

[0019] Figure 3C illustrates a method for scaling the power and/or frequency of a CPU based on a utilization rate of a GPU, according to one embodiment of the invention.

5 [0020] Figure 4A illustrates a conceptual diagram of an embodiment directed to analyzing the rate at which certain types of instructions are being executed by a CPU and scaling the voltage and/or frequency of the CPU based on the rate.

[0021] Figures 4B-4E illustrate a method for analyzing the rate at which certain types of instructions are being executed by a CPU and scaling the voltage and/or frequency of the CPU based on the rate, according to one embodiment of the invention.

[0022] Figure 5A illustrates a conceptual diagram of an embodiment directed to analyzing a rate at which a CPU performs transactions with a memory controller that manages access to a DRAM and a flash memory, according to one embodiment of the invention.

15 [0023] Figure 5B illustrates a method for optimizing operations of a CPU in a mobile computing device having the CPU configured to perform transactions with a memory controller that manages access to a DRAM and a flash memory, according to one embodiment of the invention.

[0024] Figure 6A illustrates a conceptual diagram of an embodiment directed to scaling the voltage and/or frequency of a CPU when the CPU executes workloads that have characteristics of traditional desktop/laptop computer applications, according to one embodiment of the invention.

[0025] Figure 6B illustrates a method for scaling the voltage and/or frequency of a CPU when the CPU executes workloads that have characteristics of traditional desktop/laptop computer applications, according to one embodiment of the invention.

[0026] In the figures, elements referred to with the same or similar reference numerals include the same or similar structure, use, or procedure, as described in the first instance of occurrence of the reference numeral.

DETAILED DESCRIPTION

30 [0027] Representative applications of apparatuses and methods according to the presently described embodiments are provided in this section. These examples are being provided solely to add context and aid in the understanding of the described embodiments. It will thus be apparent to one skilled in the art that the presently described embodiments can be practiced without some or all of these specific details.

In other instances, well known process steps have not been described in detail in order to avoid unnecessarily obscuring the presently described embodiments. Other applications are possible, such that the following examples should not be taken as limiting.

5 **[0028]** In the following detailed description, references are made to the accompanying drawings, which form a part of the description and in which are shown, by way of illustration, specific embodiments in accordance with the described embodiments. Although these embodiments are described in sufficient detail to enable one skilled in the art to practice the described embodiments, it is understood
10 that these examples are not limiting; such that other embodiments may be used, and changes may be made without departing from the spirit and scope of the described embodiments.

[0029] As set forth above, embodiments of the invention are directed to scaling the voltage and/or frequency of a CPU included in a mobile computing device. In
15 particular, the embodiments are directed to alleviating a variety of performance and energy efficiency issues that are often exhibited by mobile computing devices and that are not well-addressed by conventional power-management techniques. As set forth in greater detail below, the embodiments alleviate these performance and energy efficiency issues by implementing techniques that focus on various aspects of how
20 processes are carried out within the mobile computing device. For example, one embodiment of the invention involves scaling the voltage and/or frequency of the CPU based on the number of frames per second (NFPS) being input to a frame buffer in order to reduce or eliminate choppiness in animations shown on a display of the mobile computing device. Another embodiment of the invention involves scaling the
25 voltage and/or frequency of the CPU based on a utilization rate of the GPU in order to reduce or eliminate any bottleneck caused by slow issuance of instructions from the CPU to the GPU. A third embodiment of the invention involves scaling the voltage and/or frequency of the CPU based on the throughput of data transiting the memory controller. A fourth embodiment of the invention involves scaling the voltage and/or
30 frequency of the CPU based on specific types of instructions being executed by the CPU. An additional embodiment of the invention involves scaling the voltage and/or frequency of the CPU based on the duty factor of the CPU for workloads that cause a user interface to animate below a threshold rate.

[0030] As noted above, one existing performance issue exhibited by mobile computing devices involves the smoothness of animations shown on a display of the mobile computing device. For example, choppy display of an animation (e.g., scrolling of a web page) contributes to a poor user experience and should be eliminated whenever possible. Accordingly, one embodiment of the invention involves scaling the voltage and/or frequency of the CPU based on the number of frames-per-second (NFPS) being supplied to a frame buffer included in the mobile computing device. In particular, a monitor measures short-term “jitter” in the NFPS being input to the frame buffer as well as the long-term stability of the NFPS being input to the frame buffer to determine whether the CPU is operating at a power and/or frequency sufficient to produce smooth animations. More specifically, when the monitor observes changes in the NFPS, the monitor increases the voltage and/or frequency of the CPU to smooth out the NFPS. Conversely, when the monitor observes that the NFPS is stable, the monitor decreases the voltage and/or frequency of the CPU in order to conserve energy.

[0031] Another existing performance issue exhibited by mobile computing devices involves the bottleneck that often occurs between instructions issued by CPU to the GPU. For example, the CPU can operate in a sub-optimal mode when the activity of the GPU is such that the GPU requests new instructions from the CPU at a rate faster than the CPU can produce the new instructions (e.g., during GPU benchmark tests). While in the sub-optimal mode, the CPU can operate at a sub-optimal voltage and/or frequency. Accordingly, when it is determined that a utilization rate of the GPU is exceeding a threshold level and that the CPU is operating in a sub-optimal operating mode, the CPU can be configured to enter into an optimal operating mode, which increases the rate at which the CPU generates instructions to be executed by the GPU. Adjusting the operating mode of the CPU can include adjusting the voltage and/or frequency at which the CPU is operating. Later, when it can be determined that the utilization rate of the GPU is no longer exceeding the threshold level, the CPU can be configured to enter back into the sub-optimal operating mode. In this manner, critical GPU workloads—such as graphics benchmarks—are not hindered by the CPU when the CPU is operating at a sub-optimal voltage and/or frequency.

[0032] Yet another existing performance issue exhibited by mobile computing devices involves the bottleneck that is introduced when the CPU manages data flows

between the memory controller and one or more memory agents. For example, the CPU can operate in a sub-optimal mode when encoding a video and writing the result to DRAM at a rate slower than a rate at which the memory interface is capable of operating. As another example, the CPU can operate in a sub-optimal mode when
5 executing a flash memory subsystem device driver for the purpose of reading data from the flash memory subsystem, where executing the driver slowly may increase access latency. While in the sub-optimal mode, the CPU can operate at a sub-optimal voltage and/or frequency. Accordingly, when it is determined that a read or write throughput of traffic between the CPU and the memory controller is exceeding
10 threshold levels—or, that the read or write throughput of traffic between the flash memory subsystem and the memory controller is exceeding threshold levels, and that the CPU is operating in a sub-optimal operating mode—the CPU can be configured to enter into an optimal operating mode. Later, when it can be determined that these throughputs are no longer exceeding the threshold levels, the CPU can be configured
15 to enter back into the sub-optimal operating mode.

[0033] Another existing energy efficiency issue exhibited by mobile computing devices involves erroneously increasing the voltage and/or frequency of the CPU solely based on the utilization rate of the CPU. For example, a simple spin loop workload—such as a loop that frequently checks for a specific condition to be met—
20 may increase the utilization rate of the CPU to 99% and cause, via conventional algorithms, the voltage and/or frequency of the CPU to be increased. Importantly, in this example, such an increase in no way promotes a faster completion of the spin loop, so energy is wasted in doing so. However, some specific workloads executed by the CPU—such as those involving integer arithmetic, VFP arithmetic, SIMD
25 arithmetic and load-store operations—can benefit from an increase in the voltage and/or frequency of the CPU. Accordingly, yet another embodiment of the invention involves the monitor analyzing the rate at which certain types of instructions are being executed by the CPU and scaling the voltage and/or frequency of the CPU based on the rate. In this manner, wasteful CPU performance increases can be
30 avoided, thereby saving energy.

[0034] Another existing energy efficiency issue exhibited by mobile computing devices involves erroneously increasing the voltage and/or frequency of the CPU solely based on the utilization rate of the CPU while executing workloads for which a user is not waiting for results. For example, a video game containing a spinloop may

exhibit high CPU utilization, but increasing the voltage and/or frequency of the CPU by conventional algorithms will increase CPU power while possibly not increasing the NFPS produced by the game. By contrast, a circuit layout optimization tool may also exhibit high CPU utilization, but here increasing the voltage and/or frequency of the CPU by conventional algorithms may reduce the time a user must wait for the result of the computation. Accordingly, another embodiment of the invention involves monitoring the UI frame rate and increasing the voltage and/or frequency of the CPU by conventional algorithms only when the UI frame rate is below a threshold and the CPU utilization is above a threshold. When the UI is not a principal component of the workload, this embodiment permits the energy budget of the mobile device to be biased towards CPU-based computation.

[0035] According to these techniques, the default operating mode of the CPU is minimum performance, and increased performance is provided only to workloads that can benefit from such an increase. For example, by scaling the operating mode of the CPU relative to UI frame rate smoothness, interactive applications that do not benefit from the higher performance of newer CPUs can save tens to thousands of milliwatts relative to existing power management algorithms. At the same time, high-end graphical applications are permitted to access the full compute performance of both the CPU and the GPU. High-throughput data transaction workloads are similarly permitted to access the full compute performance of the CPU. Monitoring arithmetic and load-store instruction densities enables a distinction to be established between workloads that perform useful computations with additional CPU performance and those that do not, thereby saving hundreds of milliwatts on applications that task the CPU with executing “busy loops.” Finally, by considering the UI frame rate, the use of conventional algorithms that increase CPU performance in response to high CPU utilization can be enabled when the user is waiting for a time-intensive computation to complete, but disabled for animation workloads that may not benefit from increased performance.

[0036] As set forth above, various embodiments of the invention are directed to scaling of the voltage and/or frequency of a CPU included in a mobile computing device. A detailed description of the embodiments is provided below in conjunction with Figures 1, 2A-2E, 3A-3C, and 4A-4E. In particular, Figure 1 illustrates a block diagram of a mobile computing device 100 configured to implement embodiments of the invention. As shown in Figure 1, mobile computing device 100 includes

subsystems such as CPU 102, a memory controller 103, a system memory 104, GPU 106, frame buffer 108, and display device 110. As is well-known, CPU 102 generates and transmits instructions 103 to GPU 106 for execution, where GPU 106 consumes the instructions at a rate that is influenced at least by the utilization rate of GPU 106 and a rate at which CPU 102 is generating and transmitting the instructions 103 to GPU 106. Frame buffer 108 is configured to continually receive and store an updated sequence of frames that are eventually output to display device 110. Also shown in Figure 1 are monitor 112 and power manager 114, which are loaded in system memory 104 and configured to execute on mobile computing device 100. In one embodiment, system memory 104 include both a DRAM subsystem (not illustrated) and a flash memory subsystem (not illustrated) that are managed by the memory controller 103. Although not illustrated in Figure 1, each of monitor 112 and power manager 114 can run on an operating system (OS) that is configured to execute on mobile computing device 100. Additionally, monitor 112 and power manager 114 can run on a management co-processor (not illustrated) that is separate and distinct from the CPU 102.

[0037] As described in greater detail below, monitor 112 is configured to implement various techniques directed toward identifying circumstances where a change in the voltage and/or frequency of CPU 102 is beneficial to the overall performance of mobile computing device 100 and energy savings within mobile computing device 100. In particular, monitor 112 receives, from a number of controllers, control signals that scale with a focus on a particular activity within mobile computing device 100, e.g., the rate of change in the NFPS being input to the frame buffer 108, the utilization rate of GPU 106, the data throughputs of the memory controller 103, the rate at which specific types of instructions are being executed by CPU 102, or the rate at which a user interface is being updated. In turn, monitor 112 processes the control signals and outputs the control signals to power manager 114, whereupon the power manager 114 correspondingly scales the voltage and/or frequency of CPU 102. For example, one control signal can slowly increase in value (e.g., the utilization rate of GPU 106) and cause the power manager 114 to correspondingly increase the voltage and/or frequency of CPU 102, thereby reducing or eliminating a potential bottleneck that might occur between the rate at which GPU 106 is able to consume instructions issued by CPU 102.

[0038] In one embodiment, each controller produces a scalar value—also referred to herein as a “control effort”—that takes on a value from 0 to 255, where larger values are expressions of a desire for higher performance. Each of the controllers produces a control effort value independently of the other controllers. As described in greater detail below, the control effort value that determines the CPU 102 performance configuration is the maximum of the individual control efforts. Given the winning control effort, the mapping to a CPU 102 performance configuration may vary. In one embodiment, the range 0–255 may be linearly mapped to qualified CPU 102 frequencies. In a related embodiment, the mapping may instead be linear in CPU 102 voltage rather than frequency. In another embodiment, the mapping may involve the use of frequency/voltage dithering to produce a more precise mapping through pulse width modulation techniques. In yet another embodiment, the mapping may also determine the number of CPU 102 cores that may be concurrently active in a multi-core environment. For example, a lower control effort value may restrict the mobile computing device 100 to single-core operation as a means of conserving energy. In yet another embodiment, the mapping may also determine the selection of a primary core or secondary core, where the primary core is more powerful than the secondary core and is configured to operate during high demand periods, and where the secondary core is less powerful than the primary core and is configured to operate during low demand periods.

[0039] Figure 2A illustrates a conceptual diagram 200 of the embodiment directed to scaling the voltage and/or frequency of CPU 102 based on the NFPS being input to frame buffer 108. As shown in Figure 2A, the NFPS being input to frame buffer 108 is represented by frame rate 202, which is analyzed by monitor 112 and observed by a user of mobile computing device 100 via display device 110. Jitter component 204, which is managed by monitor 112, is configured to analyze (via the outermost loop of Figure 2A) short-term changes (i.e., cycle-to-cycle jitter) in the NFPS being input to frame buffer 108 within a short-term threshold amount of time. Notably, the NFPS being input to frame buffer 108 is correlated to the smoothness of user interfaces (UIs) that are displayed on display device 110, which significantly impacts overall user experience. In one embodiment, the cycle-to-cycle jitter is defined as the difference in instantaneous frame rates over two consecutive frame rate samples. Consider, for example, the absolute times of three sequential frame buffer 108 updates T_1 , T_2 and T_3 , where the instantaneous frame rate $F(1 \text{ to } 2) = 1/(T_2 - T_1)$ and

the instantaneous frame rate $F(2 \text{ to } 3) = 1/(T_3 - T_2)$. According to this example, the cycle-to-cycle jitter associated with this sequence is equal to the absolute value of $(F(2 \text{ to } 3) - F(1 \text{ to } 2))$, which is then output by jitter component 204 and compared at comparator 206 against a jitter threshold T_j 208 (e.g., three frames per second (FPS)).

5 **[0040]** As shown in Figure 2A, comparator 206 is configured to output a jitter delta $e_j(t)$ 210 to jitter control signal generator 212. When the output of jitter component 204 is less than jitter threshold T_j 208, the jitter delta $e_j(t)$ 210 is negative, which is what allows the comparator 206 to unwind when performance is sufficient to enable smooth animation. The jitter control signal generator 212 can be any form of a controller filter that is closed-loop stable. In one embodiment, the jitter control signal generator 212 can be an integrator that, in turn, integrates jitter deltas $e_j(t)$ 210 as they are output by comparator 206 and outputs a jitter control signal $c_j(t)$ 214. In one embodiment, jitter control signal generator 212 can be configured to apply a gain K_j to the integrated jitter deltas $e_j(t)$ 210 in order to amplify the jitter control signal $c_j(t)$ 214. Next, the jitter control signal $c_j(t)$ 214 is directed to max-selector 232, which outputs a maximum of the jitter control signal $c_j(t)$ 214, or a trend control signal $c_t(t)$ 230 that is produced according to the innermost loop of Figure 2A described below.

15 **[0041]** More specifically, the innermost loop of Figure 2A represents monitor 112 analyzing long-term changes that occur in the NFPS being input to frame buffer 108 within a long-term threshold amount of time. Specifically, a long-term sample 216 and a long-term sample 218 are analyzed at comparator 220 to produce a trend value that represents the rate of change of the NFPS being input to frame buffer 108 over the long-term threshold amount of time. The absolute value of the trend value is then compared at comparator 222 against a trend threshold T_t 224 (e.g., one FPS), and comparator 222 outputs a trend delta $e_t(t)$ 226 to trend control signal generator 228. The trend control signal generator 228 can be any form of a controller filter that is closed-loop stable. In one embodiment, the trend control signal generator 228 can be an integrator that, in turn, integrates trend deltas $e_t(t)$ 226 as they are output by comparator 222 and outputs the trend control signal $c_t(t)$ 230. The trend control signal generator 228 can also be configured to apply a gain K_t to the integrated trend deltas $e_t(t)$ 226 in order to amplify the trend control signal $c_t(t)$ 230.

20 **[0042]** In some cases, an animation can exhibit high short-term jitter but is still stable over the long term. For example, a game that is not performance-limited but that uses imprecise frame rate timing may display this behavior. To avoid

unnecessarily increasing CPU 102 performance for these cases, a linkage can exist between the jitter and trend controllers. More specifically, if the trend control effort is below some small value epsilon, a jitter value of zero (instead of the actual measured jitter sample) is supplied to the jitter delta term calculator, which has the effect of forcing the jitter loop to unwind so long as the trend loop is also unwound.

5 [0043] As noted above, max-selector 232 is configured to output a maximum of jitter control signal $c_j(t)$ 214, or trend control signal $c_t(t)$ 230, as a power management control signal 234 to power manager 114. In turn, power manager 114 scales the voltage and/or frequency of CPU 102 according to power management control signal 10 234. Accordingly, monitor 112 enables the performance of CPU 102 to scale dynamically in order to reduce or eliminate choppiness in the NFPS being input to frame buffer 108, thereby providing energy savings and enhancing overall user experience.

[0044] Notably, at some point, most animations stop. Accordingly, embodiments 15 of the invention incorporate a threshold amount of time after observing the last frame buffer 108 update (e.g., tens or hundreds of milliseconds). If no new update arrives in that time, the integrators are reset (and, therefore, the control efforts) to zero. As a result, shortly after an animation ends, the UI control loop will cease to have an influence on the operating mode of CPU 102.

20 [0045] Figure 2B illustrates a method 270 for updating an operating mode of CPU 102 based on monitoring a cycle-to-cycle jitter associated with a rate by which a user interface (UI) is refreshed, according to one embodiment of the invention. Although the method steps 270 are described in conjunction with Figures 1 and 2A, persons skilled in the art will understand that any system configured to perform the method 25 steps, in any order, is within the scope of the invention.

[0046] As shown in Figure 2B, the method 270 begins at step 272, which monitors a cycle-to-cycle jitter associated with a rate by which a user interface (UI) is refreshed. At step 274, monitor 112 adjusts an operating mode of the CPU based on the cycle-to-cycle jitter.

30 [0047] Figures 2C-2E illustrate a method 230 for scaling the voltage and/or frequency of CPU 102 based on the NFPS being input to frame buffer 108, according to one embodiment of the invention. Although the method steps 230 are described in conjunction with Figures 1 and 2A, persons skilled in the art will understand that any

system configured to perform the method steps, in any order, is within the scope of the invention.

[0048] As shown in Figure 2C, the method 230 begins at step 231, where monitor 112 is configured to monitor frames being input into the frame buffer 108. At step 5 232, monitor 112 establishes a first short-term sample of a NFPS being input into the frame buffer 108. At step 234, monitor 112 establishes a second short-term sample of the NFPS being input into the frame buffer 108. At step 236, monitor 112 establishes a jitter value by taking the absolute value of the difference between the first short-term sample and the second short-term sample. Notably, steps 231-236 represent 10 jitter component 204 described above in conjunction with Figure 2A.

[0049] At step 240, monitor 112 outputs a jitter delta value to a jitter integrator. At step 242, monitor 112 integrates, at the jitter integrator, the jitter delta value with previously-output jitter delta values to produce a jitter-based power management control signal. At step 244, monitor 112 outputs the jitter-based power management 15 control signal.

[0050] At step 245, which is illustrated in Figure 4D, monitor 112 monitors frames being input into the frame buffer 108. At step 246, monitor 112 establishes a first long-term sample of the NFPS being input to the frame buffer 108. At step 248, monitor 112 establishes a second long-term sample of the NFPS being input to the 20 frame buffer 108. At step 250, monitor 112 establishes a trend value by taking the absolute value of the difference between the first long-term sample and the second long-term sample. Notably, steps 245-250 represent long-term sample 216, long-term sample 218, and comparator 220 described above in conjunction with Figure 2A.

[0051] At step 254, monitor 112 outputs a trend delta value to a trend integrator. 25 At step 256, monitor 112 integrates, at the trend integrator, the trend delta value with previously-output trend delta values to produce a trend-based power management control signal. At step 258, monitor 112 outputs the trend-based power management control signal.

[0052] Turning now to Figure 2E, at step 260, monitor 112 determines whether 30 the jitter-based control signal is greater than the trend-based control signal. Notably, step 260 represents max-selector 232 described above in conjunction with Figure 2A. If, at step 260, monitor 112 determines that the jitter-based control signal is greater than the trend-based control signal, then the method 230 proceeds to step 262, where monitor 112 scales the power and/or frequency of CPU 102 according to the jitter-

based control signal. Otherwise, the method 230 proceeds to step 264, where monitor 112 scales the power and/or frequency of CPU 102 according to the trend-based control signal.

[0053] Figure 3A illustrates a conceptual diagram 300 of the embodiment directed to scaling the power and/or frequency of CPU 102 based on the utilization rate of GPU 106. As shown in Figure 3A, the conceptual diagram 300 includes a single loop that is directed to analyzing the utilization rate of GPU 106. In particular, GPU 106 provides GPU utilization rate feedback 302 to comparator 304, which is configured to compare the GPU utilization rate feedback 302 to a GPU utilization threshold T_g 306 (e.g., 99%).

[0054] If the GPU utilization threshold T_g 306 is exceeded by the GPU utilization rate feedback 302, then comparator 304 outputs a delta $e_g(t)$ 308 to control signal generator 310. The control signal generator 310 can be any form of a controller filter that is closed-loop stable. In one embodiment, control signal generator 310 can be an integrator that, in turn, integrates deltas $e_g(t)$ 308 as they are output by comparator 304 and outputs a GPU control signal $c_g(t)$ 312 to power manager 114. Control signal generator 310 can be configured to apply a gain K_g to the integrated deltas $e_g(t)$ 308 in order to amplify the power management control signal 314. In turn, power manager 114 receives the power management control signal 314 and accordingly scales the power and/or frequency of CPU 102. In this manner, the performance of CPU 102 scales with the utilization rate of GPU 106 so that CPU 102 is able to issue instructions at a rate that is commensurate with the rate at which GPU 106 is consuming the instructions. As a result, bottlenecks that often occur between CPU 102 and GPU 106 are reduced or eliminated, thereby enhancing overall performance of mobile computing device 100 and ensuring that the full potential of GPU 106 is not hindered by lack of CPU 102 performance.

[0055] Figure 3B illustrates a method 330 for entering CPU 102 into an optimal operating mode based on a utilization rate of GPU 106, according to one embodiment of the invention. Although the method steps 330 are described in conjunction with Figures 1 and 3A, persons skilled in the art will understand that any system configured to perform the method steps, in any order, is within the scope of the invention.

[0056] As shown in Figure 3B, the method 330 begins at step 331, where monitor 112 is configured to monitor the output of GPU 106. At step 332, monitor 112

samples a current utilization rate of GPU 106. At step 334, monitor 112 determines whether the current utilization rate exceeds a GPU utilization threshold. If, at step 334, monitor 112 determines that the current utilization rate exceeds the GPU utilization threshold, then the method 330 proceeds to step 335. At step 335, monitor
5 112 determines whether CPU 102 is operating in a sub-optimal operating mode. If, at step 335, monitor 112 determines that CPU 102 is operating in a sub-optimal operating mode, then the method 330 proceeds to step 336. Otherwise, the method 330 proceeds back to step 331, where steps 331-335 are repeated until the current utilization rate exceeds the GPU utilization threshold and CPU 102 is operating in a
10 sub-optimal operating mode. At step 336, monitor 112 causes CPU 102 to enter into an optimal operating mode where CPU 102 generates instructions for execution by GPU 106 at a faster rate.

[0057] Figure 3C illustrates a method 350 for scaling the power and/or frequency of CPU 102 based on the utilization rate of GPU 106, according to one embodiment
15 of the invention. Although the method steps 350 are described in conjunction with Figures 1 and 3A, persons skilled in the art will understand that any system configured to perform the method steps, in any order, is within the scope of the invention.

[0058] As shown in Figure 3C, the method 350 begins at step 351, where monitor
20 112 is configured to monitor the output of GPU 106. At step 352, monitor 112 samples a current utilization rate of GPU 106. At step 356, monitor 112 outputs a GPU utilization delta value to a GPU utilization integrator. Notably, steps 351-356 represent comparator 304 and control signal generator 310 described above in conjunction with Figure 3A. At step 358, monitor 112 integrates, at the GPU
25 utilization integrator, the GPU utilization delta value with previously-output GPU utilization delta values to produce a GPU utilization-based power management control signal. At step 360, monitor 112 scales the power and/or frequency of CPU 102 according to the GPU utilization-based power management control signal.

[0059] Figure 4A illustrates a conceptual diagram 400 of the embodiment directed
30 to analyzing the rate at which certain types of instructions are being executed by CPU 102 and scaling the power and/or frequency of CPU 102 based on the rate. As shown in Figure 4A, monitor 112 analyzes the rate at which four specific types of instructions are being executed: integer arithmetic instructions 402 (the outermost loop), SIMD arithmetic instructions 414 (the second outermost loop), VFP

instructions 426 (the third outermost loop), and load/store instructions 438 (the innermost loop). Workloads exhibiting high instruction-per-cycle densities of these instruction types are often well-optimized and can benefit from increases in the power and/or frequency of CPU 102.

5 **[0060]** Beginning with the outermost loop, the integer arithmetic instructions 402 are compared at comparator 404 against an integer threshold T_i 406 (e.g., two hundred fifty instructions per cycle). If the integer threshold T_i 406 is exceeded by the rate at which integer arithmetic instructions 402 are being processed by CPU 102, then comparator 404 outputs an integer delta $e_i(t)$ 408 to integer control signal generator 410. The integer control signal generator 410 can be any form of a controller filter that is closed-loop stable. In one embodiment, the integer control signal generator 410 can be an integrator that, in turn, integrates integer deltas $e_i(t)$ 408 as they are output by comparator 404 and outputs an integer control signal $c_i(t)$ 412. Next, the integer control signal $c_i(t)$ 412 is directed to max-selector 449, which, as described in greater detail below, outputs a maximum of the integer control signal $c_i(t)$ 412, a SIMD control signal $c_n(t)$ 424 that is produced according to the second outermost loop of Figure 4A, a VFP control signal $c_v(t)$ 436 that is produced according to the third outermost loop of Figure 4A, or a load/store control signal $c_L(t)$ 448 that is produced according to the innermost loop of Figure 4A.

20 **[0061]** At the second outermost loop of Figure 4A, the SIMD arithmetic instructions 414 are compared at comparator 416 against a SIMD threshold T_n 418 (e.g., one hundred fifty instructions per cycle). If the SIMD threshold T_n 418 is exceeded by the rate at which SIMD arithmetic instructions 414 are being processed by CPU 102, then comparator 416 outputs a SIMD delta $e_n(t)$ 420 to SIMD control signal generator 422. The SIMD control signal generator 422 can be any form of a controller filter that is closed-loop stable. In one embodiment, the SIMD control signal generator 422 can be an integrator that, in turn, integrates SIMD deltas $e_n(t)$ 420 as they are output by comparator 416 and outputs the SIMD control signal $c_n(t)$ 424 to max-selector 449.

30 **[0062]** At the third outermost loop of Figure 4A, the VFP instructions 426 are compared at comparator 430 against a VFP threshold T_v 428 (e.g., fifty instructions per cycle). If the VFP threshold T_v 428 is exceeded by the rate at which VFP instructions 426 are being processed by CPU 102, then comparator 430 outputs a VFP delta $e_v(t)$ 432 to VFP control signal generator 434. The VFP control signal generator

434 can be any form of a controller filter that is closed-loop stable. In one embodiment, the VFP control signal generator 434 can be an integrator that, in turn, integrates VFP deltas $e_v(t)$ 432 as they are output by comparator 430 and outputs the VFP control signal $c_v(t)$ 436 to max-selector 449.

- 5 **[0063]** At the innermost loop of Figure 4A, the load/store instructions 438 are compared at comparator 442 against a load/store threshold T_L 439. If the load/store threshold T_L 439 is exceeded by the rate at which load/store instructions 438 are being processed by CPU 102, then comparator 442 outputs a load/store delta $e_L(t)$ 444 to load/store control signal generator 446. The load/store control signal generator 446
10 can be any form of a controller filter that is closed-loop stable. In one embodiment, the load/store control signal generator 446 can be an integrator that, in turn, integrates load/store deltas $e_L(t)$ 444 as they are output by comparator 442 and outputs the load/store control signal $c_L(t)$ 448 to max-selector 449.

- [0064]** As noted above, max-selector 449 is configured to output a maximum of
15 the integer control signal $c_i(t)$ 412, the SIMD control signal $c_n(t)$ 424, the VFP control signal $c_v(t)$ 436, and the load/store control signal $c_L(t)$ 448 as a power management control signal 440 to power manager 114. In turn, power manager 114 scales the power and/or frequency of CPU 102 according to power management control signal 440. Accordingly, monitor 112 enables the performance of CPU 102 to scale
20 dynamically when specific types of instructions that benefit from such a scaling are being executing by CPU 102. Notably, monitor 112 can be further configured to provide similar scaling in response to other types of instructions being executed by CPU 102, such as load/store instructions.

- [0065]** Figures 4B-4E illustrate a method 450 for analyzing the rate at which
25 certain types of instructions are being executed by CPU 102 and scaling the voltage and/or frequency of CPU 102 based on the rate, according to one embodiment of the invention. Although the method steps 450 are described in conjunction with Figures 1 and 4A, persons skilled in the art will understand that any system configured to perform the method steps, in any order, is within the scope of the invention.

- 30 **[0066]** As shown in Figure 4B, the method 450 begins at step 452, where monitor 112 samples a first rate at which SIMD instructions are being executed by a central processing unit (CPU). At step 456, monitor 112 outputs a SIMD delta value to a SIMD arithmetic integrator. At step 458, monitor 112 integrates, at the SIMD arithmetic integrator, the SIMD arithmetic delta value with previously-output SIMD

arithmetic delta values to produce a SIMD arithmetic-based power management control signal.

[0067] Turning now to Figure 4C, at step 460, monitor 112 samples a second rate at which vector floating point (VFP) instructions are being executed by CPU 102. At
5 step 464, monitor 112 outputs a VFP arithmetic delta value to a VFP arithmetic integrator. At step 466, monitor 112 integrates, at the VFP arithmetic integrator, the VFP arithmetic delta value with previously-output VFP delta values to produce a VFP-based power management control signal.

[0068] Turning now to Figure 4D, at step 468, monitor 112 samples a third rate at
10 which load/store instructions are being executed by CPU 102. At step 472, monitor 112 outputs a load/store delta value to a load/store integrator. At step 474, monitor 112 integrates, at the load/store integrator, the load/store delta value with previously-output load/store delta values to produce a load/store-based power management control signal.

[0069] Turning now to Figure 4E, at step 476, monitor 112 samples a fourth rate
15 at which integer arithmetic instructions are being executed by CPU 102. At step 480, monitor 112 outputs an integer arithmetic delta value to an integer arithmetic integrator. At step 482, monitor 112 integrates, at the integer arithmetic integrator, the integer arithmetic delta value with previously-output integer arithmetic delta
20 values to produce an integer arithmetic-based power management control signal.

[0070] At step 484, monitor 112 selects a largest of the SIMD arithmetic-based power management control signal, the VFP-based power management control signal, and the integer arithmetic-based power management control signal. At step 486,
25 monitor 112 scales a power level of the CPU based on the selected power management control signal.

[0071] As previously noted herein, embodiments of the invention also include a method for optimizing operations of the CPU 102 where the CPU 102 is configured to perform transactions with the memory controller 103 that manages access to a DRAM and a flash memory. According to one embodiment, memory controller 103 is
30 configured to separately measure the throughput of traffic to and from the CPU 102 and also separately measure the throughput of traffic to and from the flash memory subsystem. This technique provides increased CPU 102 performance for high-throughput data transaction workloads, e.g., video encoding and high-performance

photography. Once memory activity exceeds the relevant threshold(s), CPU performance is elevated.

[0072] Figure 5A illustrates a conceptual diagram of an embodiment directed to analyzing a rate at which the CPU 102 performs transactions with the memory controller 103, according to one embodiment of the invention. As shown in Figure 5A, monitor 112 analyzes the throughput of traffic to and from the CPU and also the throughput of traffic to and from the flash memory subsystem, which is represented in Figure 5A as: flash memory OUT 502 (the outermost loop), flash memory IN 514 (the second outermost loop), CPU OUT 526 (the third outermost loop), and CPU IN 538 (the innermost loop), respectively.

[0073] Beginning with the outermost loop, flash memory OUT 502 is compared at comparator 504 against a flash memory OUT threshold T_i 506. If flash memory OUT threshold T_i 506 is exceeded by the rate of flash memory OUT 502, then comparator 504 outputs a flash memory OUT delta $e_i(t)$ 508 to flash memory OUT control signal generator 510. The flash memory OUT control signal generator 510 can be any form of a controller filter that is closed-loop stable. In one embodiment, the flash memory OUT control signal generator 510 can be an integrator that, in turn, integrates flash memory OUT deltas $e_i(t)$ 508 as they are output by comparator 504 and outputs a flash memory OUT control signal $c_i(t)$ 512. Next, the flash memory OUT control signal $c_i(t)$ 512 is directed to max-selector 549, which, as described in greater detail below, outputs a maximum of the flash memory OUT control signal $c_i(t)$ 512, a flash memory IN control signal $c_n(t)$ 524 that is produced according to the second outermost loop of Figure 5A, a CPU OUT control signal $c_v(t)$ 536 that is produced according to the third outermost loop of Figure 5A, or a CPU IN control signal $c_L(t)$ 548 that is produced according to the innermost loop of Figure 5A.

[0074] At the second outermost loop of Figure 5A, flash memory IN 514 is compared at comparator 516 against a flash memory IN threshold T_n 518. If flash memory IN threshold T_n 518 is exceeded by the rate of flash memory IN 514, then comparator 516 outputs a flash memory IN delta $e_n(t)$ 520 to flash memory IN control signal generator 522. The flash memory IN control signal generator 522 can be any form of a controller filter that is closed-loop stable. In one embodiment, the flash memory IN control signal generator 522 can be an integrator that, in turn, integrates flash memory IN deltas $e_n(t)$ 520 as they are output by comparator 516 and outputs the flash memory IN control signal $c_n(t)$ 524 to max-selector 549.

[0075] At the third outermost loop of Figure 5A, CPU OUT 526 is compared at comparator 530 against a CPU OUT threshold T_v 528. If CPU OUT threshold T_v 528 is exceeded by the rate of CPU OUT 526, then comparator 530 outputs a CPU OUT delta $e_v(t)$ 532 to CPU OUT control signal generator 534. The CPU OUT control signal generator 534 can be any form of a controller filter that is closed-loop stable. In one embodiment, the CPU OUT control signal generator 534 can be an integrator that, in turn, integrates CPU OUT deltas $e_v(t)$ 532 as they are output by comparator 530 and outputs the CPU OUT control signal $c_v(t)$ 536 to max-selector 549.

[0076] At the innermost loop of Figure 5A, CPU IN 538 is compared at comparator 542 against a CPU IN threshold T_L 539. If CPU IN threshold T_L 539 is exceeded by the rate of CPU IN 538, then comparator 542 outputs a CPU IN delta $e_L(t)$ 544 to CPU IN control signal generator 546. The CPU IN control signal generator 546 can be any form of a controller filter that is closed-loop stable. In one embodiment, the CPU IN control signal generator 546 can be an integrator that, in turn, integrates CPU IN deltas $e_L(t)$ 544 as they are output by comparator 542 and outputs the CPU IN control signal $c_L(t)$ 548 to max-selector 549.

[0077] As noted above, max-selector 549 is configured to output a maximum of the flash memory OUT control signal $c_i(t)$ 512, the flash memory IN control signal $c_n(t)$ 524, the CPU OUT control signal $c_v(t)$ 536, and the CPU IN control signal $c_L(t)$ 548 as a power management control signal 540 to power manager 114. In turn, power manager 114 scales the power and/or frequency of CPU 102 according to power management control signal 540. Accordingly, monitor 112 enables the performance of CPU 102 to scale dynamically when executing high-throughput data transaction workloads, e.g., video encoding and high-performance photography.

[0078] Figure 5B illustrates a method 550 for optimizing operations of CPU 102 when CPU 102 is configured to perform transactions with memory controller 103 and memory controller 103 is configured to manage access to a DRAM and a flash memory, according to one embodiment of the invention. As shown, the method 550 begins at step 552, where monitor 112 samples first, second, third, and fourth rates at which traffic is being throughput through a memory controller, where the first rate and the second rate correspond to traffic throughput to/from the CPU 102, respectively, and the third rate and the fourth rate correspond to traffic throughput to/from a flash memory subsystem, respectively.

[0079] At step 554, monitor 112 outputs, for each of the first, second, third, and fourth rates, a throughput delta value to a first, second, third, and fourth throughput integrator, respectively. At step 556, monitor 112, at each of the first, second, third, and fourth throughput integrators, integrate the first, second, third, and fourth throughput delta values, respectively, with previously-output first, second, third, and fourth throughput delta values, respectively, to produce first, second, third, and fourth throughput-based power management control signals, respectively.

[0080] At step 558, monitor 112 selects a largest of the first, second, third, and fourth throughput-based power management control signals. At step 560, monitor 112 scales a power level of the CPU 102 on the selected power management control signal.

[0081] Figure 6A illustrates a conceptual diagram 600 of an embodiment directed to scaling the voltage and/or frequency of the CPU 102 when the CPU 102 executes workloads that have characteristics of traditional desktop/laptop computer applications, according to one embodiment of the invention. In particular, according to this embodiment, the voltage and/or frequency of the CPU 102 is only scaled when the mobile computing device 100 does not appear to be executing a UI-oriented workload. As illustrated in Figure 6A, the technique involves two control loops, where the first control loop is configured to monitor CPU 102 for a CPU complex (or “package”) utilization measurement 601, and where the second control loop is configured to monitor CPU 102 for a core utilization measurement 613.

[0082] According to the embodiment illustrated in Figure 6A, the first control loop involves measuring a fraction of the sample interval in which at least one core of CPU 102 is active. The complex utilization measurement 601 is compared at comparator 602 against a complex utilization target 603 (e.g., 99%), and a delta $e_U(t)$ 604 is output by comparator 602 to an integrator 605. Next, the output of integrator 605 is fed into a max-selector 628, which outputs a maximum of the output of the integrator 605 and an integrator 624 of the second loop (described in greater detail below).

[0083] The second control loop involves measuring the duty factor of the cores of the CPU 102 by adding up the time each core spends active. For example, a dual-core CPU 102 would report a utilization of 100% if both of the cores were active throughout an entire sample interval. As shown in Figure 6A, the core utilization measurement 613 is compared at comparator 614 against a core utilization target 615

(e.g., 90%), and a delta $e_u(t)$ 616 is output by comparator 614 to an integrator 624. Next, the output of integrator 624 is fed into the max-selector 628, which, as noted above, outputs a maximum of the output of the integrator 605 and the integrator 624. Finally, component 630 takes into account whether or not a threshold NFPS are being
5 input into the frame buffer 108. In particular, if a threshold NFPS (e.g., 15 FPS) are being input into the frame buffer 108, then the output of the max-selector 628 is not fed into the power manager 114; otherwise, the output of the max-selector 628 is fed into the power manager 114, and the voltage and/or frequency of the CPU 102 is scaled according to the output of the max-selector 628.

10 **[0084]** Figure 6B illustrates a method 650 for scaling the voltage and/or frequency of a CPU when the CPU executes workloads that have characteristics of traditional desktop/laptop computer applications, according to one embodiment of the invention.

[0085] As shown, the method 650 begins at step 652, where monitor 112
15 generates a first control signal based at least in part on measuring a sample interval in which at least one core of a central processing unit (CPU) is active. At step 654, monitor 112 generates a second control signal based at least in part on measuring an amount of time that each core of the CPU is active. At step 656, monitor 112 selects a maximum of the first control signal and the second control signal. At step 658,
20 monitor 112 determines if a user interface activity level exceeds a threshold (e.g., by monitoring a NFPS being input into the frame buffer 108). If, at step 658, monitor 112 determines that the user interface activity level exceeds the threshold, then method 650 ends; otherwise, at step 660, monitor 112 scales a voltage and/or frequency of the CPU based on the control signal selected at step 656.

25 **[0086]** Although the foregoing invention has been described in detail by way of illustration and example for purposes of clarity and understanding, it will be recognized that the above described invention may be embodied in numerous other specific variations and embodiments without departing from the spirit or essential characteristics of the invention. Certain changes and modifications may be practiced,
30 and it is understood that the invention is not to be limited by the foregoing details, but rather is to be defined by the scope of the appended claims.

CLAIMS

What is claimed is:

1. A method for updating an operating mode of a central processing unit (CPU),
5 comprising:
monitoring a cycle-to-cycle jitter associated with a rate by which a user
interface (UI) is refreshed; and
adjusting an operating mode of the CPU based on the cycle-to-cycle jitter.
2. The method of claim 1, wherein adjusting the operating mode of the CPU
10 comprises adjusting the voltage and/or frequency at which the CPU is operating.
3. The method of claim 1, wherein monitoring the cycle-to-cycle jitter comprises
analyzing a rate of change in a number of frames per second (NFPS) being input to a
frame buffer associated with the CPU.
4. The method of claim 1, wherein monitoring the cycle-to-cycle jitter comprises
15 establishing:
a jitter control signal based on short-term sampling of the NFPS being input to
the frame buffer; and
a trend control signal based on long-term sampling of the NFPS being input to
the frame buffer.
- 20 5. The method of claim 4, wherein the operating mode of the CPU is adjusted via
a control signal that is selected from a maximum of the jitter control signal and the
trend control signal.
6. The method of claim 4, further comprising applying a first gain value to the
jitter control signal and applying a second gain value to the trend control signal.
- 25 7. The method of claim 1, wherein the CPU is a central processing unit (CPU)
included in a mobile computing device.
8. In a mobile computing device having a central processing unit (CPU)
configured to issue instructions to a graphical processing unit (GPU), a method for
optimizing operations of the CPU, comprising:
30 determining that a utilization rate of the GPU is exceeding a threshold level;
determining that the CPU is operating in a sub-optimal operating mode; and
causing the CPU to enter into an optimal operating mode where the CPU
generates instructions for execution by the GPU at a faster rate.

9. The method of claim 8, wherein causing the CPU to enter into the optimal operating mode comprises:
establishing a control signal by a control signal generator.
10. The method of claim 8, wherein causing the CPU to enter into the optimal operating mode comprises adjusting a power and/or frequency at which the CPU is operating.
11. The method of claim 8, further comprising:
determining that the utilization rate of the GPU is no longer exceeding the threshold level; and
causing the CPU to enter back into the sub-optimal operating mode.
12. The method of claim 8, wherein the threshold level is a dynamic value that is adjusted to influence when the CPU enters into the optimal operating mode or enters into the sub-optimal operating mode.
13. A method for updating an operating mode of a central processing unit (CPU), comprising:
determining that the CPU is tasked with executing instructions that are associated with a high instruction-per-cycle density; and
causing the CPU to enter into a high-performance operating mode to cause an increase in the rate at which the CPU executes the instructions.
14. The method of claim 13, wherein the instructions comprise integer arithmetic instructions, single instruction multiple data (SIMD) instructions, vector floating point (VFP) instructions, and load/store instructions.
15. The method of claim 14, wherein the step of determining comprises establishing:
an integer arithmetic control signal based on the rate at which integer arithmetic instructions are being executed by the CPU;
a SIMD control signal based on the rate at which SIMD instructions are being executed by the CPU;
a VFP control signal based on the rate at which VFP instructions are being executed by the CPU; and
a load/store control signal based on the rate at which load/store instructions are being executed by the CPU.

16. The method of claim 15, wherein the operating mode of the CPU is adjusted via a control signal that is selected from a maximum of the integer arithmetic control signal, the SIMD control signal, the VFP control signal, and the load/store control signal.
- 5 17. The method of claim 15, further comprising applying a first gain value to the integer arithmetic control signal, applying a second gain value to the SIMD control signal, applying a third gain value to the VFP control signal, and applying a fourth gain value to the load/store control signal.\
18. A system, comprising:
- 10 a central processing unit (CPU);
a frame buffer; and
a memory storing instructions that, when executed by the CPU, cause the CPU to implement a method, wherein the method comprises:
monitoring a cycle-to-cycle jitter associated with a rate by which a
15 user interface (UI) is refreshed; an
adjusting an operating mode of the CPU based on the cycle-to-cycle jitter.
19. The system of claim 18, wherein adjusting the operating mode of the CPU comprises adjusting the power and/or frequency at which the CPU is operating.
- 20 20. The system of claim 18, wherein monitoring the cycle-to-cycle jitter comprises analyzing a rate of change in a number of frames per second (NFPS) being input to the frame buffer.
21. The system of claim 18, wherein monitoring the cycle-to-cycle jitter comprises establishing:
- 25 a jitter control signal based on short-term sampling of the NFPS being input to the frame buffer; and
a trend control signal based on long-term sampling of the NFPS being input to the frame buffer.
22. The system of claim 21, wherein the jitter control signal and the trend control
30 signal are used to produce a final control signal whose value maps to a particular operating mode of the CPU.
23. The system of claim 22, wherein the value of the final control signal defines a number of cores to be operating within the CPU.

24. A method for optimizing operations of a central processing unit (CPU) that is configured to perform transactions with a memory controller, the method comprising:
monitoring data throughput within the memory controller, wherein the
memory controller is configured to interface with a dynamic random
access memory (DRAM) and a flash memory; and
adjusting an operating mode of the CPU based on the monitored data
throughput.
25. The method of claim 24, wherein monitoring the data throughput includes:
monitoring data throughput to and from the CPU; and
monitoring data throughput to and from the flash memory.
26. The method of claim 25, further comprising:
establishing first and second control signals based on the monitored data
throughput to and from the CPU, respectively; and
establishing third and fourth control signals based on the monitored data
throughput to and from the flash memory, respectively.
27. The method of claim 26, further comprising a final control signal based on a maximum of the first, second, third, and fourth control signals, wherein the operating mode of the CPU is based on the final control signal.
28. A method for updating an operating mode of a central processing unit (CPU) when the CPU is executing workloads that are characteristic of traditional desktop/laptop computer applications, the method comprising:
generating a first control signal based at least in part on measuring a sample interval in which at least one core of the CPU is active;
generating a second control signal based at least in part on measuring an amount of time that each core of the CPU is active; and
adjusting the operating mode of the CPU based on a maximum of the first control signal and the second control signal when less than a threshold number of frames per second are being input into a frame buffer.
29. The method of claim 28, wherein adjusting the operating mode of the CPU comprises adjusting the voltage and/or frequency at which the CPU is operating.
30. The method of claim 28, further comprising applying a first gain value to the first control signal and applying a second gain value to the second control signal.

31. The method of claim 28, wherein the CPU is a central processing unit (CPU) included in a mobile computing device.

1/18

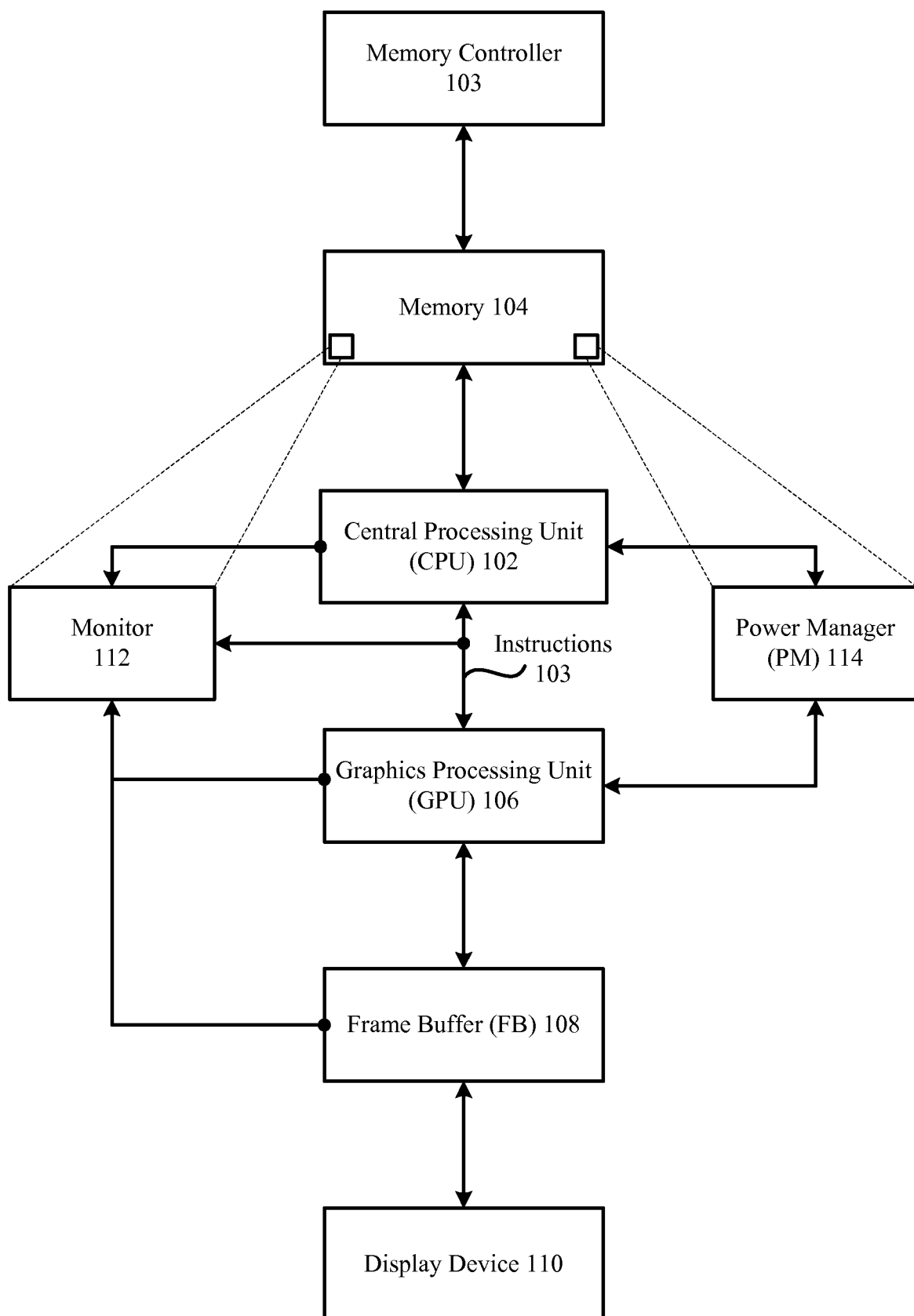

100 

FIG. 1

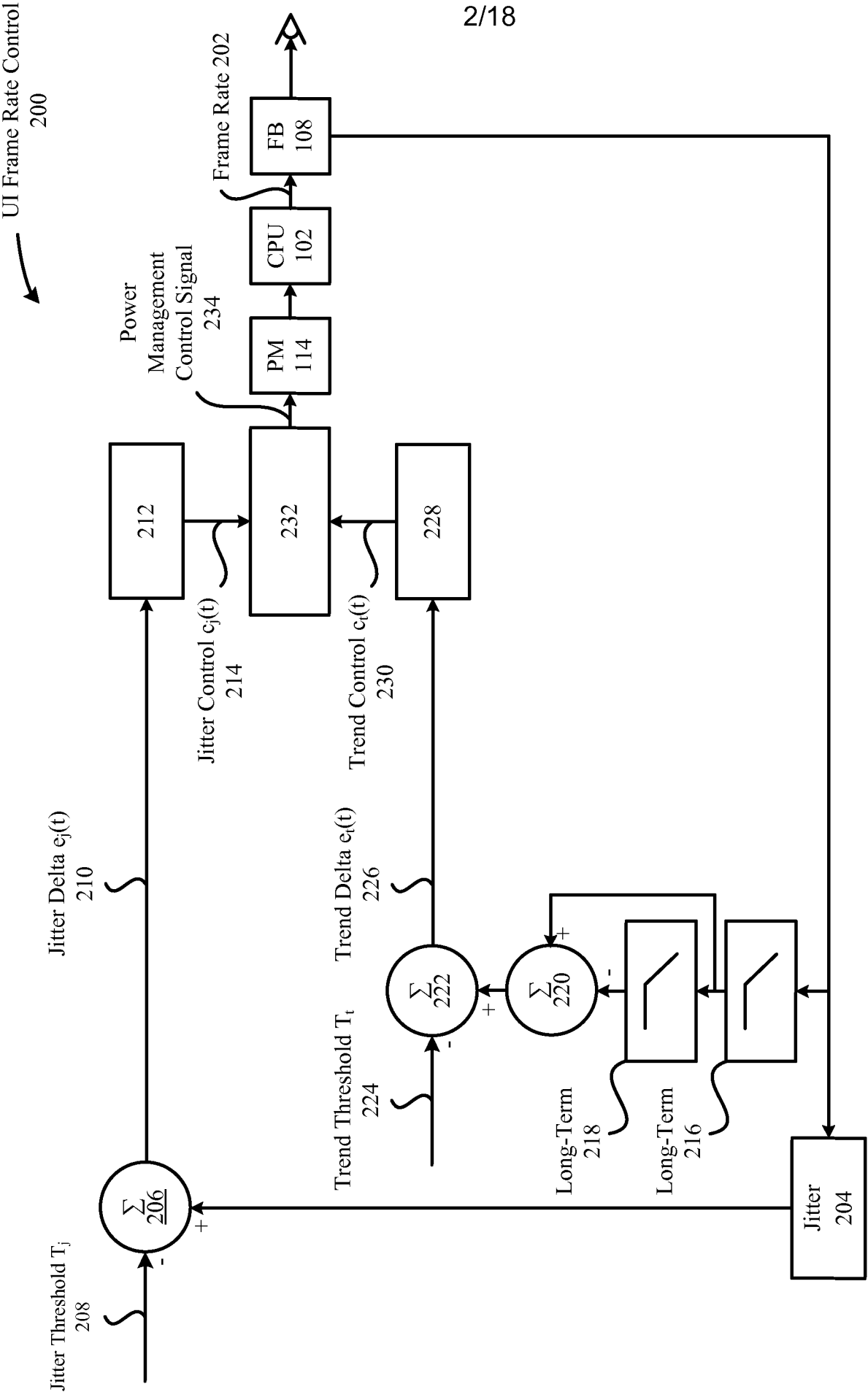


FIG. 2A

3/18

270

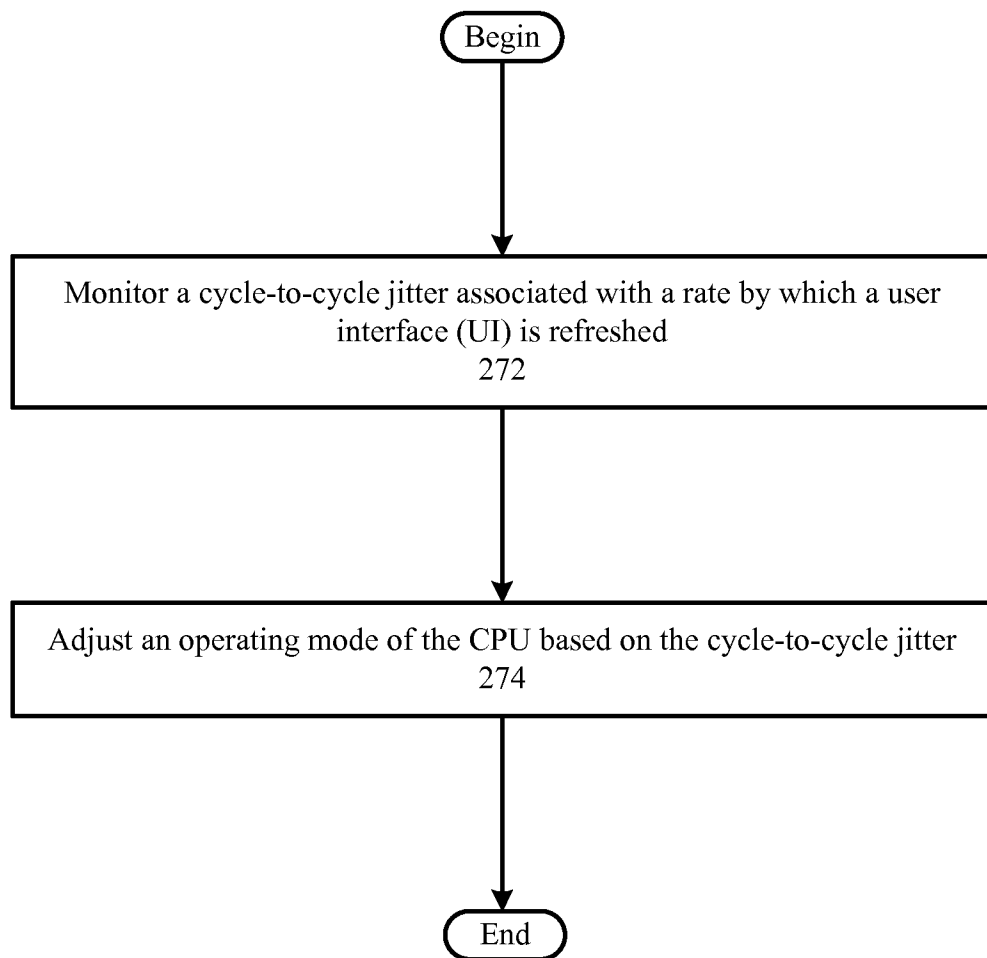


FIG. 2B

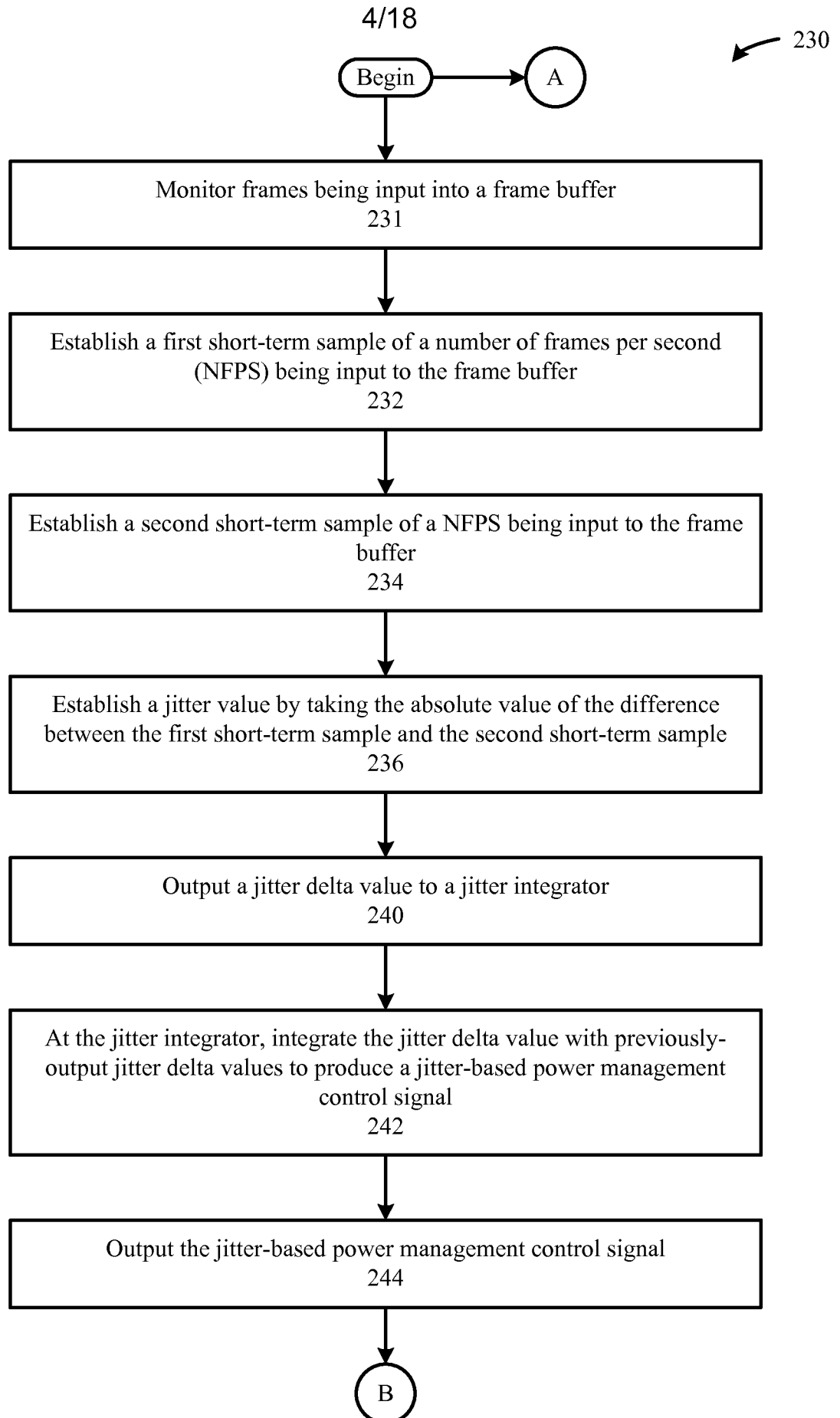


FIG. 2C

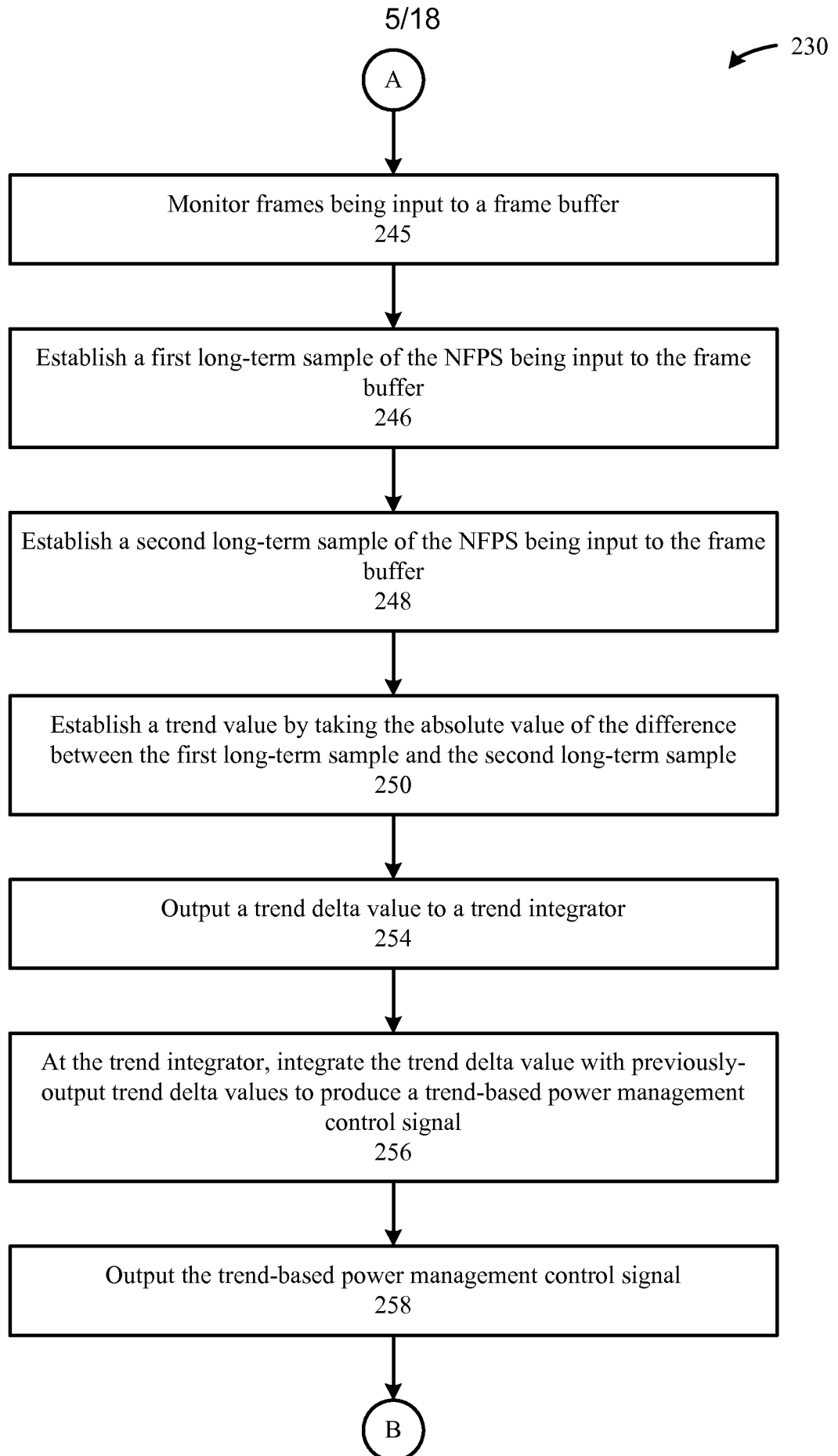


FIG. 2D

6/18

230

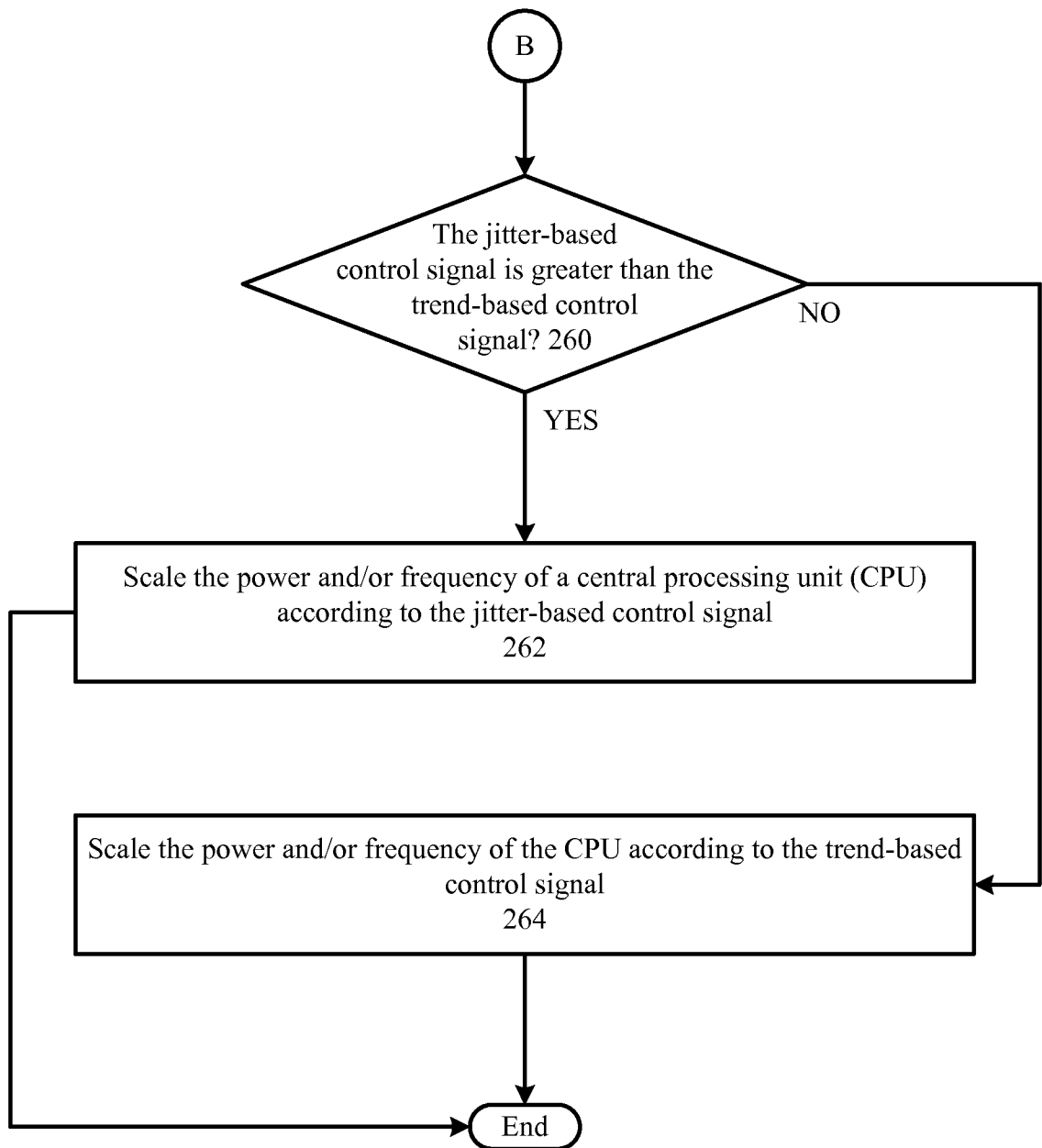


FIG. 2E

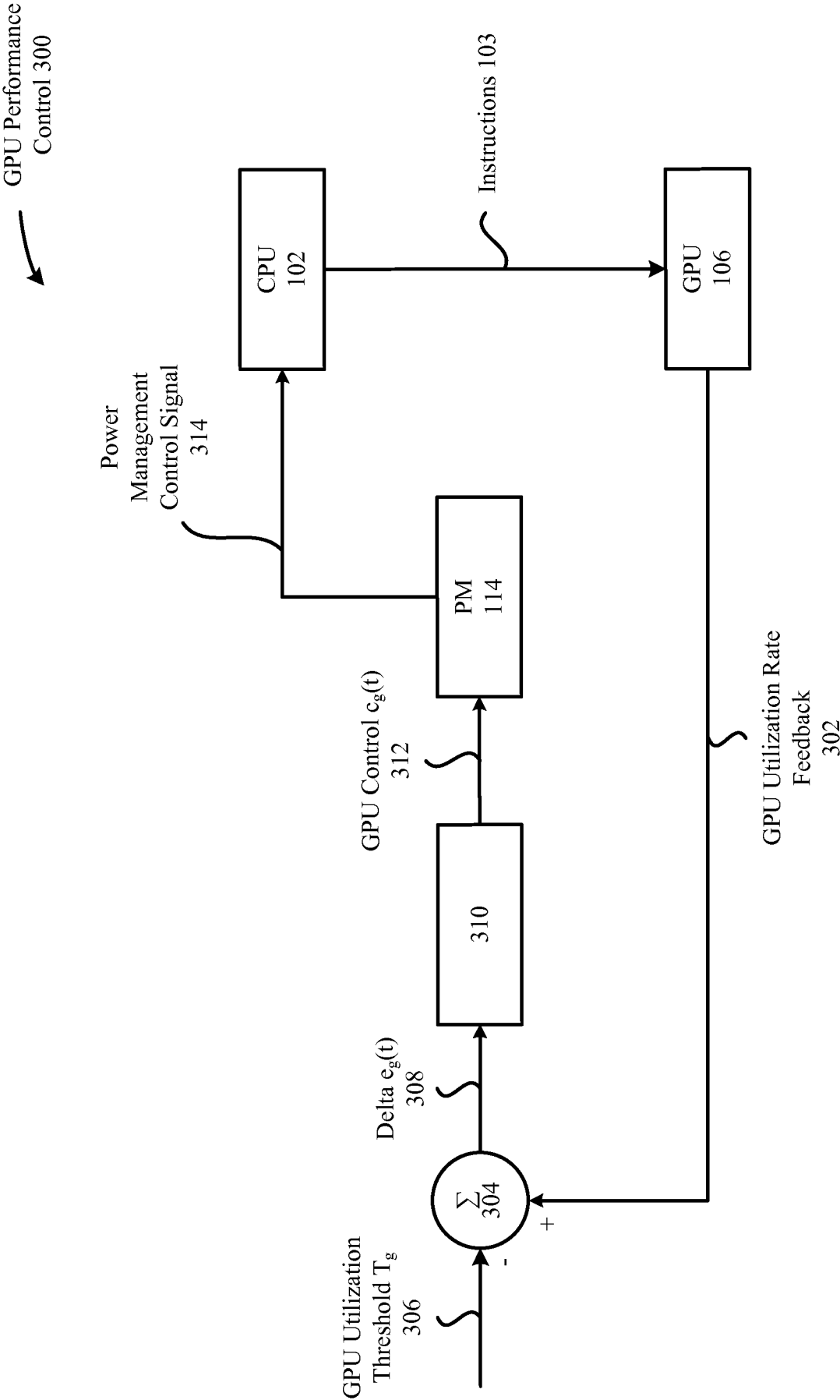


FIG. 3A

8/18

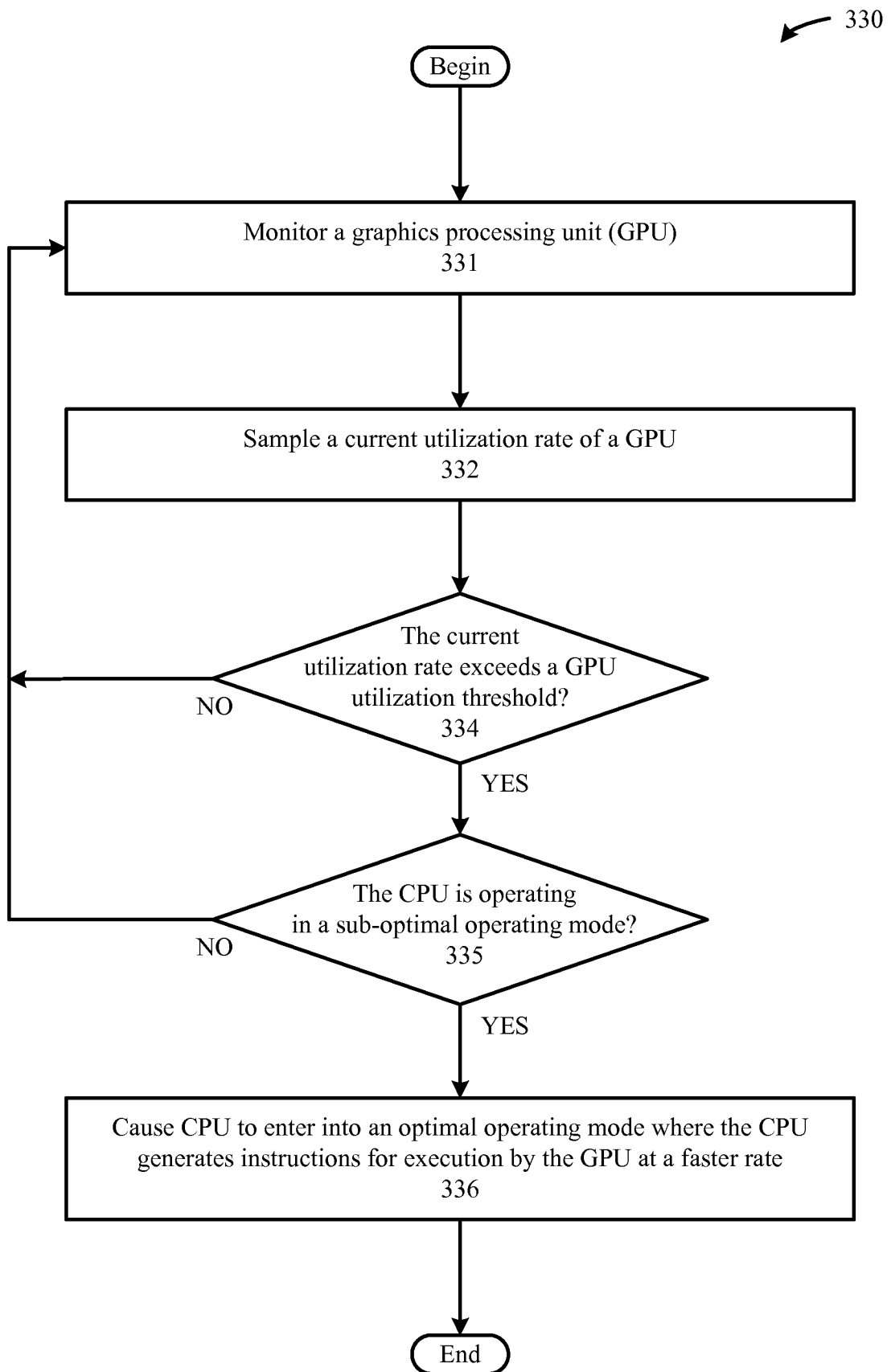


FIG. 3B

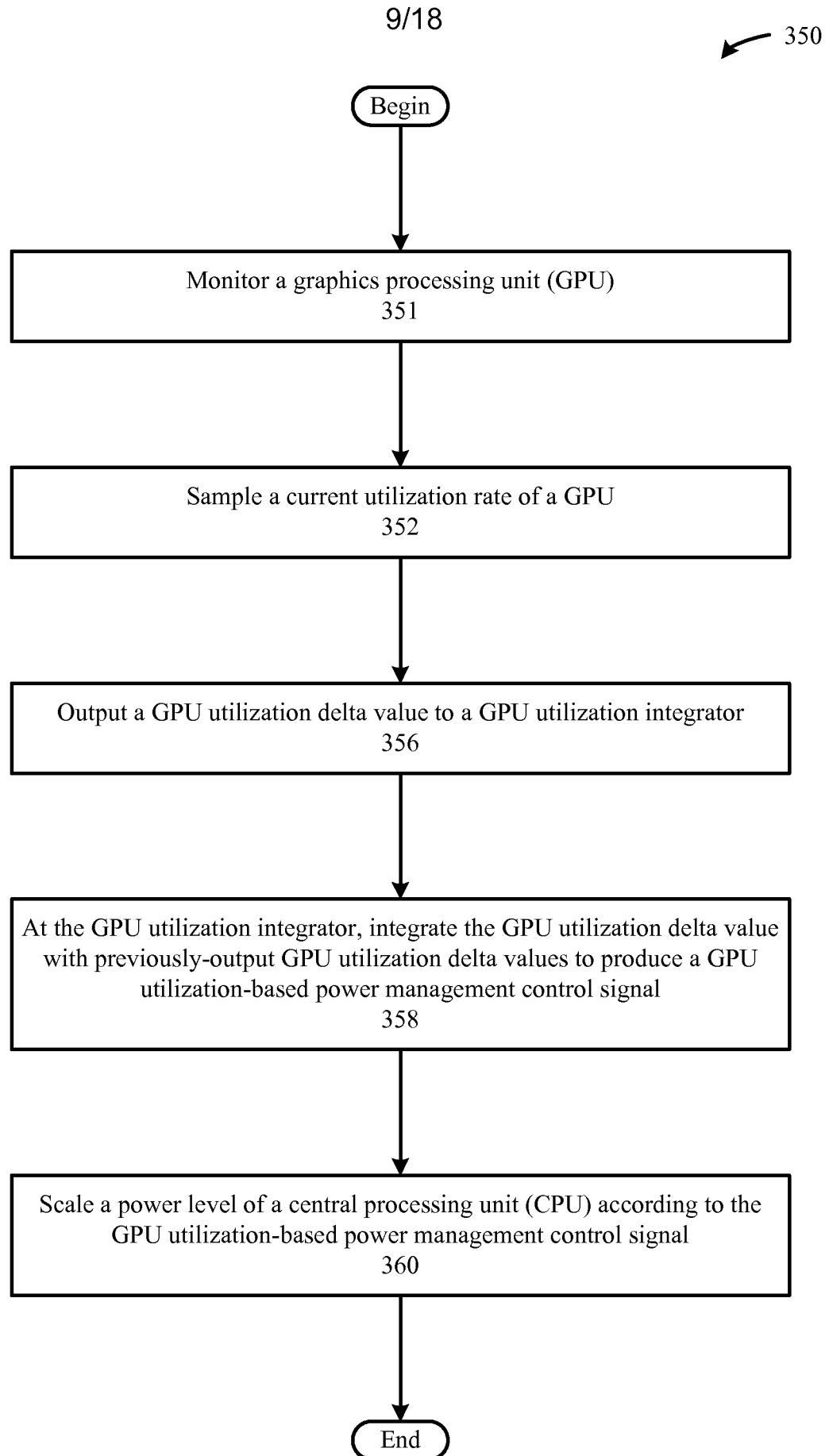
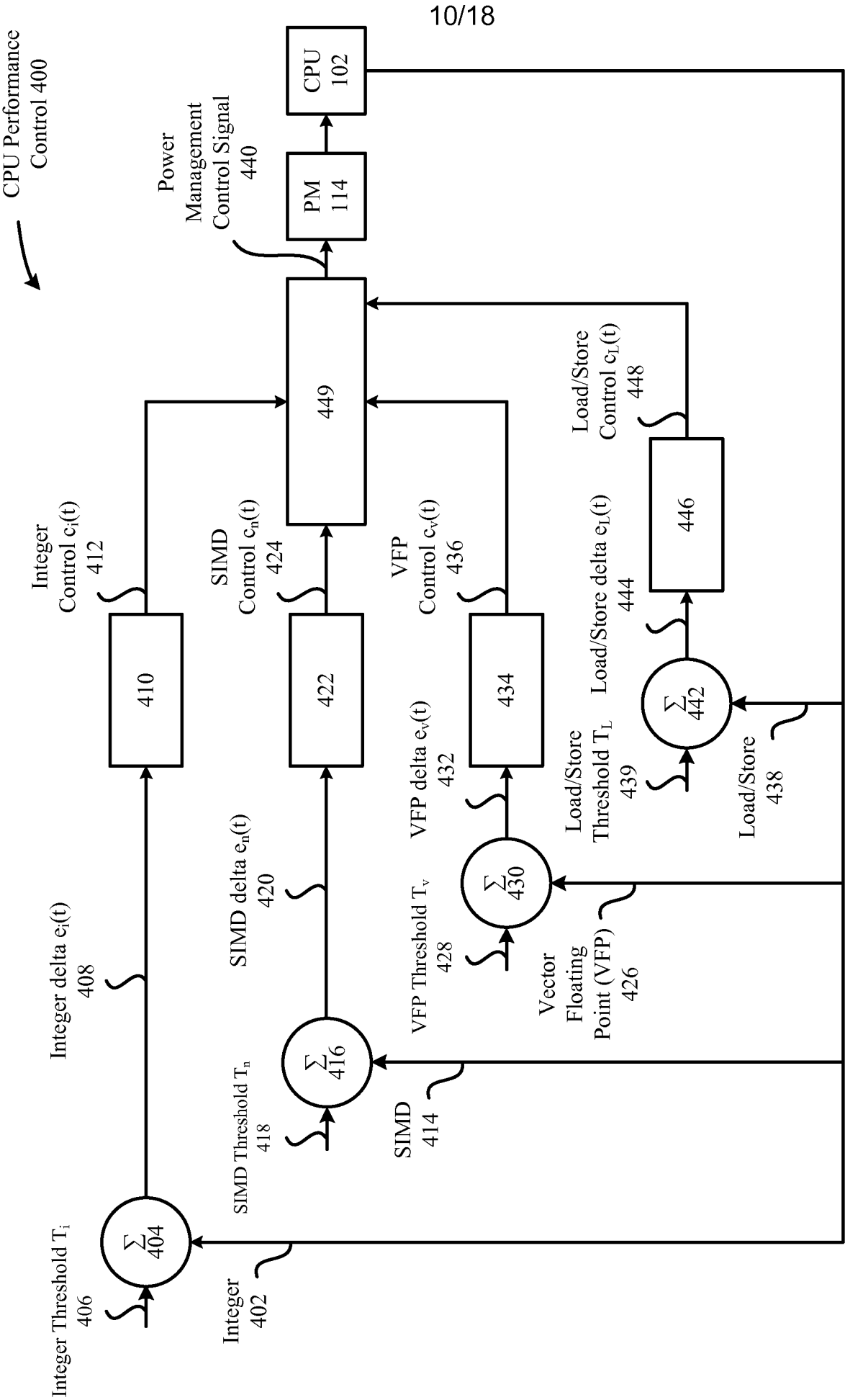


FIG. 3C



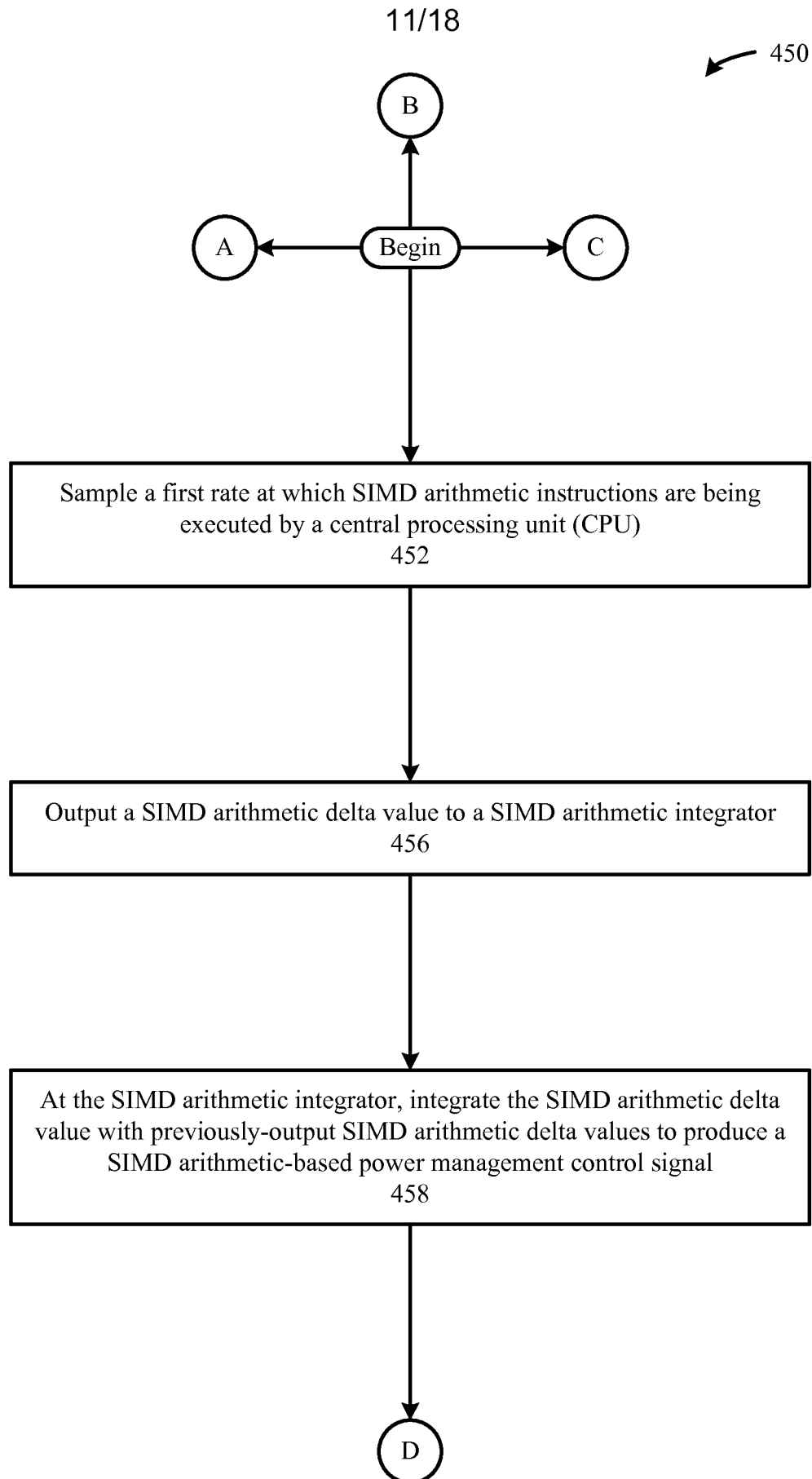


FIG. 4B

12/18

450

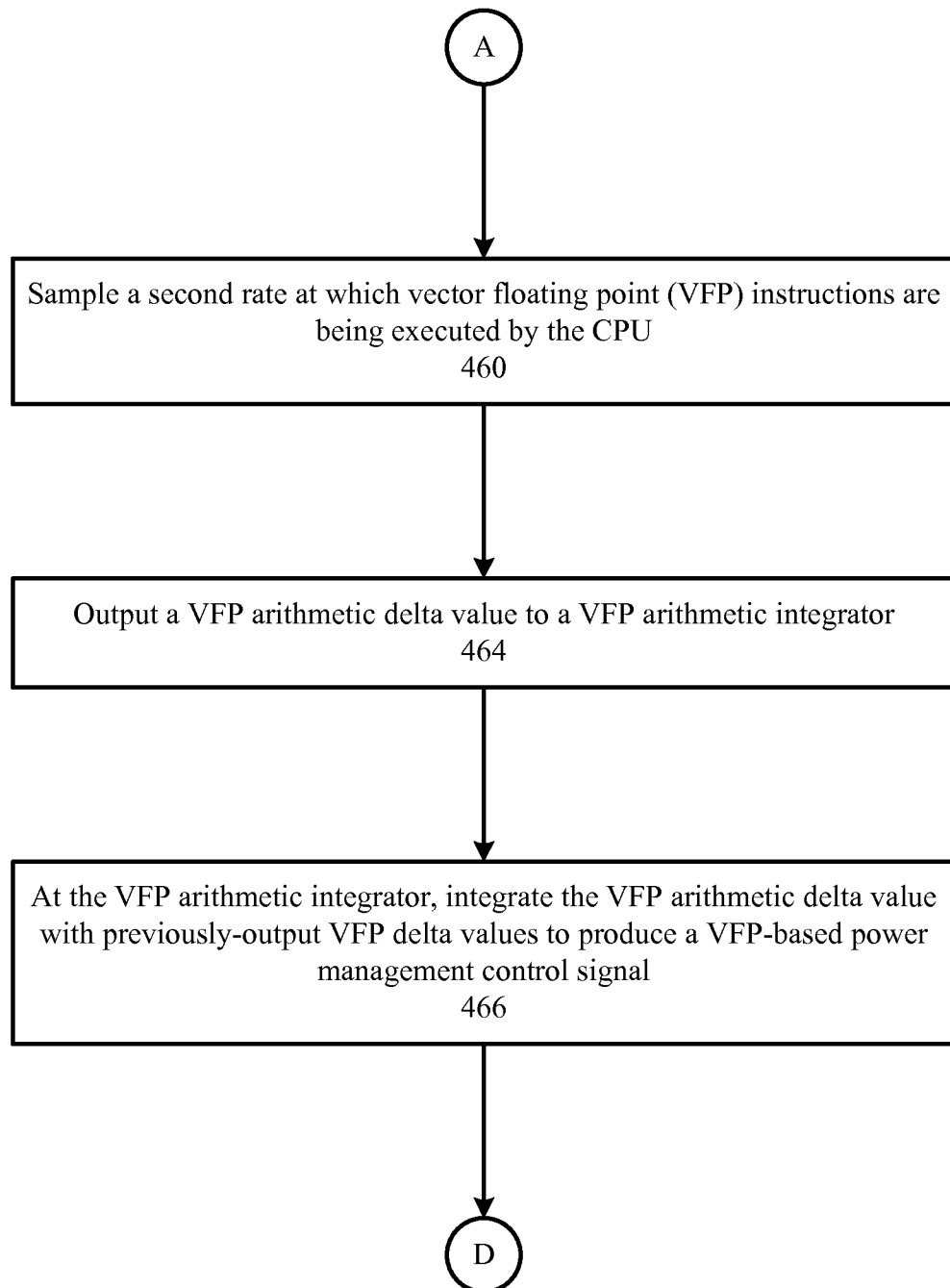


FIG. 4C

13/18

450

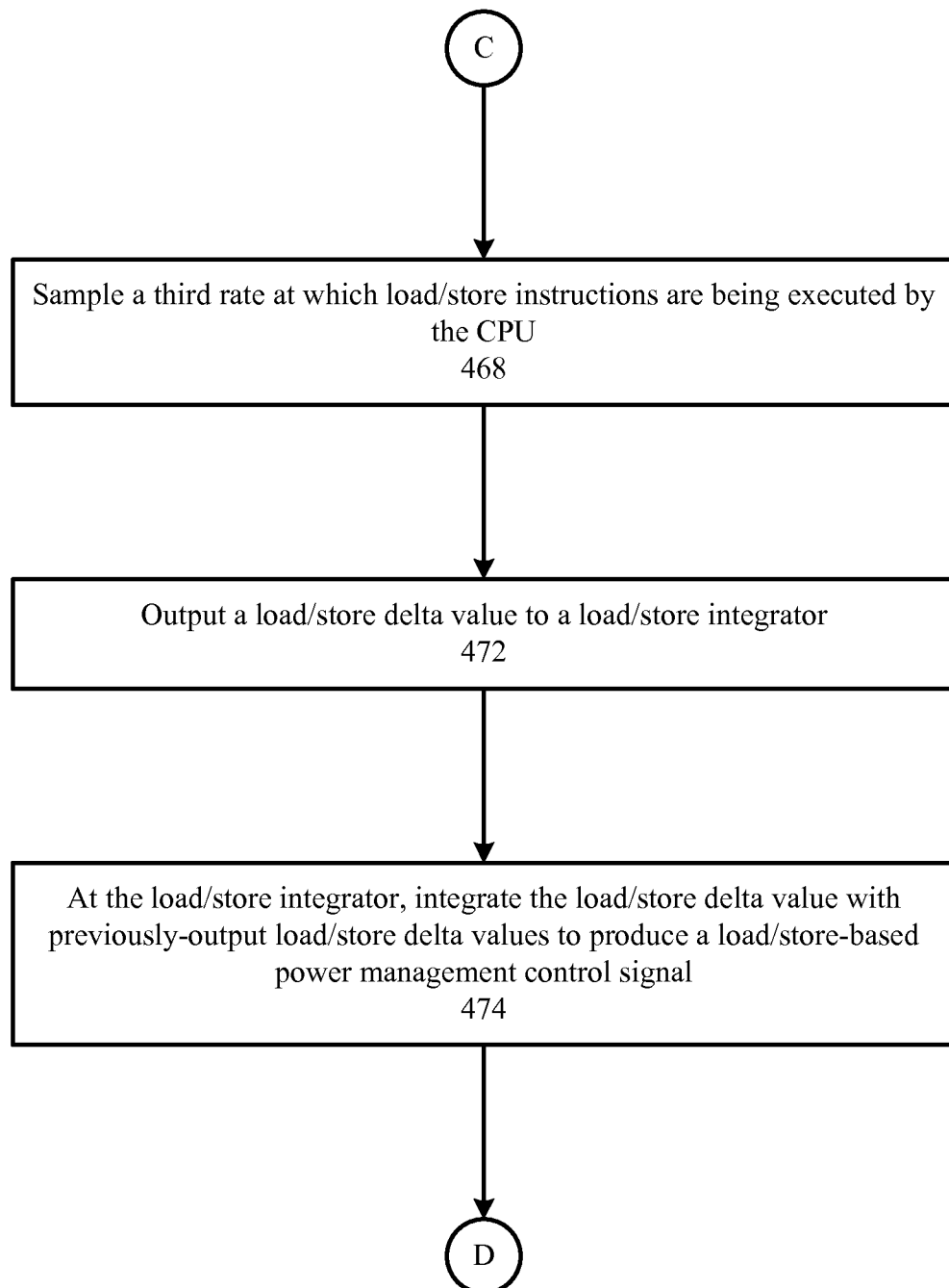


FIG. 4D

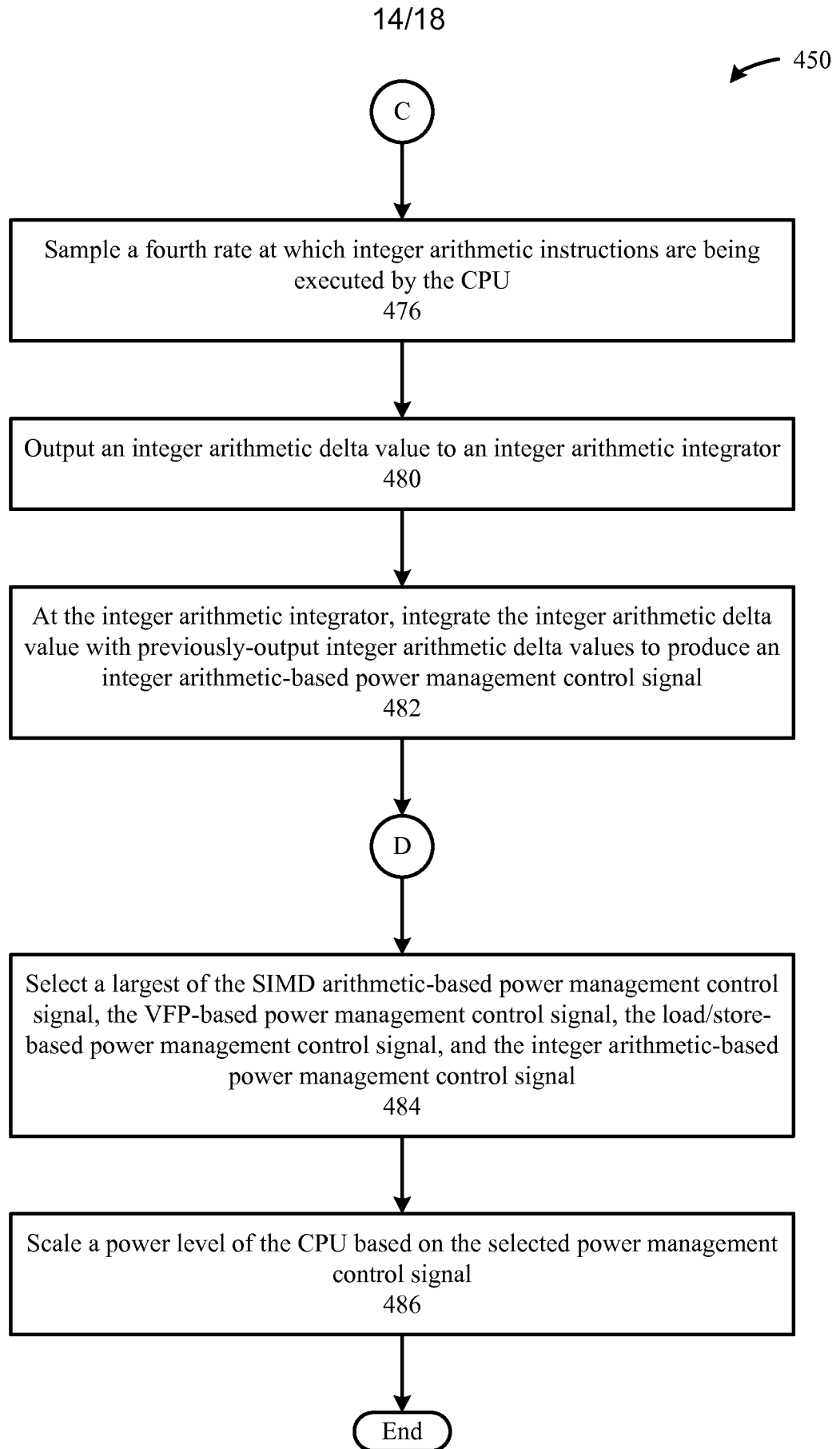


FIG. 4E

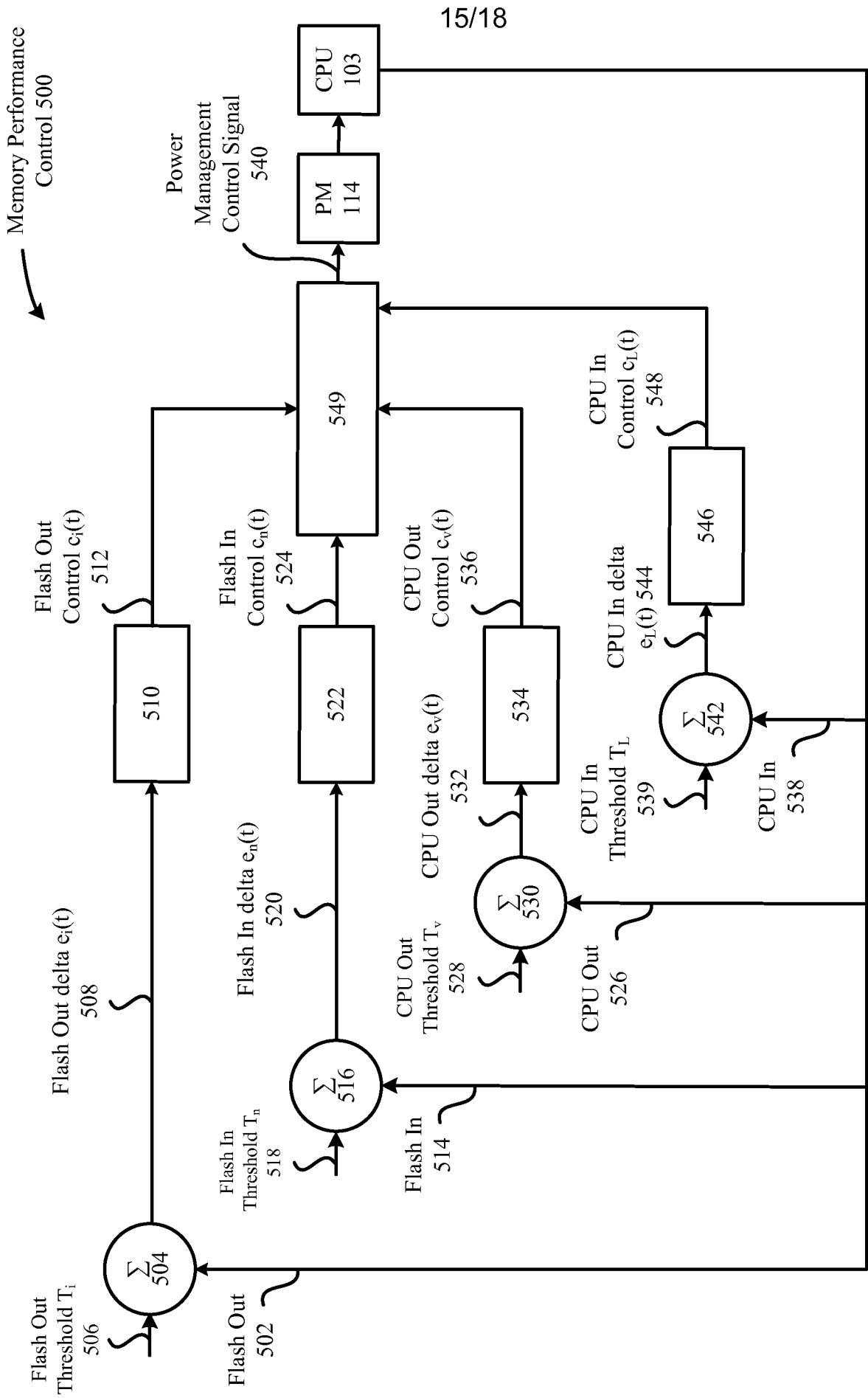


FIG. 5A

16/18

550

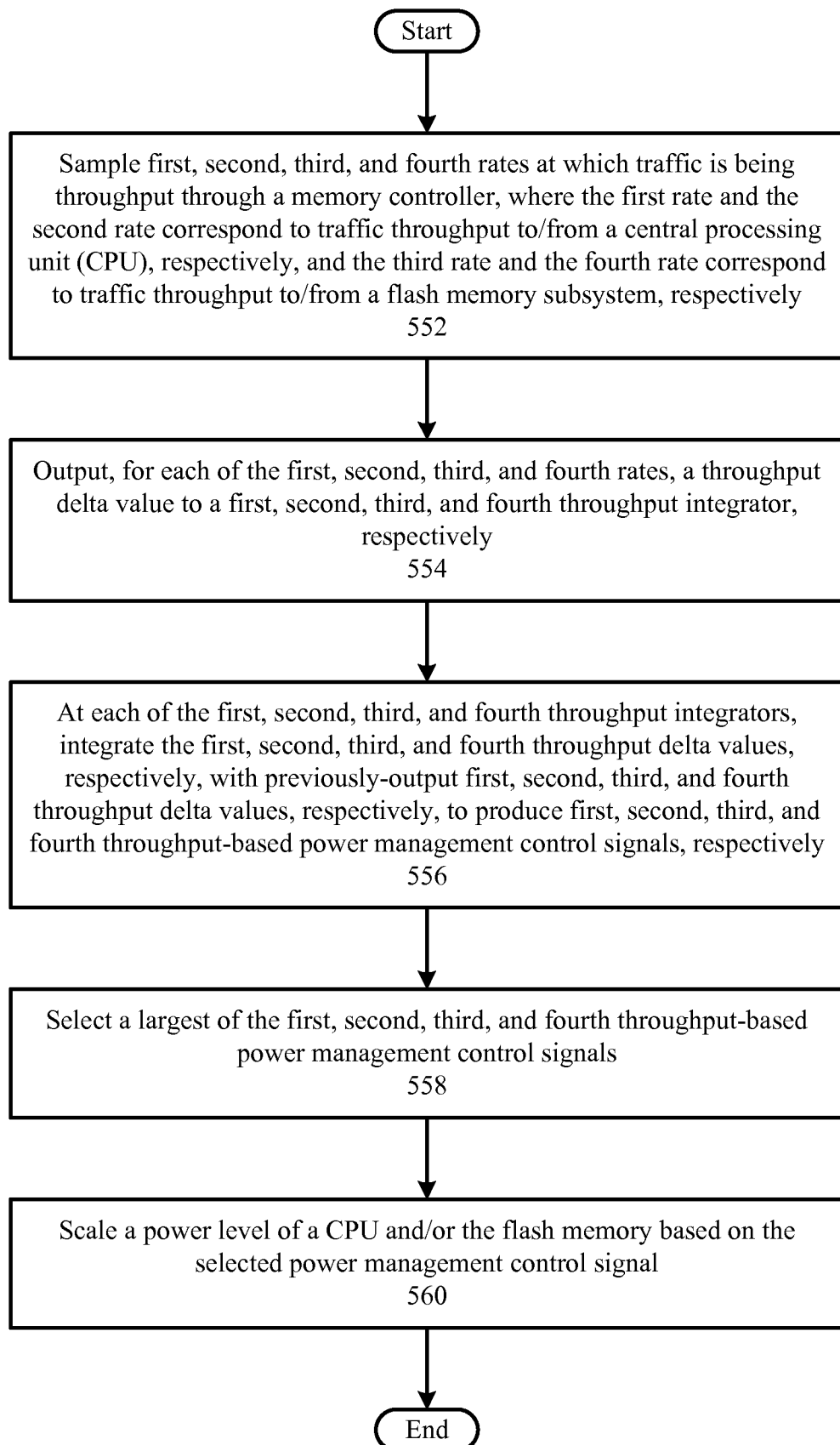


FIG. 5B

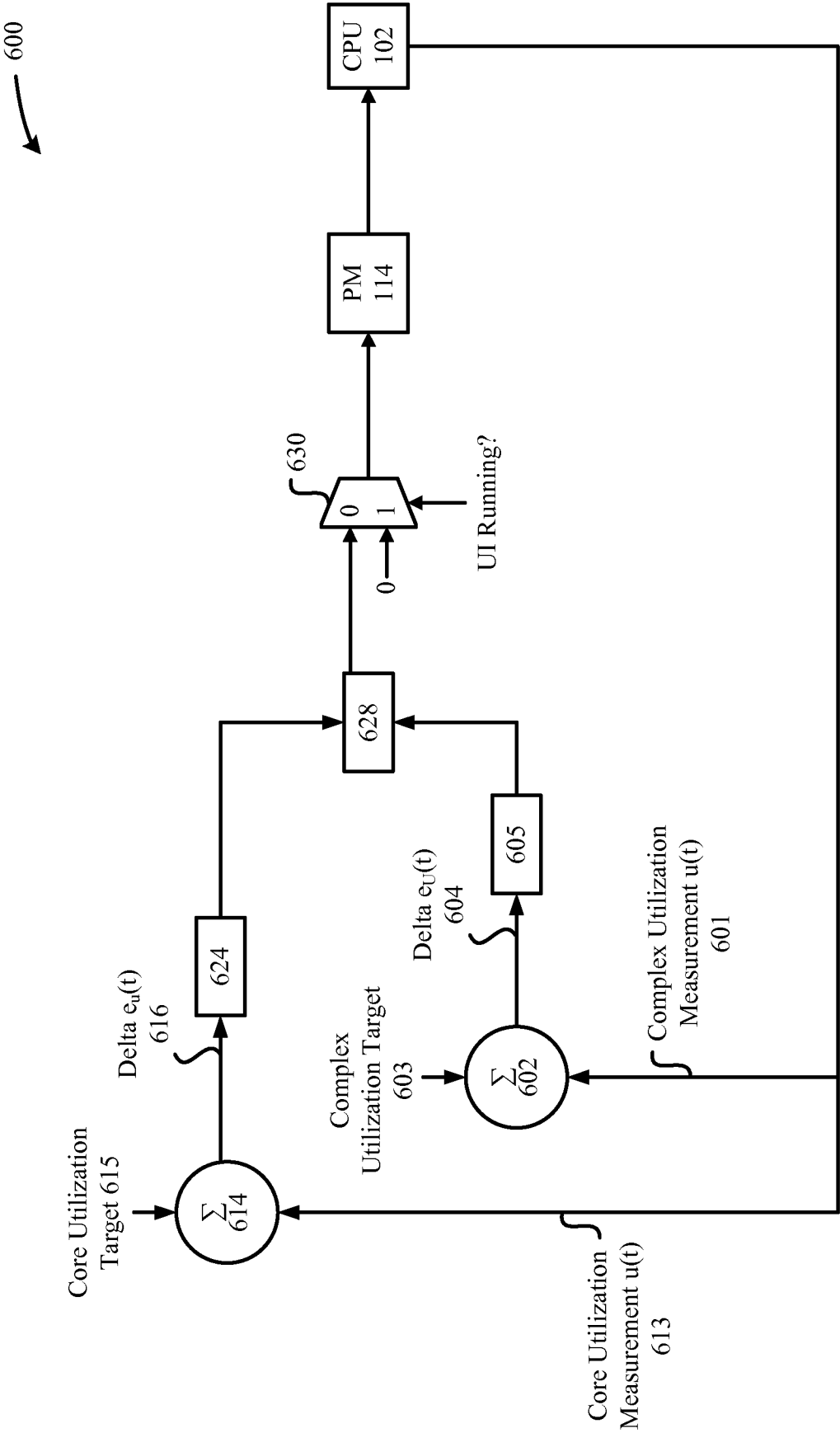


FIG. 6A

18/18

650

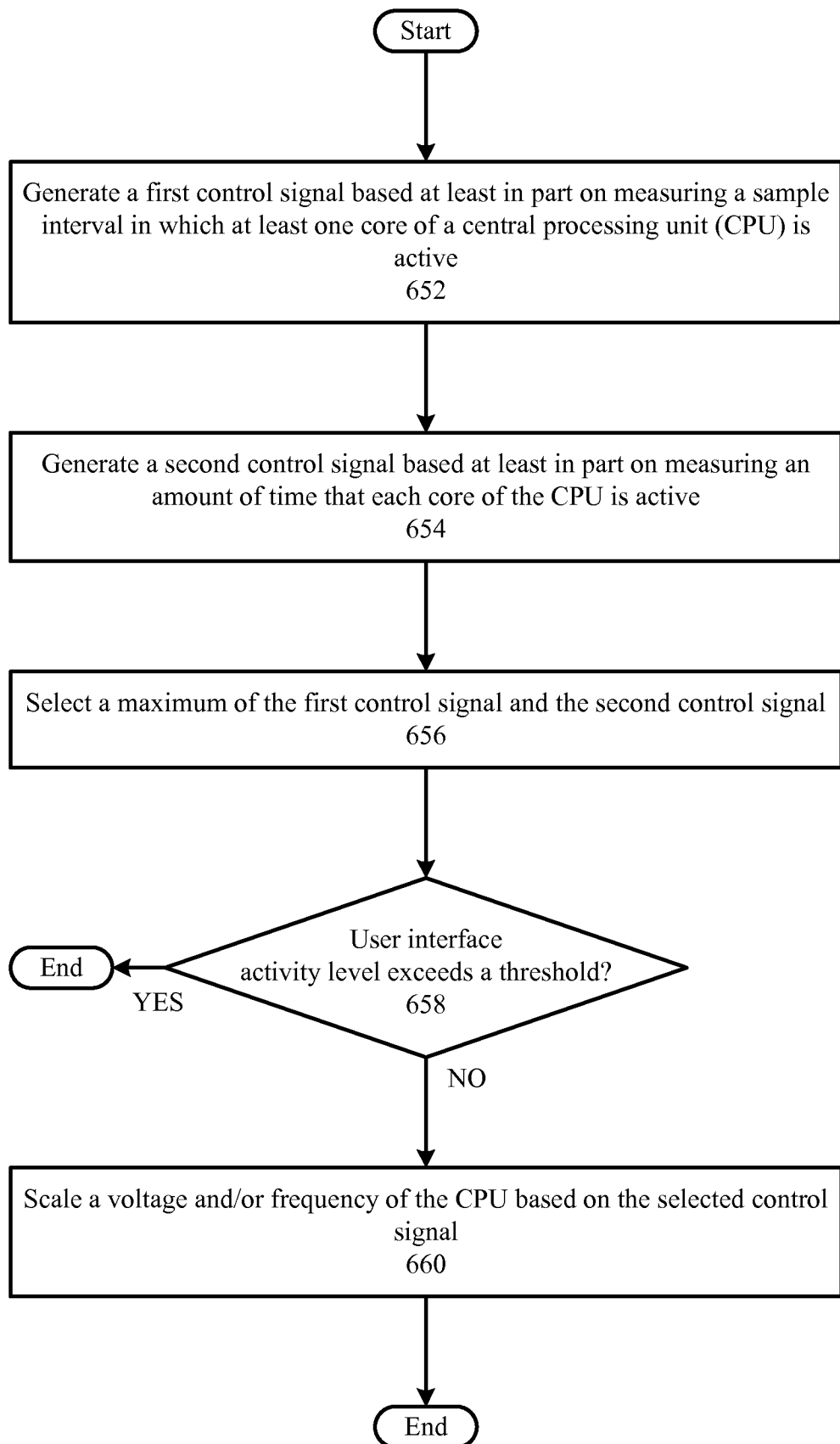


FIG. 6B

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2013/062024**A. CLASSIFICATION OF SUBJECT MATTER****G06F 1/26(2006.01)i, G06F 1/08(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 1/26; G06F 11/30; H04B 3/28; G01R 23/16; G06F 15/173; G06F 15/00; H03K 17/16; G06T 1/20; G06T 1/60; G06F 9/00; G01R 13/34; G06T 1/00; G06F 11/00; G06F 13/00; G06F 1/08

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:

frequency, CPU, jitter, rate, user, interface, GPU, instruction, cycle, memory, control, signal, threshold

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2011-0031996 A1 (AFSHIN MOMTAZ) 10 February 2011 See paragraphs [0005], [0012], [0020], [0032], [0048], [0054], [0056]; and figures 1, 3.	1-2, 7
Y		18-19
A		3-6, 20-23
Y	WO 1999-046608 A1 (LECROY S.A.) 16 September 1999 See page 14, lines 5-10; and figures 12-13.	18-19
X	US 2012-0162234 A1 (PAUL BLINZER et al.) 28 June 2012 See paragraphs [0016], [0037], [0044], [0046], [0080], [0091], [0105]; and figures 1A, 3.	8-10
A		11-12
A	US 2012-0249564 A1 (WANGGEN LIU et al.) 04 October 2012 See paragraphs [0028]-[0029], [0037]; and figures 1-3.	8-12
X	US 2006-0106923 A1 (RAJEEV BALASUBRAMONIAN et al.) 18 May 2006 See paragraphs [0006], [0048]-[0049]; and figures 3-7.	13
Y		14
A		15-17



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents:

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

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

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

22 January 2014 (22.01.2014)

Date of mailing of the international search report

23 January 2014 (23.01.2014)

Name and mailing address of the ISA/KR

Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City,
302-701, Republic of Korea

Facsimile No. +82-42-472-7140

Authorized officer

LEE, Dong Yun

Telephone No. +82-42-481-8734



INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2013/062024

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2005-0125637 A1 (WILCO DIJKSTRA et al.) 09 June 2005 See paragraphs [0009]–[0010], [0114], [0139], [0243]; and figures 1, 8–9.	14
X	US 2007-0250689 A1 (ARIS ARISTODEMOU et al.) 25 October 2007 See paragraphs [0020], [0023], [0047], [0056], [0058]; and figures 2–3.	24–25
A		26–27
A	US 2010-0077232 A1 (SANJEEV JAHAGIRDAR et al.) 25 March 2010 See paragraphs [0017], [0028]–[0029], [0034]; and figures 1, 3.	24–27
A	US 2002-0087291 A1 (BARNES COOPER) 04 July 2002 See paragraphs [0020], [0023]–[0024]; and figures 2, 7.	28–31
A	WO 1995-031782 A1 (AST RESEARCH, INC.) 23 November 1995 See page 4, lines 24–29; page 5, lines 15–20; and figures 1, 13.	28–31

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2013/062024

Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

Group I: Claims 1-7 and 18-23, drawn to a method and a system for updating an operating mode of a central processing unit (CPU) based on a cycle-to-cycle jitter.

Group II: Claims 8-12, drawn to a method for optimizing operations of a central processing unit (CPU) in a mobile computing device having the CPU configured to issue instructions to a graphical processing unit (GPU).

Group III: Claims 13-17, drawn to a method for updating an operating mode of a central processing unit (CPU) with executing instructions.

Group IV: Claims 24-27, drawn to a method for optimizing operations of a central processing unit (CPU) that is configured to perform transactions with a memory controller.

Group V: Claims 28-31, drawn to a method for updating an operating mode of a central processing unit (CPU) when the CPU is executing workloads that are characteristic of traditional desktop/laptop computer applications..

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.
2. ☒ As all searchable claims could be searched without effort justifying an additional fees, this Authority did not invite payment of any additional fees.
3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

- ☐ The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee.
- ☐ The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation.
- ☐ No protest accompanied the payment of additional search fees.

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2013/062024

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2011-0031996 A1	10/02/2011	US 2010-0182045 A1 US 7839161 B2 US 8289045 B2	22/07/2010 23/11/2010 16/10/2012
WO 99-46608 A1	16/09/1999	AT 273519 T AU 1999-30690 A1 AU 1999-30690 B2 CA 2323085 A1 CN 1230683 C0 CN 1292876 A0 DE 69919337 D1 DE 69919337 T2 EP 1062521 A1 EP 1062521 B1 JP 2002-506975 A NZ 506934 A US 2001-0001850 A1 US 6195617 B1 US 6311138 B2	15/08/2004 27/09/1999 27/03/2003 16/09/1999 07/12/2005 25/04/2001 16/09/2004 01/09/2005 27/12/2000 11/08/2004 05/03/2002 20/12/2002 24/05/2001 27/02/2001 30/10/2001
US 2012-0162234 A1	28/06/2012	EP 2652611 A1	23/10/2013
US 2012-0249564 A1	04/10/2012	CN 102656603 A EP 2513860 A1 WO 2011-072419 A1	05/09/2012 24/10/2012 23/06/2011
US 2006-0106923 A1	18/05/2006	CN 101023417 A EP 1771792 A2 EP 1771792 A4 JP 2008-502083 A KR 10-2007-0022386 A KR 10-2007-0072848 A US 2009-0216997 A1 US 7490220 B2 US 8103856 B2 WO 2006-083291 A2 WO 2006-083291 A3	22/08/2007 11/04/2007 17/12/2008 24/01/2008 26/02/2007 06/07/2007 27/08/2009 10/02/2009 24/01/2012 10/08/2006 11/01/2007
US 2005-0125637 A1	09/06/2005	GB 0328524 D0 GB 2411973 A GB 2411973 B JP 2005-174299 A JP 2011-048860 A JP 4708761 B2 US 7689811 B2	14/01/2004 14/09/2005 27/09/2006 30/06/2005 10/03/2011 22/06/2011 30/03/2010
US 2007-0250689 A1	25/10/2007	WO 2007-112031 A2 WO 2007-112031 A3	04/10/2007 02/10/2008

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2013/062024

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2010-0077232 A1	25/03/2010	CN 101676833 A	24/03/2010
		CN 101676833 B	05/12/2012
		DE 102009041723 A1	15/04/2010
		DE 102009041723 B4	29/05/2013
		JP 2012-503233 A	02/02/2012
		JP 5090569 B2	05/12/2012
		KR 10-1254878 B1	15/04/2013
		KR 10-2011-0055674 A	25/05/2011
		TW 201024986 A	01/07/2010
		US 2012-210105 A1	16/08/2012
		US 8028181 B2	27/09/2011
		WO 2010-033446 A2	25/03/2010
		WO 2010-033446 A3	27/05/2010
US 2002-0087291 A1	04/07/2002	AT 338304 T	15/09/2006
		AU 2002-226936 A1	16/07/2002
		AU 2002-226936 A8	16/07/2002
		BR 0116651 A	13/07/2004
		CN 100338581 C0	19/09/2007
		CN 1633644 A	29/06/2005
		DE 60122780 D1	12/10/2006
		DE 60122780 T2	13/09/2007
		EP 1358557 A2	05/11/2003
		EP 1358557 B1	30/08/2006
		HK 1058088 A1	13/04/2007
		TW 552547 A	11/09/2003
		US 6711526 B2	23/03/2004
		WO 02-054244 A2	11/07/2002
		WO 02-054244 A3	12/09/2003
WO 95-31782 A1	23/11/1995	AU 2364095 A	05/12/1995
		CA 2186349 A1	23/11/1995
		CA 2186349 C	23/09/2008
		US 05564015 A	08/10/1996