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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
24 May 2018 (24.05.2018)

WIPO | PCT

(10) International Publication Number
WO 2018/090130 A1

(51) International Patent Classification:
G06F 1/32 (2006.01) *H02J 13/00* (2006.01)

(21) International Application Number:
PCT/CA2017/051322

(22) International Filing Date:
07 November 2017 (07.11.2017)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
15/355,569 18 November 2016 (18.11.2016) US

(71) Applicant: **ATI TECHNOLOGIES ULC** [CA/CA]; One Commerce Valley Drive East, Markham, Ontario L3T7X6 (CA).

(72) Inventors: **PEZESHGI, Shahriar**; 47 Chalmers Rd, Richmond Hill, Ontario L4B1S7 (CA). **HUANG, Jun**; One Commerce Valley Drive East, Markham, Ontario L3T 7X6 (CA). **MOUSAZADEH, Mohammad Hamed**; One Commerce Valley Drive East, Markham, Ontario L3T 7X6 (CA). **DUENAS, Alexander S.**; One Commerce Valley Drive East, Markham, Ontario L3T 7X6 (CA).

(74) Agent: **SMART & BIGGAR**; 1100-150 York Street, Toronto, Ontario M5H 3S5 (CA).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

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

(54) Title: APPLICATION PROFILING FOR POWER-PERFORMANCE MANAGEMENT

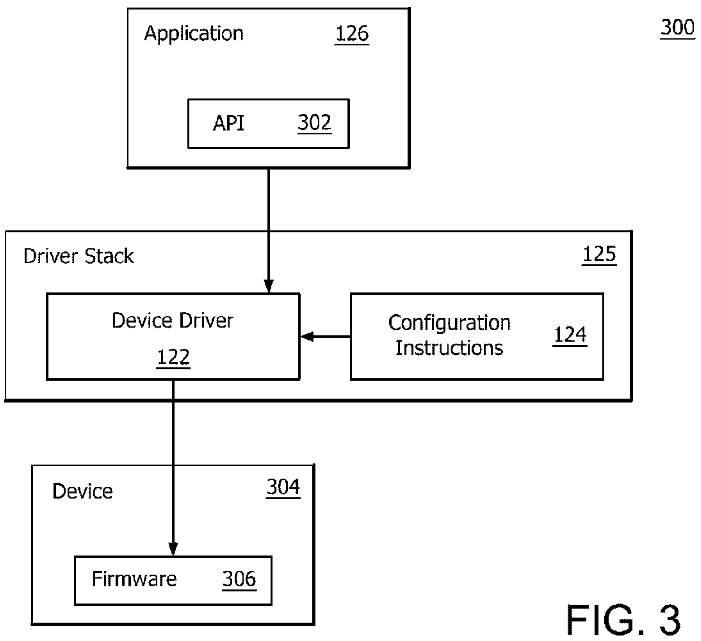

FIG. 3

(57) Abstract: A processing apparatus is provided which includes memory configured to store hardware parameter settings for each of a plurality of applications. The processing apparatus also includes a processor in communication with the memory configured to store, in the memory, the hardware parameter settings, identify one of the plurality of applications as a currently executing application and control an operation of hardware by tuning a plurality of hardware parameters according to the stored hardware parameter settings for the identified application.

WO 2018/090130 A1

[Continued on next page]

Published:

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

APPLICATION PROFILING FOR POWER-PERFORMANCE MANAGEMENT

CROSS REFERENCE TO RELATED APPLICATION

[0001] This application claims the benefit of U.S. Patent Application No. 15/355,569 filed November 18, 2016, which is incorporated by reference as if fully set forth herein.

BACKGROUND

[0002] The hardware of some processing devices is managed by hardware parameters stored at the device (e.g., firmware) which are used to control operation of the hardware. The settings (i.e., dynamic power management (DPM) settings) for these parameters affect both the performance of the hardware as well as the power consumed by the hardware to perform various functions, such as executing applications. Further, the performance and the power consumption are interdependent. For example, an increase in hardware performance level often includes increased power consumption. DPM settings have historically been hardcoded in driver software, however, which cannot be altered without modifying the driver software.

BRIEF DESCRIPTION OF THE DRAWINGS

[0003] A more detailed understanding can be had from the following description, given by way of example in conjunction with the accompanying drawings wherein:

[0004] FIG. 1 is a block diagram of an example device in which application profiling for power-performance management is implemented;

[0005] FIG. 2 is a block diagram illustrating additional details related to execution of processing tasks on the accelerated processing device shown in FIG. 1;

[0006] FIG. 3 is a block diagram illustrating an exemplary interconnection and information flow in an exemplary power-performance management system;

[0007] FIG. 4 is a flow diagram illustrating an exemplary method of receiving hardware parameter settings to manage power-performance; and

[0008] FIG. 5 is a flow diagram illustrating an exemplary method of power-performance management at application runtime.

DETAILED DESCRIPTION

[0009] Efforts have been made to provide global DPM settings for a particular integrated circuit (IC), such as an application-specific integrated circuit (ASIC). These global settings result, however, in varied performance levels. For example, an application executing on the IC at the global settings performs at a first level of performance (e.g., performance level renders a high quality of video display) while another application executes on the IC performs at a second level of performance (e.g., performance level renders a low quality of video display).

[0010] The present application discloses an apparatus and method of managing power and performance by utilizing hardware parameter settings customized for each application to be executed on a device. Customization also includes hardware parameter settings for each application programming interface (API) and each processor of the device. The hardware parameter settings are customized for each application by changing (i.e., overwriting) a configuration file of a driver stack without changing the device driver. The hardware parameter settings for an application are used to tune hardware parameters (e.g., set values of hardware parameters) stored on firmware at the device to control the power consumed by the hardware to execute the application and maintain a level of performance without affecting performance when executing other applications.

[0011] The present application provides a processing apparatus which includes memory configured to store hardware parameter settings for each of a plurality of applications. The processing apparatus also includes a processor in communication with the memory configured to store, in the memory, the hardware parameter settings, identify one of the plurality of applications as a currently executing application and control an operation of hardware by tuning a plurality of hardware parameters according to the stored hardware parameter settings for the identified application.

[0012] The present application provides a computer implemented method of power-performance management. The method includes receiving hardware parameter settings for each of a plurality of applications, storing the hardware parameter settings, executing an application of the plurality of applications, identifying the executing application and controlling an operation of hardware by tuning hardware parameters according to the stored hardware parameter settings for the identified executing application.

[0013] The present application provides a non-transitory computer readable medium including instructions for causing a computer to execute a method of power-performance management. The instructions include storing hardware parameter settings for each of a plurality of applications, executing an application of the plurality of applications, identifying the executing application and controlling an operation of hardware by tuning hardware parameters according to the stored hardware parameter settings for the identified executing application.

[0014] As used herein, programs includes any sequence of instructions to be executed using one or more processors to perform procedures or routines (e.g., operations, computations, functions, processes, jobs). As used herein, execution of programmed instructions (e.g., applications, drivers, operating systems or other software) on a processor includes any of a plurality of stages, such as but not limited to fetching, decoding, scheduling for execution, beginning execution and execution of a particular portion (e.g., rendering of video on full screen) of the programmed instructions. Programmed instructions include parameter settings (e.g., hardware parameter settings) and parameters (e.g., hardware parameters) having tunable (i.e., changeable) values used to control operation of hardware.

[0015] FIG. 1 is a block diagram of an exemplary device 100. The device 100 includes, for example, a computer, a gaming device, a handheld device, a set-top box, a television, a mobile phone, or a tablet computer. As shown in FIG. 1, exemplary device 100 includes a processor 102, memory 104, a storage 106, one or more input devices 108, one or more output devices 110, an input driver 112 and an output driver 114. It is understood that the device 100 can include additional components not shown in FIG. 1.

[0016] Example processor types for processor 102 include a CPU, a GPU, a CPU and GPU located on the same die, or one or more processor cores, wherein each processor core is a CPU or a GPU. Memory 104 is, for example, located on the same die as the processor 102 or located separately from the processor 102. Example memory types for memory 104 include volatile memory, (e.g., random access memory (RAM), dynamic RAM, or a cache) and non-volatile memory (e.g., a hard-disk, motherboard boot read only memory (ROM), and BIOS memory) configured to store, for example firmware which includes hardware parameters, as described in more detail below.

[0017] Example storage types for storage 106 include a fixed or removable storage, for example, a hard disk drive, a solid state drive, an optical disk, or a flash drive. Example input device types for input device 108 include a keyboard, a keypad, a touch screen, a touch pad, a detector, a microphone, an accelerometer, a gyroscope, a biometric scanner, or a network connection (e.g., a wireless local area network card for transmission and/or reception of wireless IEEE 802 signals). Example output device types for output devices 110 include a display, a speaker, a printer, a haptic feedback device, one or more lights, an antenna, or a network connection (e.g., a wireless local area network card for transmission and/or reception of wireless IEEE 802 signals).

[0018] The input driver 112 communicates with the processor 102 and the input devices 108, and permits the processor 102 to receive input from the input devices 108. The output driver 114 communicates with the processor 102 and the output devices 110, and permits the processor 102 to send output to the output devices 110. It is noted that the input driver 112 and the output driver 114 are optional components and that the device 100 will operate in the same manner if the input driver 112 and the output driver 114 are not present.

[0019] As shown in FIG. 1, the output driver 114 (e.g., graphics card) includes an accelerated processing device (APD) 116 which is coupled to a display device 118. The APD 116 is configured to accept compute commands and graphics rendering commands from processor 102, to process those compute and graphics rendering commands, and to provide pixel output to display device 118 for display. As described in further detail below, the APD 116 includes one or more parallel processing units configured to perform computations in accordance with a single-instruction-multiple-data (SIMD) paradigm. Although various functionality is described herein as being performed by or in conjunction with the APD 116, the functionality described as being performed by the APD 116 is also performed by other computing devices having similar capabilities that are not driven by a host processor (e.g., processor 102) and configured to provide graphical output to a display device 118. The functionality described herein is, for example, performed by any processing system that performs processing tasks in accordance with a SIMD paradigm. Alternatively, the functionality described herein is performed by computing systems that do not perform processing tasks in accordance with a SIMD paradigm.

[0020] FIG. 2 is a block diagram illustrating additional details related to execution of processing tasks on the APD 116 of the device 100 shown in FIG. 1. The processor 102 maintains, in system memory 104, one or more control logic modules which include programmed instructions for execution by the processor 102. Programmed instructions can also be included in processor 102 (e.g., CPU). The control logic modules include an operating system 120, a device driver 122 (e.g., kernel mode driver, user mode driver, universal mode driver), applications 126 which execute on the APD 116 and configuration instructions 124 (e.g., instructions in a configuration file, blob file, extension file, and the like) which store predetermined hardware parameter settings for each executable application 126. As shown in FIG 2, the device driver 122, and configuration instructions 124 are part of a driver stack 125. Driver stack 125 also includes, for example, programmed instructions which support the tuning of hardware parameters (e.g., clock speed parameters, clock voltage parameters, clock gating parameters) to manage the power consumption by the hardware during execution of each application.

[0021] These control logic modules control various aspects of the operation of the processor 102 and the APD 116. For example, the operating system 120 directly communicates with hardware (e.g., CPU, GPU and compute units 132) and provides an interface to the hardware for other software executing on the processor 102. The device driver 122 controls operation of the APD 116 by, for example, providing an API to software (e.g., applications 126) executing on the processor 102 to access various functionality of the APD 116. The device driver 122 also includes a just-in-time compiler that compiles programs for execution by processing components of the APD 116.

[0022] The APD 116 executes commands and programs for selected functions, such as graphics operations and non-graphics operations suited, for example, to perform parallel processing. The APD 116 is used, for example, to execute graphics pipeline operations such as pixel operations, geometric computations, and rendering an image to display device 118 based on commands received from the processor 102. The APD 116 also executes compute processing operations that are not directly related to graphics operations, such as operations related to video, physics simulations, computational fluid dynamics, or other tasks, based on commands received from the processor 102.

[0023] Exemplary processor types for APD 116 include a CPU, a GPU, a CPU and GPU located on the same die, or one or more processor cores (i.e., compute units) 132 wherein each processor core is a CPU or a GPU. Each compute unit (i.e., compute core) 132 includes one or more SIMD units 138 each configured to perform operations at the request of the processor 102 in a parallel manner according to a SIMD paradigm. The SIMD paradigm is one in which multiple processing elements share a single program control flow unit and program counter and thus execute the same program but are able to execute that program with different data. In one example, each SIMD unit 138 includes sixteen lanes, where each lane executes the same instruction at the same time as the other lanes in the SIMD unit 138 but can execute that instruction with different data. Lanes can be switched off with predication if not all lanes need to execute a given instruction. Predication can also be used to execute programs with divergent control flow. More specifically, for programs with conditional branches or other instructions where control flow is based on calculations performed by an individual lane, predication of lanes corresponding to control flow paths not currently being executed, and serial execution of different control flow paths allows for arbitrary control flow.

[0024] A scheduler 136 is configured to perform operations related to scheduling various units of execution (e.g., work groups and wavefronts) on different compute units 132 and SIMD units 138. Execution of processing tasks on the APD 116 is suitable for graphics related operations such as pixel value calculations, vertex transformations, and other graphics operations. A graphics pipeline 134 which accepts graphics processing commands from the processor 102 can therefore provide computation tasks to the compute units 132 for execution in parallel.

[0025] The compute units 132 are also used to perform computation tasks not related to graphics or not performed as part of the "normal" operation of a graphics pipeline 134 (e.g., custom operations performed to supplement processing performed for operation of the graphics pipeline 134). An application 126 or other software executing on the processor 102 transmits programs that define such computation tasks to the APD 116 for execution.

[0026] FIG. 3 is a block diagram 300 illustrating an exemplary interconnection and information flow in an exemplary power-performance management system. As shown in FIG. 3, the system includes an application 126 provided with API 302 (which represents one or more APIs), a driver stack 125 including device driver 122 and configuration instructions 124 and device 304 (e.g., output driver 114 shown in FIG. 1) including firmware 306 configured to execute at device 304.

[0027] The driver stack 125 includes device driver 122 used to interface between the operating system 120 and the firmware 306 and configuration instructions 124. The configuration instructions 124 include, for each application 126 to be processed, predetermined (e.g., determined prior to application runtime) hardware parameter settings, used to tune the plurality of hardware parameters configured to control the operation of the hardware during execution of each application 126.

[0028] Firmware 306 includes hardware parameters and associated values to control operation of hardware of the device 304 (e.g., graphics card) and provide an interface between the hardware (e.g., APD 116) of the device 304 and device driver 122. As described above, firmware is stored in non-volatile memory (e.g., a hard-disk, motherboard boot read only memory (ROM), and BIOS memory). Processor 102 is configured to identify an executing application. For example, an application executing on APD 116 at device 304 (e.g., output driver 114) is identified and firmware 306 is read from non-volatile memory (e.g., portion of memory 104) to be processed at device 304, as shown in FIG. 3. Alternatively, an application executing on a processor at device 100, which does not include output driver, is identified. The firmware 306 is used, along with the device driver 122, to control operation of hardware (e.g., APD 116 of output driver 114, one or more other processors of device 100 as well as encoders, decoders, memory cells and circuitry (not shown)).

[0029] Examples of hardware parameters include dynamic power management (DPM) parameters (e.g., clock speed parameters, clock voltage parameters, and clock gating parameters), memory timing parameters, heat generated (e.g., thermal design power (TDP)) and other parameters used to control the power distribution to execute the identified application. The APD 116 is configured to execute (e.g., schedule for execution, execute) an application 126 using, for example, the operating system 120, the device driver 122 and the configuration instructions 124. For example, the operating system 120 communicates with firmware 306 and provides an interface to the hardware for application 126 executing on the APD 116. The device driver 122 controls operation of the APD 116 by, for example, providing API 302 to applications 126 executing on the APD 116 to access various functionality of the APD 116.

[0030] Processor 102 identifies the application being executed at device 304, for example, via an application name. Processor 102 is also configured to identify the executing application 126 via an API identifier which identifies the API 302 used by the application 126. API 302 includes, for example, instructions used to interface (e.g., make requests) with the device driver 122. Examples of APIs include but are not limited to open computing language (OpenCL), open graphics library (OpenGL) and DirectX APIs.

[0031] Hardware parameter settings for an application 126 are stored in configuration instructions 124. When a new application becomes available (e.g., available to execute) for execution by a particular processor or device (e.g., graphics card), the application 126, using the provided API 302, is tested (e.g., executed) for the device with different parameter settings resulting in different levels of power consumption and performance. Each device (e.g., device 100, output driver 114, device 304) includes hardware (e.g., an IC, a processor such as processor 102 or APD 116, memory cells, an encoder, a decoder and circuitry). The hardware is identified, for example, using a hardware identifier (e.g., device ID or a revision ID). When the hardware parameter settings for the identified hardware are determined for the new application 126, the settings are sent, for example, as part of updated configuration instructions (e.g., configuration file, blob file or other file) via a network (not shown). The settings are received (e.g., at device 100) which includes the device (e.g., output driver 114, device 304) and hardware (e.g., APD 116) When the updated configuration instructions are received at the device 100, the parameter settings for the new application 126 are stored (e.g., via processor 102) in volatile memory (e.g., volatile portion of memory 104) at the device 100 by overwriting the configuration instructions 124 of the driver stack 125 without changing the device driver 122.

[0032] When the executing application 126 is identified, processor 102 controls the operation of the hardware by executing firmware 306 (e.g., at the device 304) and tuning a plurality of hardware parameters (e.g., clock voltage) according to the stored hardware parameter settings in the configuration instructions 124 corresponding to the executing application 126. Tuning is performed by setting the values associated with hardware parameters according to the hardware parameter settings stored in the configuration instructions 124. For example, when the executing application 126 is identified, device driver 122 parses the configuration instructions 124 and reads the settings from the configuration instructions 124 for the identified application 126 and API 302. The stored data is passed to device driver 122 (e.g., kernel mode driver) and additional structures, which support the stored settings to manage power and performance.

[0033] The stored settings include, for example, increasing or decreasing a value of the clock voltage parameter based on a comparison of an activity level of the hardware to a value of an activity level threshold parameter. The activity level of the hardware and the value of the activity level threshold parameter correspond, for example, to a percentage or fraction of time a processor executes during a sampling interval (e.g., 10 ms), a number of times a processor executes during a sampling interval, or another metric for measuring activity level of a processor executes during a sampling interval. For example, the value of the clock voltage parameter is decreased when an activity level of the hardware during a sampling interval is less than the value of the activity level threshold parameter and the value of the clock voltage parameter is increased when the activity level of the hardware during a sampling interval is equal to or greater than the value of the activity level threshold parameter. The settings also include, for example, setting a limit for the value of the clock voltage parameter.

[0034] FIG. 4 is a flow diagram 400 illustrating an exemplary method of receiving hardware parameter settings for power-performance management.

[0035] As shown in block 402 of FIG. 4, the method 400 includes receiving hardware parameter settings at a device (e.g., via a device identifier) for a plurality of hardware parameters used to control operation of hardware. The customized hardware settings are received at a device, for example, as part of a set of configuration instructions (e.g., configuration file) over a network (not shown) for updating the currently stored configuration instructions.

[0036] The configuration file is updated at block 404 of method 400. Updating the configuration instructions include, for example, overwriting each portion of the configuration file (i.e., overwriting the file itself), overwriting one or more portions of the configuration file (e.g., updating hardware parameter settings for previously received settings for one or more applications) and adding hardware parameter settings for one or more newly tested applications. Currently stored parameter settings (e.g., previously received settings for one or more applications) are identified, for example, via application names or API identifiers. The instructions are stored by overwriting the configuration instructions without overwriting or changing the device driver. That is, the settings are not hardcoded in the device driver and can be altered without modifying the device driver

[0037] The updated configuration instructions are stored (e.g., as firmware) in non-volatile memory (e.g., hard-disk, motherboard boot ROM, and the like). Identifiers (e.g., application names and API identifiers) are also stored to identify hardware parameter settings with corresponding application names and API identifiers provided for the applications. When an application is executed, the instructions, including the identified hardware parameter settings for the corresponding application, are executed via memory at the device.

[0038] FIG. 5 is a flow diagram 500 illustrating an exemplary method of power-performance management at application runtime.

[0039] As shown in block 502 of FIG. 5, the method 500 includes executing an application. The application is executed, for example, by fetching the application, decoding the application, scheduling the application for execution, beginning execution of the application and executing a particular portion (e.g., rendering of video on full screen) of the application.

[0040] As shown in block 504 of FIG. 5, the method 500 includes identifying the executing application. The application is identified, for example, by an application identifier that identifies the application and an API identifier which identifies an API provided for the application. Because each application and each API can result in a different performance level using the same power distribution values, the settings are customized for each application and each API to control power consumed by the hardware to execute the application and maintain a level of performance without affecting performance when executing another application.

[0041] As shown at decision block 506 of FIG. 5, the method includes determining whether hardware parameter settings are stored for the identified application. For example, it is determined whether hardware parameter settings, for the identified application, are stored in the configuration file described in FIG. 4. When parameter settings The hardware parameter settings are stored for the identified application

[0042] As shown in block 508 of method 500 in FIG. 5, when it is determined that hardware parameter settings are stored for the identified application (e.g., via an application identifier and an API identifier), the hardware parameters are tuned to execute the application according to the stored hardware parameter settings for the identified application. The settings are executed via memory at the device. The tuning is used to control the operation of the hardware, such as controlling power that is consumed by the hardware to execute the identified application while maintaining a performance level of execution. For example, the power distribution is managed by tuning a clock voltage parameter (e.g., changing a value of the clock voltage parameter) based on a comparison of an activity level of the hardware during a sampling interval to an activity level threshold parameter for the sampling interval. As shown in block 510 of method 500 in FIG. 5, when it is determined that hardware parameter settings are not stored for the identified application (e.g., hardware parameter settings are not yet received for an identified application), the hardware parameters are tuned to execute the application according to stored global hardware parameter settings.

[0043] It should be understood that many variations are possible based on the disclosure herein. Although features and elements are described above in particular combinations, each feature or element can be used alone without the other features and elements or in various combinations with or without other features and elements.

[0044] The methods provided include implementation in a general purpose computer, a processor, or a processor core. Suitable processors include, by way of example, a general purpose processor, a special purpose processor, a conventional processor, a digital signal processor (DSP), a plurality of microprocessors, one or more microprocessors in association with a DSP core, a controller, a microcontroller, Application Specific Integrated Circuits (ASICs), Field Programmable Gate Arrays (FPGAs) circuits, any other type of integrated circuit (IC), and/or a state machine. Such processors can be manufactured by configuring a manufacturing process using the results of processed hardware description language (HDL) instructions and other intermediary data including netlists (such instructions capable of being stored on a computer readable media). The results of such processing can be maskworks that are then used in a semiconductor manufacturing process to manufacture a processor which implements application profiling for power-performance management.

[0045] The methods or flow charts provided herein can be implemented in a computer program, software, or firmware incorporated in a non-transitory computer-readable storage medium for execution by a general purpose computer or a processor. Examples of non-transitory computer-readable storage mediums include a ROM, a random access memory (RAM), a register, cache memory, semiconductor memory devices, magnetic media such as internal hard disks and removable disks, magneto-optical media, and optical media such as CD-ROM disks, and digital versatile disks (DVDs).

CLAIMS

What is claimed is:

1. A processing apparatus comprising:

memory configured to store hardware parameter settings for each of a plurality of applications;

a processor, in communication with the memory, configured to:

store, in the memory, the hardware parameter settings;

identify one of the plurality of applications as a currently executing application; and

control an operation of hardware by tuning a plurality of hardware parameters according to the stored hardware parameter settings for the identified one application.

2. The processing apparatus of claim 1, wherein the hardware comprises an accelerated processing device, comprising a plurality of processor types, configured to execute the application.

3. The processing apparatus of claim 1, wherein the processor is further configured to identify the application executing on the hardware via an application identifier which identifies the executing application and an application programming interface identifier which identifies an application programming interface provided to the executing application.

4. The processing apparatus of claim 1, wherein the hardware includes a hardware identifier and the hardware parameter settings are used to control the operation of the hardware identified by the hardware identifier.

5. The processing apparatus of claim 1, wherein the plurality of hardware parameters are power management parameters configured to control an amount of power that is consumed by the hardware to execute the identified one application while maintaining a performance level of execution, and

the processor is further configured to control the amount of power that is consumed by the hardware by tuning the plurality of power management parameters according to the stored hardware parameter settings for the identified one application.

6. The processing apparatus of claim 5, wherein the power management parameters comprise a clock voltage parameter and an activity level threshold parameter for a sampling interval, and

the processor is further configured to tune the clock voltage parameter by:

comparing an activity level of the hardware during the sampling interval to the activity level threshold parameter for the sampling interval; and

changing a value of the clock voltage parameter based on the comparing of the activity level of the hardware during the sampling interval to the activity level threshold parameter for the sampling interval.

7. The processing apparatus of claim 1, wherein the processor is further configured to:

determine whether the hardware parameter settings are stored for the identified one application;

when it is determined that hardware parameter settings are stored for the identified one application, tune the hardware parameters to execute the application according to the stored hardware parameter settings for the identified one application; and

when it is determined that hardware parameter settings are not stored for the identified one application, tune the hardware parameters to execute the application according to stored global hardware parameter settings.

8. The processing apparatus of claim 1, wherein the processor is further configured to store the hardware parameter settings in a configuration

file separate from a device driver which is configured to control operation of the hardware by using the configuration file.

9. The processing apparatus of claim 8, wherein the processor is further configured to update the configuration file without modifying the device driver when new parameter settings are received for one or more of the applications.

10. The processing apparatus of claim 1, wherein the memory is further configured to store an executable device driver and the processor is further configured to

store the hardware parameter settings in a non-executable configuration file; and

control the operation of the hardware by using the executable device driver to tune the plurality of hardware parameters according to the stored hardware parameter settings stored in the non-executable configuration file.

11. A computer implemented method of power-performance management, the method comprising:

receiving hardware parameter settings for each of a plurality of applications;

storing the hardware parameter settings;

executing an application of the plurality of applications;

identifying the executing application; and

controlling an operation of hardware by tuning hardware parameters according to the stored hardware parameter settings for the identified executing application.

12. The method of claim 11, further comprising:

determining whether the hardware parameter settings are stored for the identified executing application;

when it is determined that hardware parameter settings are stored for the identified executing application, tuning the hardware parameters to execute the application according to the stored hardware parameter settings for the identified executing application; and

when it is determined that hardware parameter settings are not stored for the identified executing application, tuning the hardware parameters to execute the application according to stored global hardware parameter settings.

13. The method of claim 11, further comprising providing one or more application programming interfaces to the executing application, and wherein identifying the executing application comprises identifying the one or more application programming interfaces provided to the executing application.

14. The method of claim 11, further comprising storing the hardware parameter settings in a configuration file separate from a device driver which is configured to control operation of the hardware by using the hardware parameter settings stored in the configuration file.

15. The method of claim 14, further comprising updating the configuration file without modifying the device driver when the hardware parameter settings are stored for each of the plurality of applications.

16. The method of claim 11, wherein the plurality of hardware parameters are power management parameters which control an amount of power that is consumed by the hardware to execute the identified executing application while maintaining a performance level of execution, and

the method further comprises controlling the amount of power that is consumed by the hardware by tuning the plurality of power management parameters according to the stored hardware parameter settings.

17. The method of claim 16, wherein the power management parameters comprise a clock voltage parameter and an activity level threshold parameter for a sampling interval, and

the method further comprises tuning the clock voltage parameter by:

comparing an activity level of the hardware during the sampling interval to the activity level threshold parameter for the sampling interval; and

changing a value of the clock voltage parameter based on the comparison of the activity level of the hardware during the sampling interval to the activity level threshold parameter for the sampling interval.

18. The method of claim 17, further comprising

storing an executable device driver which includes instructions to control the operation of the hardware;

storing the hardware parameter settings in a non-executable configuration file; and

controlling the operation of the hardware by using the executable device driver to tune the plurality of hardware parameters according to the stored hardware parameter settings stored in the non-executable configuration file.

19. A non-transitory computer readable medium comprising instructions for causing a computer to execute a method of power-performance management, the instructions comprising:

storing hardware parameter settings for each of a plurality of applications;

executing an application of the plurality of applications;

identifying the executing application; and

controlling an operation of hardware by tuning hardware parameters according to the stored hardware parameter settings for the identified executing application.

20. The non-transitory computer readable medium of claim 19, wherein the instructions further comprise:

storing, for each of the plurality of applications, the hardware parameter settings in a configuration file separate from a device driver which is configured to control operation of the hardware by using the hardware parameter settings stored in the configuration file; and

updating the configuration file without modifying the device driver when the hardware parameter settings are stored for each of the plurality of applications.

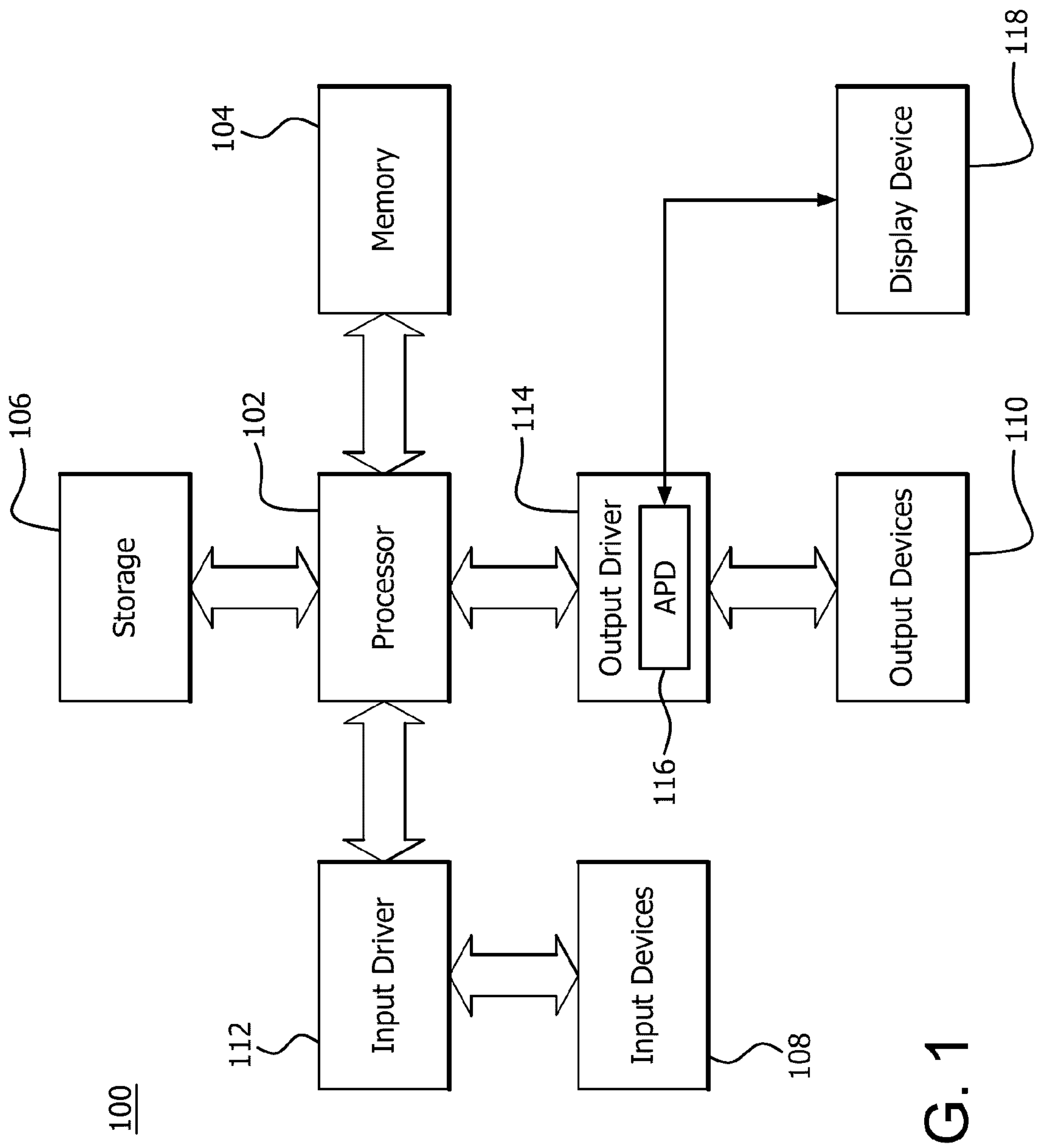
100
Storage
106
Processor
102
Memory
104
Input Driver
112
Input Devices
108
Output Driver
114
APD
116
Output Devices
110
Display Device
118

FIG. 1

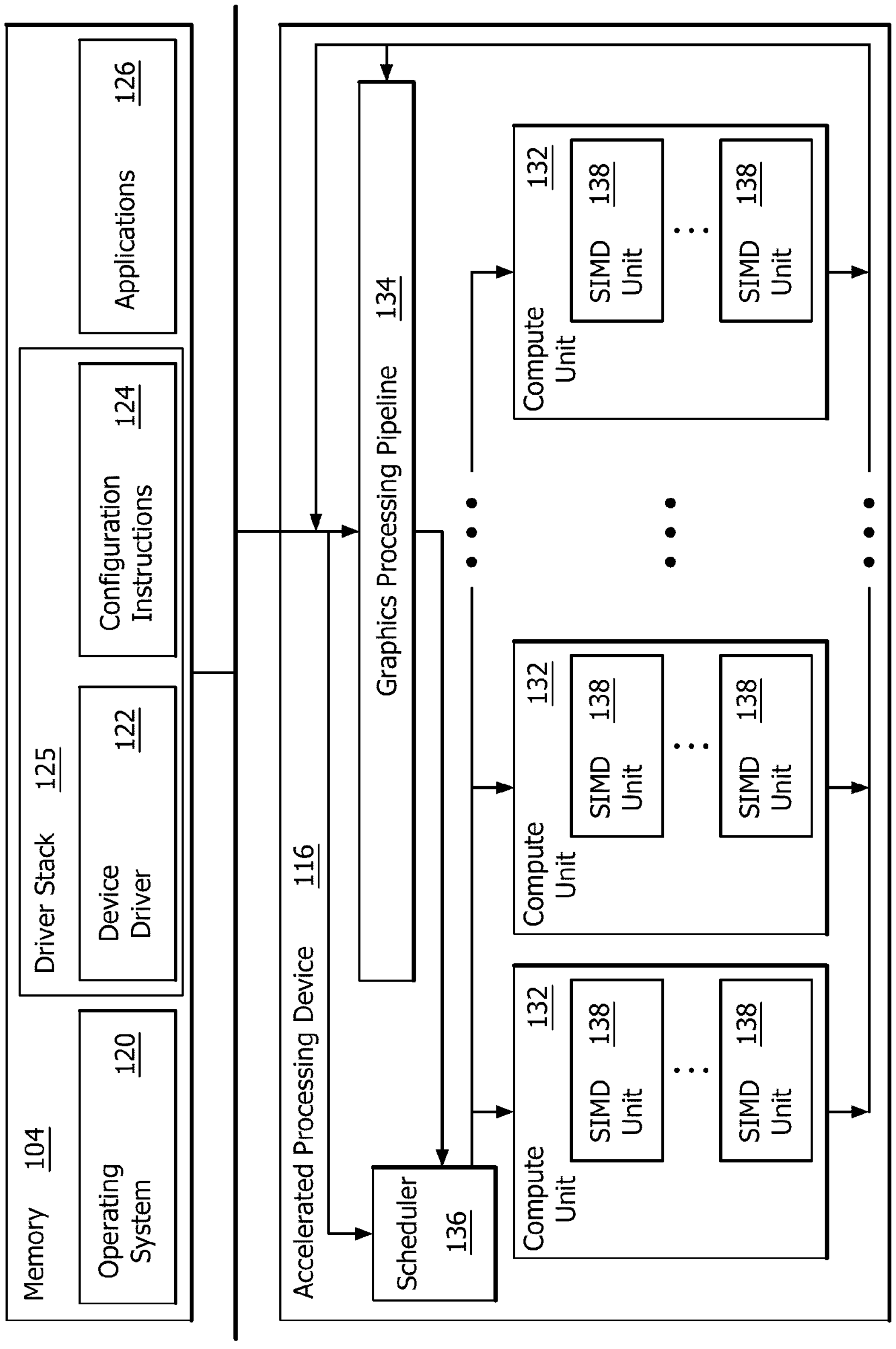
Memory 104
Operating System 120
Driver Stack 125
Device Driver 122
Configuration Instructions 124
Applications 126
Accelerated Processing Device 116
Scheduler 136
Graphics Processing Pipeline 134
Compute Unit 132
SIMD Unit 138
SIMD Unit 138
Compute Unit 132
SIMD Unit 138
SIMD Unit 138
Compute Unit 132
SIMD Unit 138
SIMD Unit 138

FIG. 2

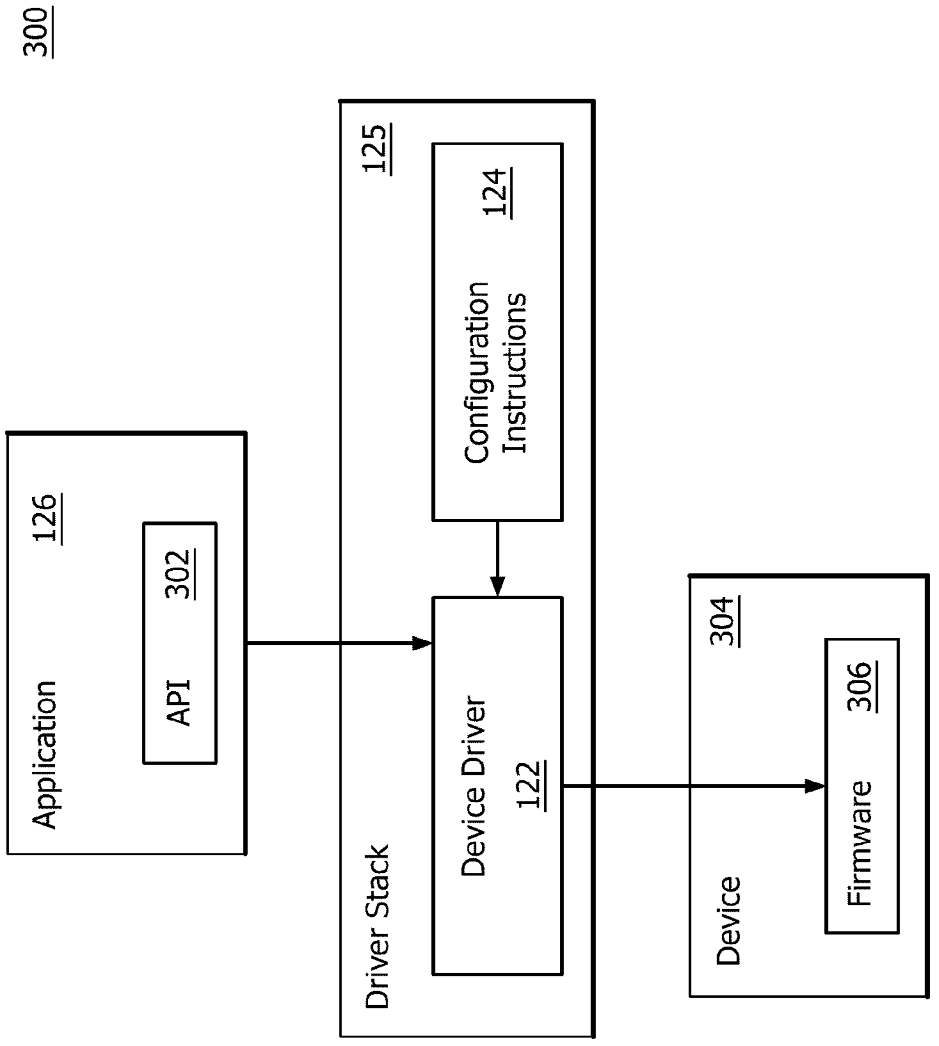
300
Application
126
API
302
Driver Stack
125
Configuration Instructions
124
Device Driver
122
Device
304
Firmware
306

FIG. 3

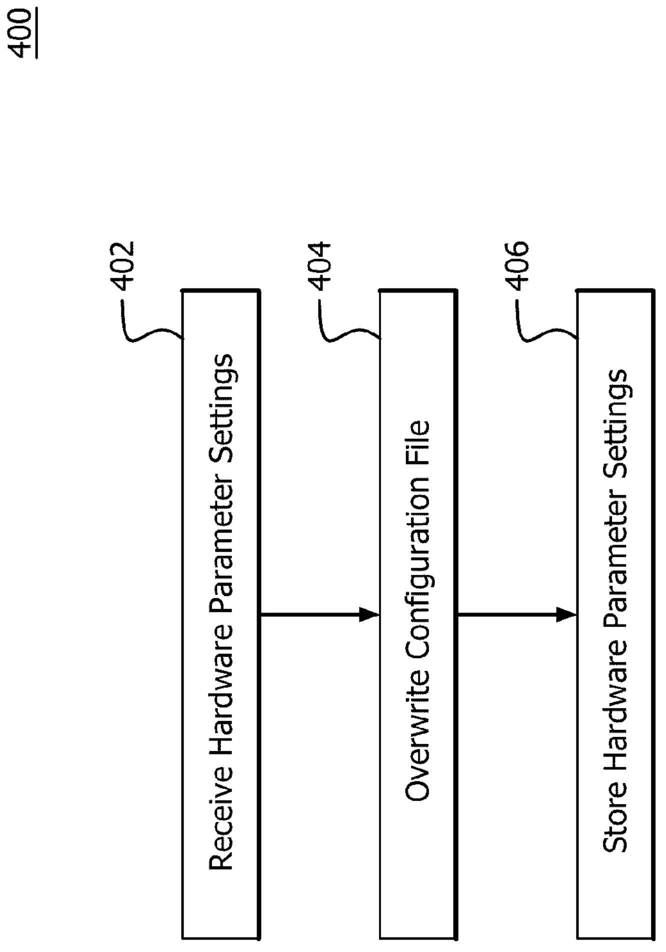
400
402
Receive Hardware Parameter Settings
404
Overwrite Configuration File
406
Store Hardware Parameter Settings

FIG. 4

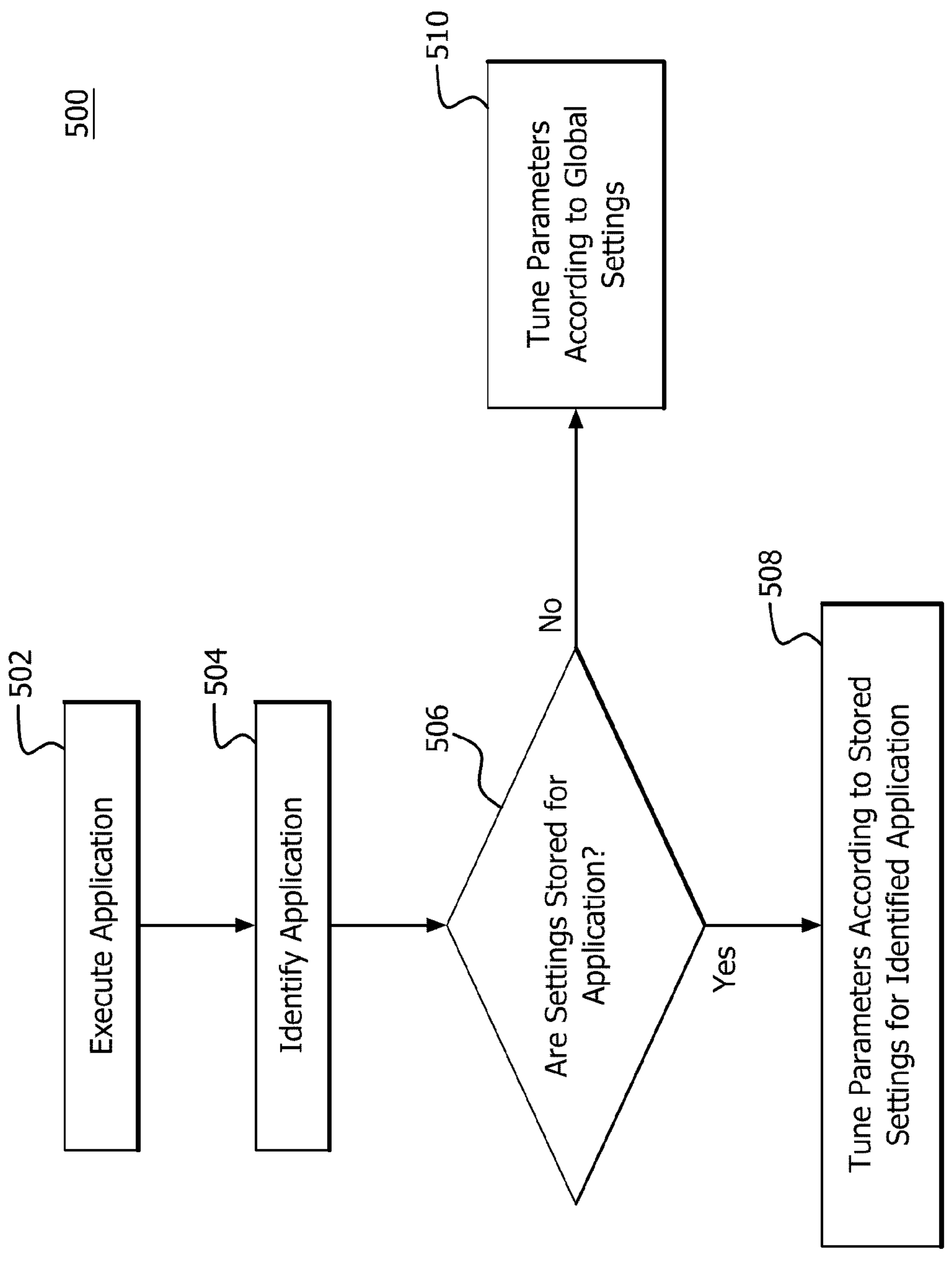
500
502
Execute Application
504
Identify Application
506
Are Settings Stored for Application?
No
Yes
510
Tune Parameters According to Global Settings
508
Tune Parameters According to Stored Settings for Identified Application

FIG. 5

INTERNATIONAL SEARCH REPORT	International application No. **PCT/CA2017/051322**

A. CLASSIFICATION OF SUBJECT MATTER
IPC: ***G06F 1/32*** (2006.01) , ***H02J 13/00*** (2006.01)

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)
IPC (2006.01): G06F 1/32, H02J 13/00

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

Electronic database(s) consulted during the international search (name of database(s) and, where practicable, search terms used)
Database: EPOQUE, Canadian Patent Database, Google
Keywords: hardware, parameter?, setting?, application?, tun+, dynamic?, power+, resource?, manag+, adjust+, chang+, driver?

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2005/0204125 A1 (Chin), 15 September 2005 (15-09-2005) *paragraphs [0007], [0016]-[0018], [0021]-[0025], [0033] *Figures 2, 3	1-20
A	US 8,276,133 B1 (Lebaredian et al.), 25 September 2012 (25-09-2012) *whole document	1-20
A	US 2006/0080677 A1 (Louie), 13 April 2006 (13-04-2006) *whole document	1-20

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

*	Special categories of cited documents:	"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
"A"	document defining the general state of the art which is not considered to be of particular relevance	"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
"E"	earlier application or patent but published on or after the international filing date	"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
"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)	"&"	document member of the same patent family
"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		

Date of the actual completion of the international search 06 February 2018 (06-02-2018)	Date of mailing of the international search report 08 February 2018 (08-02-2018)
Name and mailing address of the ISA/CA Canadian Intellectual Property Office Place du Portage I, C114 - 1st Floor, Box PCT 50 Victoria Street Gatineau, Quebec K1A 0C9 Facsimile No.: 819-953-2476	Authorized officer Dominic Lam (819) 576-2195

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CA2017/051322

Patent Document Cited in Search Report	Publication Date	Patent Family Member(s)	Publication Date
US2005204125A1	15 September 2005 (15-09-2005)	US2005204125A1	15 September 2005 (15-09-2005)
		TWI237793B	11 August 2005 (11-08-2005)
		TW200530918A	16 September 2005 (16-09-2005)
US8276133B1	25 September 2012 (25-09-2012)	None	
US2006080677A1	13 April 2006 (13-04-2006)	US2006080677A1	13 April 2006 (13-04-2006)
		US7636921B2	22 December 2009 (22-12-2009)
		EP1662386A2	31 May 2006 (31-05-2006)
		EP1662386A3	26 September 2007 (26-09-2007)
		EP1662386B1	17 June 2015 (17-06-2015)
		US2010115534A1	06 May 2010 (06-05-2010)
		US8051435B2	01 November 2011 (01-11-2011)