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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
07 September 2023 (07.09.2023)

WIPO | PCT

(10) International Publication Number
WO 2023/167616 A1

(51) International Patent Classification:
G06F 1/324 (2019.01) *G06F 1/3234* (2019.01)
G06F 1/3206 (2019.01) *H04W 52/00* (2009.01)

(21) International Application Number:
PCT/SE2022/050214

(22) International Filing Date:
04 March 2022 (04.03.2022)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **TELEFONAKTIEBOLAGET LM ERICSSON (PUBL)** [SE/SE]; 164 83 Stockholm (SE).

(72) Inventors: **EKER, Johan**; Äpplehagen 6, 223 55 Lund (SE). **FIALLOS, Edgard**; Trubadurvägen 2 Unit 1402, 171 69 Solna (SE). **MILLNERT, Victor**; Friisgatan 29, 214 21 Malmö (SE). **GÉHBERGER, Dániel**; APT 8 - 2345 rue Grand Trunk, Montreal, Québec H3K1M8 (CA).

(74) Agent: **PATENT UNIT LUND, Ericsson AB**; Nya Vattentornet, 221 83 Lund (SE).

(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, IT, JM, 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, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, 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, SC, 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).

Published:
— *with international search report (Art. 21(3))*
— *in black and white; the international application as filed contained color or greyscale and is available for download from PATENTSCOPE*

(54) Title: NETWORK AWARE CPU CLOCK FREQUENCY GOVERNOR FOR CLOUD INFRASTRUCTURE

(57) Abstract: A CPU clock frequency governor for a CPU in a cloud RAN that uses network information to control the frequency of the clock signal supplied to the CPU to conserve power without contributing to increased latency. The CPU receives radio data for processing at the start of a period. The CPU offloads heavy computations involved in processing the 5G workloads to a GPU or other accelerator. The CPU enters an idle state when the radio data is handed over to a GPU for processing. A first timer is set based on an estimated execution time of the GPU to ramp-up the CPU clock before the radio data is handed back to the CPU. If the processing of the radio data by the CPU is finished before the deadline, the CPU enters an idle state and a second timer is set to ramp-up the CPU clock before the start of the next period.

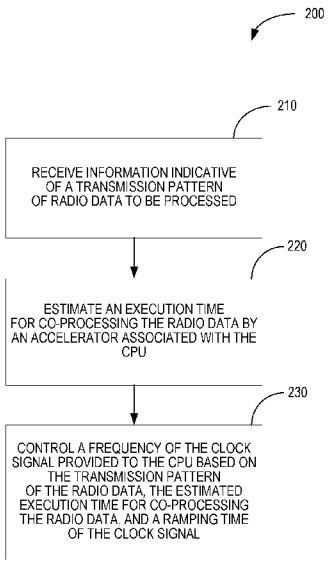

FIG. 11

WO 2023/167616 A1

NETWORK AWARE CPU CLOCK FREQUENCY GOVERNOR FOR CLOUD INFRASTRUCTURE

TECHNICAL FIELD

The present disclosure relates generally to a cloud-based communication networks and, more particularly to a network aware CPU clock frequency governor for a cloud infrastructure in a virtualized radio access network (RAN).

BACKGROUND

Cloud technology has swiftly transformed the information and communications technology (ICT) industry and it is continuing to spread to new areas. Many traditional ICT applications have relaxed timing or performance requirements and are suitable for cloud deployment. Cloud systems are usually built on top on large scale commodity servers, typically x86 based systems. Different kinds of accelerators and new software scheduling techniques are needed to apply the cloud concepts outside the traditional ICT domain to more mission critical use cases such as telecom, industrial automation and real-time analytics.

Network Function Virtualization (NFV) is a virtualization technology for communication networks that eschews proprietary hardware and implements network functions (NFs) in the communication network as software running on industry-standard servers supported by high volume storage. A virtual network function (VNF) or containerized network function (CNF) implemented as a software component can be deployed at essentially any location within a cloud infrastructure without the need of installing new equipment. NFV enables rapid deployment of new services, ease of scalability, and reduced redundancy in the NFs.

The Internet Engineering Task Force (IETF) NFV initiative is standardizing a virtual networking infrastructure and a VNF architecture for wireless communication networks. The main focus is currently on core network (CN) functions, such as a virtual Internet Protocol (IP) Multimedia Subsystem (vIMS), virtual Evolved packet Core (vEPC), and virtual Mobility Management Entity (vMME). However, the possibility of moving Radio Access Network (RAN) functions to the cloud is being explored.

Cloud RAN is a new architecture for RANs where the certain RAN functions are moved into the cloud and realized using commercial-off-the-shelf (COTS) hardware. In a cloud RAN, the functions of a 5G NodeB (gNB) in a 5G network, also known as a base station, are divided into two parts: a Radio Unit (RU) and a baseband unit (BBU). The RU serves as the radio part of the base station and contains the radio frequency (RF)

circuitry and antennas for transmitting signals to and receiving signals from user equipment (UEs) over a wireless communication channel. The BBU serves as the control part of the base station. The BBU processes signals transmitted and received by the gNB and handles most control functions, such as scheduling, power control, etc. In this split architecture, the BBUs can be implemented in as virtual NFs that can be pooled and shared by multiple RUs.

While commodity servers are becoming increasingly powerful, they still fall short for certain tasks compared to existing proprietary solutions. Moving RAN functions to the cloud currently involves a trade-off between efficiency of computations provided by proprietary solutions and lower capital expenditures (capex) for operators when using COTS hardware. To get reasonable performance from COTS hardware, the software architecture and resource scheduling must be carefully designed. In particular the COTS hardware must be augmented with accelerators, e.g., Graphics Processing Units (GPUs), Field Programmable Gate Arrays (FPGAs) and Application Specific Integrated Circuits (ASICs). These accelerators are typically challenging to share efficiently in multi-user/multi-tenant/multi-service setting, since they lack support for pre-empting and resuming jobs as is done on standard Central Processing Units (CPUs).

In the context of a cloud RAN, periodic radio data arrives at the CPU for processing. While processing the radio data, the CPU may hand over part of the processing task to the accelerator to speed up the processing. In order to conserve power, the CPU can be set to an idle mode or low power state when data is handed over to the accelerator for processing and the frequency of the clock signal supplied to the CPU will be reduced by the clock frequency governor upon detection of the idle state. When the accelerator completes its accelerated calculations and the CPU returns to an active state, the clock signal needs to be ramped up to normal speed. The ramping of the clock signal contributes to the processing delay and can cause the CPU to miss its deadline for processing the current task.

SUMMARY

The present disclosure relates generally to a CPU clock frequency governor for a CPU in a cloud RAN that uses network information to control the frequency of the clock signal supplied to the CPU to conserve power without contributing to increased latency. The CPU receives radio data for processing at the start of a period and the processing needs to be completed before the end of the period. The CPU offloads heavy computations involved in processing the 5G workloads to a GPU or other accelerator. In

order to conserve power, the CPU enters an idle state when the radio data is handed over to the GPU for processing and the frequency of the CPU clock signal is reduced. A first timer is set based on an estimated execution time of the GPU to ramp-up the CPU clock before the radio data is handed back to the CPU so that the CPU is ready to process the results immediately once the GPU finishes its processing. Furthermore, if the processing of the radio data by the CPU is finished before the deadline, the CPU enters the idle state and a second timer is set to ramp-up the CPU clock before the start of the next period. Ramping the CPU clock in advance of the time that the CPU returns to the active state enables the CPU to fully process the radio data before the next period.

A first aspect of the disclosure comprises methods of controlling a clock signal for CPU resources in a cloud RAN to conserve power. In one embodiment, the method comprises receiving information indicative of a transmission pattern of radio data to be processed. The method further comprises estimating an execution time for co-processing the radio data by an accelerator associated with the CPU. The method further comprises controlling a frequency of the clock signal provided to the CPU based on the transmission pattern of the radio data, the estimated execution time for co-processing the radio data, and a ramping time of the clock signal.

A second aspect of the disclosure comprises clock frequency governor for controlling a frequency of a clock signal for a CPU in a cloud RAN. The clock frequency governor is configured to receive information indicative of a transmission pattern of radio data to be processed. The clock frequency governor is further configured to estimate an execution time for co-processing the radio data by an accelerator associated with the CPU. The clock frequency governor is further configured to control the frequency of the clock signal provided to the CPU based on the transmission pattern of the radio data, the estimated execution time for co-processing the radio data, and a ramping time of the clock signal.

A third aspect of the disclosure comprises a computing system for a cloud RAN. The computing system network interface circuitry for communicating radio data with other network devices, a CPU and accelerator configured to process the radio data, and a clock frequency governor for controlling a frequency of a clock signal provided to the CPU. The clock frequency governor is configured to receive information indicative of a transmission pattern of radio data to be processed. The clock frequency governor is further configured to estimate an execution time for co-processing the radio data using

an accelerator associated with the CPU. The clock frequency governor is further configured to control the frequency of the clock signal provided to the CPU based on the transmission pattern of the radio data, the estimated execution time for co-processing the radio data, and a ramping time of the clock signal.

A fourth aspect of the disclosure comprises a computer program for clock frequency governor configured to control a frequency of a clock signal for a CPU in a cloud RAN. The computer program comprises executable instructions that, when executed by processing circuitry in the workload scheduler, causes the workload scheduler to perform the method according to the first aspect.

A fifth aspect of the disclosure comprises a carrier containing a computer program according to the fourth aspect. The carrier is one of an electronic signal, optical signal, radio signal, or a non-transitory computer readable storage medium.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates a cloud infrastructure model for a wireless communication network.

Figure 2 illustrates an exemplary cloud RAN architecture for a wireless communication network.

Figure 3 illustrates an arrangement of computational resources in a cloud RAN.

Figure 4 is a timeline illustrating one example of how processing is divided between a CPU and accelerator.

Figure 5 is a timeline illustrating operation of the CPU clock in a conventional system.

Figure 6 is a graph illustrating the relation between power (W) and core voltage (V).

Figure 7 is a timeline illustrating operation of the CPU clock to reduce latency according to an embodiment.

Figure 8 illustrates the computation of an estimated execution time for the accelerator to process radio data.

Figure 9 illustrates communications between the main functional components of the computing system to support network aware control of the CPU clock.

Figure 10 is exemplary pseudocode for the CPU clock frequency governor.

Figure 11 illustrates a method implemented by a network-aware CPU clock frequency governor.

Figure 12 illustrates an embodiment of a network-aware CPU clock frequency governor.

Figure 13 illustrates a computer system for a cloud RAN including a network aware CPU clock frequency governor.

DETAILED DESCRIPTION

The present disclosure relates generally to the control of a clock signal for a CPU providing computational resources for processing periodic radio data for a cloud RAN. The CPU receives radio data for processing at the start of a processing period and the processing needs to be completed before the end of the period. The CPU offloads heavy computations involved in processing the 5G workloads to a GPU or other accelerator. In order to conserve power, the CPU enters an idle state when the radio data is handed over to the GPU for processing and the frequency of the CPU clock signal is reduced. A first timer is set based on an estimated execution time of the GPU to ramp-up the CPU clock before the radio data is handed back to the CPU so that the CPU is ready to process the results immediately once the GPU finishes its processing. Furthermore, if the CPU completes the processing of the radio data before the deadline, the CPU enters the idle state and a second timer is set to ramp-up the CPU clock before the start of the next period. Ramping the CPU clock in advance of the time that the CPU returns to the active state reduces latency due to ramping of the CPU clock and enables the CPU to fully process the radio data before the next period.

Figure 1 illustrates an exemplary cloud computing model 10 for a communication network using virtualized NFs (VNFs) 25 and containerized NFs (CNFs) 35. Instead of proprietary hardware, the cloud computing model 10 uses industry standard servers, high volume storage and generic networking hardware that are organized into one or more data centers 15. The physical resources of the data centers 15 are shared via a virtualization layer (e.g., OpenStack or VMware) 20 that provides infrastructure as a service (IaaS) and/or a containerization layer (e.g., Kubernetes) 30 that provides containers as a service (CaaS). The containerization layer 30 may be deployed on bare metal (BM) or on a virtual machine (VM) provided by the virtualization layer 20. The various NFs in a communication network are implemented as software running on a VM provided by a virtualization layer 20, or in a container provided by a containerization layer 30. The cloud computing model 10 provides ubiquitous, on-demand access to cloud resources that enables rapid provisioning and configuration of VNFs 25 and CNFs 35.

A NFV Management and Orchestration (MANO) architecture provides a framework for the management and orchestration of the network resources including computing, networking, storage, virtual machine (VM) and container resources. NFV MANO includes three main components: a Virtualized Infrastructure Manager (VIM) 40, a NFV Orchestrator 45 and one or more Virtual NF Managers (VNFMs) 50. The VIM 40 is responsible for controlling, managing, and monitoring the compute, storage, and network hardware provided by the datacenter 15. The NFV Orchestrator 45 is responsible for resource coordination including the creation, allocation and termination of VMs and containers that are used by the VNFs/CNFs 25, 35. The VNFM 50 is responsible for the life period management of VNFs 25 and CNFs 35. VNFM 50 operations include instantiation of VNFs/CNFs 25, 35, scaling of VNFs/CNFs 25, 35, updating and/or upgrading VNFs/CNFs 25, 35 and termination of VNFs/CNFs 25, 35. A VNFM 50 can be assigned to either a single VNF/CNF 25, 35 instance or to multiple VNF/CNF 25, 35 instances. The VNFs/CNFs 25, 35 managed by a VNFM 50 can be all of the same type of NF or a mix of different types of NFs.

While network operators have focused primarily on virtualization of core network functions, such as the vIMS, vEPC and vMME, NFV offers a new alternative, known as cloud RAN, to conventional RAN architectures based on proprietary hardware. In a cloud RAN, certain RAN functions are moved into the cloud and implemented using generic, commercial off-the-shelf (COTS) hardware. The virtualized RAN functions are implemented as microservices in containers that can be deployed and managed according to cloud-native principles.

Figure 2 illustrates an exemplary cloud RAN architecture. Essentially, the BBU resources are virtualized, moved into the cloud and implemented using COTS servers. The virtual BBUs (vBBUs) are typically part of a BBU pool shared by multiple RUs. The centralization of BBU equipment in the cloud helps lower the total cost of hardware and is more easily scalable.

In the embodiment shown in Figure 2, the vBBU is further divided into a virtualized data unit (vDU) 120 and a virtualized central unit (vCU) 110. The vDU 120 is typically implemented in an edge network close to the RU while the vCU 110 can be centralized. The vDU 120 is connected to one or more RUs by a fronthaul network, such as an optical fiber network, and is responsible for most baseband processing tasks, some of which require accelerators to meet stringent latency requirements, for Ultra Reliable Low Latency Communications (URLLC). Exemplary processing tasks handled

by the vDU 120 include Layer 1 (L1) processing, Medium Access Control (MAC) and Radio Link Control (RLC) processing, and scheduling. The L1 processing includes Physical Downlink Control Channel (PDCCH) signal processing, Physical Downlink Shared Channel (PDSCH) signal processing, Physical Uplink Shared Channel (PUSCH) signal processing, Sounding Reference Signal (SRS) processing and beamforming weight (BFw) calculations. Processing of the PDSCH, PUSCH, SRS and BFw calculations typically requires accelerators to meet low latency requirements. These processes are highlighted in figure 2 by a box drawn with a bold line. The vCU 110 is located between the vDU 120 and core network in the processing pipeline and aggregates packet flows for many vDUs 120. The vCU 110 communicates with the vDU 120 over a midhaul network (F1) and with the core network over a backhaul network. The midhaul and backhaul networks may comprise IP networks.

One of the key cell configuration parameters in 5G RAN is numerology. The five numerologies currently defined in the Third Generation Partnership Project (3GPP) standards for 5G are shown in Table 1 below.

μ	N_{symb}^{slot}	$N_{slot}^{frame,\mu}$	$N_{slot}^{subframe,\mu}$
0	14	10	1
1	14	20	2
2	14	40	4
3	14	80	8
4	14	160	16

Table 1: 3GPP Numerologies

The choice of numerology for a RAN is generally driven by the wave propagation model of the deployment frequency. In a high band, for example, more phase noise is observed and therefore greater carrier separation is required (e.g., 120KHz or numerology 3). On the other hand, when targeting sub-6GHz deployment, a 30KHz sub-carrier separation (numerology 1) is better suited to the task. The numerology provides a sense of the largest possible cell radius and rough HARQ latency based on a 500µs Transmission Time Interval (TTI). These numerologies serve the purpose of configuring the baseband to meet given latency and QoS requirements. In 5G deployments, the numerology configuration is typically assigned statically upon deployment.

In the 5G network, radio data is transmitted in a periodic interval referred to as a transmission time interval. Transmit or receive data arrives periodically at the vBBU or

vDU and needs to be processed before the next data arrives. The length of the period for processing the data depends on the numerology.

Figure 3 illustrates an arrangement of computational resources for processing periodic data in a cloud RAN. The cloud RAN baseband processing is typically divided between two processing elements: the CPU and a GPU (or other accelerator) connected to the GPU that handles heavy computations. The incoming data is received by the CPU and copied to the GPU, which handles the heavy computations. The GPU execution is controlled by the CPU. When the GPU computations are completed, the result is copied back to main memory of the CPU.

Figure 4 is a timeline illustrating the division of the baseband processing between the CPU and GPU in a cloud RAN. The period, start time and deadline for periodic 5G workloads are shown on the timeline. For simplicity, the deadline corresponds to the end of the period. Those skilled In the art will appreciate that the deadline could occur before the end of the period and the principles and mechanisms herein described can be generalized and extended to those embodiments. As illustrated in Figure 4, the periodic workloads have hard deadlines. The CPU must finish processing the current task before the period is over, because a new period will start at the deadline and a new task for processing will be received. If the current task is not completed, it will delay the start of the second period, and thus eventually push all subsequent periods later and later in time in a cascading effect.

As shown in the timeline, the CPU enters an idle state when the data is handed over to the GPU for processing and the frequency of the CPU clock signal is reduced to conserve power. The CPU returns to the active state when the GPU finishes its processing. Furthermore, if the CPU completes processing of the radio data before the deadline, the CPU enters the idle state to wait for the start of the next processing period. For each transition from the idle state to the active state, the CPU clock is ramped up gradually until it reaches full speed. During this ramping period, data processing will be slower than normal.

Reducing the CPU clock frequency while the CPU is idle can result in significant savings in terms of power. The power used by the system is quadratically dependent on the voltage as shown in Figure 6. This nonlinear relation shows that voltage scaling can provide substantial benefits in terms of power saving and is an important feature in the design of energy efficient systems.

Figure 5 is another timeline showing the normal operation of the CPU clock during a processing period. Before the start of the period and after the CPU has transferred data to the GPU, the CPU is idle. The idle state triggers the standard CPU clock frequency governor to ramp down the CPU clock, i.e., reduce the frequency of the clock signal provided to the CPU, to save power. The clock frequency governor will then ramp up the CPU clock when the CPU returns to the active state. A conventional sample-based clock frequency governor detects and reacts to the change in state of the CPU. The overhead involved in ramping up the clock leads to a delay. In practice, the CPU processes data significantly slower while the clock signal is ramping compared to full speed, which results in some delay in processing the data. The processing delay is not only caused by the fact that the clock needs time to ramp up, but also by the length of the sampling period for detecting state changes. In many cases, the delay due to the ramping of the clock signal leads to a processing overrun where the deadline is missed and the CPU continues to process the current task even though it has received a new task for the subsequent period. These delays can have a cascading effect on subsequent periods, where greater delay is introduced in each successive period.

One aspect of the present disclosure is to reduce these processing delays by using knowledge of the workload characteristics and system information to ramp the CPU clock before the predicted end of the idle periods so that the CPU is operating at full speed immediately upon returning to the active state. In one embodiment, three pieces of information are used for controlling the frequency of the clock signal from the CPU clock: the length of the processing period allotted to the CPU for processing the data; the execution time needed by the GPU for performing the accelerated calculations, and the amount of time required to ramp up the CPU clock to full speed when the CPU transmission from the idle state to an active state Using these pieces of information, a CPU clock frequency governor proactively controls the frequency of the CPU clock signal so that the CPU is running at full speed at the start of the processing period, and when the GPU hands over results of its accelerated calculations.

The first factor, the length of the processing period, depends on the transmission (TX) pattern (e.g., periodicity) of the data. In 5G systems, user data transmitted on the PUSCH or PDSCH is typically transmitted in periodic intervals referred to as TTIs. The length of a TTI depends on the numerology, i.e., channel spacing, used for data transmissions. For a conventional 15kHz channel spacing, a TTI is equal to 1 ms, which comprises fourteen Orthogonal Frequency Division Multiplexing (OFDM) symbols. The

length of the TTI will vary, however, depending on the numerology. For higher numerologies, the length of the TTI will decrease and for lower numerologies the length of the TTI will increase. In embodiments of the present disclosure, knowledge of the numerology can thus be used to determine the length of a processing period, which in general will be the same as the length of the TTI.

In 5G, periodic data with equal length TTIs is the most common TX pattern. Those skilled in the art will appreciate, however, that the principles herein described can also be applied to more complex TX patterns. For example, the techniques herein described could be applied to semi-periodic TX patterns where the length of different data transmissions varies in a predictable or reoccurring manner. In general, the technique herein described can be applied to any predictable or reoccurring TX pattern.

The second factor, the GPU execution time, refers to the amount of time needed by the GPU to perform accelerated calculations and can be measured from the time the CPU hands over data to the GPU until the time that the GPU returns the results of the accelerated calculations to the CPU. It has been observed that a) the execution time of the GPU is proportional to the workload size, and b) the cellular workload will typically vary slowly (compared to the execution time). One import of these observations is that a simple moving average estimation can provide sufficiently accurate execution time estimates. As described in more detail below, measurements or observations of the execution times of the GPU over a large number of processing periods can also be described by a statistical distribution, which is continuously updated.

The third factor, the ramping time for the CPU clock, is the amount of time needed to ramp up the CPU clock from a low speed to full speed when the CPU returns to an active state. The CPU clock cannot switch instantaneously from low speed to full speed but needs some time to gradually ramp up. The ramping time depends on the bandwidth and response time of the phase-locked loop (PLL) controlling the clock signal. The ramping time is typically available as part of the system information.

In embodiments of the present disclosure, a CPU clock frequency governor uses the length of the processing period, the GPU execution time, and the ramping time of the CPU clock to proactively control the CPU clock in contrast to a generic sample-based governor that detects and reacts to state changes of the CPU. In a typical computer, the Advanced Configuration and Power Interface (ACPI) provides an open standard that operating systems can use to discover and configure computer hardware components, to perform power management, e.g., putting unused hardware components to sleep or to

perform status monitoring. The ACPI Specification defines the following four global "Gx" states and six sleep "Sx" states for an ACPI-compliant computer system are shown in Table 2 below:

Gx	Name	Sx	Description
G0	Working	S0	The computer is running and the CPU executes instructions. "Awaymode" is a subset of S0, where monitor is off but background tasks are running
G1	Sleeping	S0ix	Modern Standby, or "Low Power S0 Idle". Partial processor SoC sleep. Known to ARM and x86 devices.
		S1	*Power on Suspend (POS)*: Processor caches are flushed, and the CPU(s) stops executing instructions. The power to the CPU(s) and RAM is maintained. Devices that do not indicate they must remain on may be powered off
		S2	CPU powered off. Dirty cache is flushed to RAM
		S3	commonly referred to as *Standby, Sleep, or Suspend to RAM (STR)*: RAM remains powered
		S4	*Hibernation or Suspend to Disk*: All content of the main memory is saved to non-volatile memory such as a hard drive, and the system is powered down
G2	Soft Off	S5	G2/S5 is almost the same as G3 Mechanical Off, except that the power supply unit (PSU) still supplies power, at a minimum, to the power button to allow return to S0. A full reboot is required. No previous content is retained. Other components may remain powered so the computer can "wake" on input from the keyboard, clock, modem, LAN, or USB device
G3	Mechanical Off		The computer's power has been totally removed via a mechanical switch (as on the rear of a PSU). The power cord can be removed and the system is safe for disassembly (typically, only the real-time clock continues to run using its own small battery)

Table 2: Global States For ACPI

The CPU receives radio data for processing at the start of a processing period and the processing needs to be completed before the end of the period. The processing to be performed is referred to herein as a task and the data to be processed is referred to as a workload. The CPU offloads heavy computations involved in processing the 5G workload to a GPU or other accelerator (e.g., FPGA or ASIC). In order to conserve power, the CPU enters an idle state when the radio data is handed over to the GPU for processing and the CPU clock frequency governor reduces the frequency of the CPU clock signal to conserve power.

As previously noted, the estimated execution time of the GPU is constantly measured and updated. One simple and straightforward method for updating the estimated execution time is to use a simple moving average over a predetermined

number *n* of periods. Another method is to fit measurements over a large number of periods to a statistical distribution (e.g., ensuring that the cumulative distribution function (CDF) of latency-model matches the measured latency distribution, for example by using the Matlab dfittool). One example of a statistical distribution is illustrated in Figure 8, where the model estimator could be configured to use the p95-latency from the estimated latency distribution, e.g., the right-most bin. The characteristics of the workload may depend on the type of 5G workload, for example the speed at which it may vary. This information may optionally be supplied to the estimator to control how often new estimates should be calculated. There are several alternative methods to estimate the execution time, including techniques based on machine learning. The particular method used to estimate the execution time of the GPU is not material.

The CPU clock frequency governor uses the estimate of the GPU execution time to set a timer that triggers the start of the ramping period for ramping up the CPU clock before the GPU completes its computations. More particularly, a first timer referred to as Timer A is set equal to the estimated execution time of the GPU minus the ramping period for the CPU clock provided by system information. When Timer A expires, the clock frequency governor beings ramping the CPU clock so that when the GPU completes is calculations and hands the results back to the CPU, the CPU will be running at full speed. The proactive ramping of the CPU clock signal thus avoids processing delays attributable to the ramping of the CPU clock. After receiving processing results from the GPU, the CPU completes any remaining processing. If the CPU completes processing before the end of the current period, the CPU can be set to idle mode and a second timer referred to as Timer B is set to start ramping of the CPU clock before the start of the next period. The second timer can be computed by subtracting the ramping time from the time left in the current period.

Figure 9 illustrates the main logical functions of the CPU clock frequency governor 300. The CPU clock frequency governor 300 includes four main logical functions to perform process control, period determination, GPU execution time estimation, and frequency control. Some functions, such as the clock control, period determination, and execution time estimation could be performed by software modules executing on the CPU. Some functions, such as the frequency control, could be performed by a separate processor, firmware, or hardware circuit. In other embodiments, software executing on the CPU or other processor could perform all of the functions shown in Figure 9. In still other embodiments, the functions could be

distributed between software executed by the CPU and software executed by another processor. The precise details of the implementation are not material.

The CPU clock frequency governor receives the workload type and the numerology associated with the data transmission as inputs. The processing period determination function determines the processing period for processing the radio data based on the numerology and updates the frequency control function with information about the period length by calling setPeriodTime(). As one example, the period determination function determines the processing period for processing the radio data. In this example, the processing period for processing PDSCH/PUSCH is determined based on the numerology. In one embodiment, the TTI interval for PDSCH/PUSCH transmission is determined based on the numerology and the processing period is set equal to the time duration of the TTI, or some other function of the TTI. During execution of the task, the process control function calls sleepAndWakeUpLater() every time the GPU executes. During normal operation, the process control function will also report the measured execution time of the GPU for and the workload type for each task to the execution time estimation function. As previously described, the execution time estimation function continuously estimates the execution time of the GPU using a moving average or statistical distribution. The execution time estimation function continuously, periodically or responsive to some trigger updates the frequency control function with the relevant estimated execution times by calling SetExecutionTime(). Using the provided information, the frequency control function outputs a frequency control signal to the clock circuit or hardware platform to control the frequency of the CPU clock and the ramp-up times for the CPU clock in order to ensure that the CPU is ready to process the results once the GPU has completed its computations and also to ensure that the CPU is ready to start once the next period begins. The clock circuit or hardware platform may include an API that is exposed to the clock frequency governor and the frequency control command may be a part of the API.

For illustrative purposes, Figure 10 provides example pseudo-code for implementing the CPU clock frequency governor. When the CPU clock frequency governor 300 receives the SleepAndWakeUpLater() function call from the controller 350, it determines the call type. The call type can be either to wait for GPU execution or to wait for a new period to start. In the first case, the CPU clock frequency governor 300 sets a first timer equal to the estimated GPY execution time minus the ramp up time for the CPU clock signal. In the second case, the clock frequency governor 300 sets a

second timer equal to the time until the start of the new period minus the ramp up time of the clock signal. In either case, when timer expires, the CPU clock frequency governor 300 will start ramping the CPU clock signal.

Figure 11 illustrates an exemplary method 200 implemented by a CPU clock frequency governor 300 of controlling a clock signal for CPU resources in a cloud RAN to conserve power. A CPU clock frequency governor receives information indicative of a TX pattern of radio data to be processed (block 210). The CPU clock frequency governor further estimates an execution time for co-processing the radio data by an accelerator (e.g. GPU, FPGS, ASIC, etc.) associated with the CPU (block 220). The CPU clock frequency governor controls a frequency of the clock signal provided to the CPU based on the TX pattern of the radio data, the estimated execution time for co-processing the radio data, and a ramping time of the clock signal (block 230).

In some embodiments of the method 200, receiving information indicative of a TX pattern of the radio data to be processed comprises receiving information indicative of a periodicity of the radio data. In some embodiments, the information comprises a numerology used for the transmission of the radio data.

Some embodiments of the method 200 further comprise determining a processing period for processing the radio data based on information indicative of the TX pattern. In this context, period refers to the time interval for processing the data. In the case of PDSCH/PUSCH processing, the processing period may be periodic. In this case the processing period may be determined as a function of the TTI for sending PUSCH/PDSCH data. In other scenarios, such as processing BFw signals and SRS, the processing period may not be a periodic time interval.

In some embodiments of the method 200, estimating an execution time for co-processing the radio data comprises estimating the execution time based on historical data. In some embodiments of the method 200, estimating the execution time based on historical data comprises measuring the execution time for co-processing the radio data over a plurality of processing periods to obtain measurement data, and estimating the execution time based on the measurement data.

In one embodiment, the execution time is estimated based on the measurement data comprises computing a running average over a predetermined number of the processing periods.

In some embodiments of the method 200, estimating the execution time based on the measurement data comprises generating a statistical distribution of the execution

time over the plurality of the processing periods, and computing an estimated execution time based on the statistical distribution.

In some embodiments of the method 200, controlling the frequency of the clock signal comprises determining a start time of a processing period for processing the radio data based on the received information, and increasing the frequency of the clock signal during a first ramping period before the start time of the processing period, wherein the first ramping period is determined based on the ramping time of the clock signal.

Some embodiments of the method 200, further comprise switching the CPU to an idle state and decreasing the frequency of the clock signal while radio data is co-processed by the accelerator; switching the CPU to an active state when the co-processing is completed based on the estimated execution time, and increasing a frequency of the clock signal during a second ramping period prior to the return of the CPU to the active state, wherein the second ramping period is determined based on the ramping time.

Some embodiments of the method 200, further comprise completing, by the CPU, processing of the radio data in the active state, and switching the CPU to the idle state when the processing of the radio data is completed.

An apparatus can perform any of the methods herein described by implementing any functional means, modules, units, or circuitry. In one embodiment, for example, the apparatuses comprise respective circuits or circuitry configured to perform the steps shown in the method figures. The circuits or circuitry in this regard may comprise circuits dedicated to performing certain functional processing and/or one or more microprocessors in conjunction with memory. For instance, the circuitry may include one or more microprocessor or microcontrollers, as well as other digital hardware, which may include Digital Signal Processors (DSPs), special-purpose digital logic, and the like. The processing circuitry may be configured to execute program code stored in memory, which may include one or several types of memory such as read-only memory (ROM), random-access memory, cache memory, flash memory devices, optical storage devices, etc. Program code stored in memory may include program instructions for executing one or more telecommunications and/or data communications protocols as well as instructions for carrying out one or more of the techniques described herein, in several embodiments. In embodiments that employ memory, the memory stores program code that, when executed by the one or more processors, carries out the techniques described herein.

Figure 12 illustrates an exemplary clock frequency governor 300 according to an embodiment. The clock frequency governor 300 comprises a receiving unit 310, a estimating unit 320 and a frequency control unit 330. The various units 310-330 can be implemented by hardware and/or by software code that is executed by one or more processors or processing circuits. The receiving unit 310 is configured receive information indicative of a TX pattern of radio data to be processed. The estimating unit 320 is configured to estimate an execution time for co-processing the radio data by an accelerator associated with the CPU. The frequency control unit 330 is configured to control the frequency of the clock signal provided to the CPU based on the TX pattern of the radio data, the estimated execution time for co-processing the radio data, and a ramping time of the clock signal.

Figure 13 illustrates a computing system 400 in a cloud RAN for processing radio data transmitted or received by a cell in a wireless communication network. The computing system 400 generally comprises network interface circuitry (NIC) 410 for communicating radio data with other network nodes/devices, a CPU 420 and accelerator 430 configured to process the radio data, and a clock frequency governor 300 to control the frequency of a clock signal supplied to the CPU 410, and memory 450.

The NIC 410 connects the computing system 400 to a communication network for communication with the radio unit (RU) and other network nodes/devices. The NIC 410 may comprise a wired or wireless interface operating according to any standard, such as the Ethernet, Wireless Fidelity (WiFi) and Synchronous Optical Networking (SONET) standards.

The CPU 420 may comprise a single core or multi-core processor, such as a reduced instruction set computing (RISC) processor or complex instruction set computing (CISC) processor. The accelerator 420 may comprise a graphics processing unit (GPU), Field Programmable Gate Array (FPGA), Application Specific Integrated Circuit (ASIC), hardware accelerator, or combination thereof.

The clock frequency governor 300 may comprise a separate microchip or microprocessor that generates the clock signal and regulates the timing of the CPU 420 and/or accelerator 430. Alternatively, the clock frequency governor 300 may be implemented by a control unit in the CPU. As another alternative, the various functions of the clock frequency governor can be distributed between the control unit in the CPU and other processing circuitry. Regardless of the specific details of the implementation, the clock frequency governor 300 is configured to perform the method 200 of Figure 11.

Memory 450 comprises both volatile and non-volatile memory for storing computer program code and data needed by the computing system 400 for operation. Memory 450 may store computer programs executed by the CPU 420, accelerator 430 and clock circuit 440. Memory 450 may comprise any tangible, non-transitory computer-readable storage medium for storing data including electronic, magnetic, optical, electromagnetic, or semiconductor data storage. In one embodiment, memory 450 stores a computer program 460 comprising executable instructions that configures the CPU clock frequency governor 300 to implement the method 200 according to Figure 11. A computer program 460 in this regard may comprise one or more code modules corresponding to the means or units described above. In general, computer program instructions and configuration information are stored in a non-volatile memory, such as a ROM, erasable programmable read only memory (EPROM) or flash memory. Temporary data generated during operation may be stored in a volatile memory, such as a random access memory (RAM). In some embodiments, computer program 460 may be stored in a removable memory, such as a portable compact disc, portable digital video disc, or other removable media. The computer program 460 may also be embodied in a carrier such as an electronic signal, optical signal, radio signal, or computer readable storage medium.

Those skilled in the art will also appreciate that embodiments herein further include corresponding computer programs. A computer program comprises instructions which, when executed on at least one processor of an apparatus, cause the apparatus to carry out any of the respective processing described above. A computer program in this regard may comprise one or more code modules corresponding to the means or units described above.

Embodiments further include a carrier containing such a computer program. This carrier may comprise one of an electronic signal, optical signal, radio signal, or computer readable storage medium.

In this regard, embodiments herein also include a computer program product stored on a non-transitory computer readable (storage or recording) medium and comprising instructions that, when executed by a processor of an apparatus, cause the apparatus to perform as described above.

Embodiments further include a computer program product comprising program code portions for performing the steps of any of the embodiments herein when the

computer program product is executed by a computing device. This computer program product may be stored on a computer readable recording medium.

CLAIMS

1. A method (200) of controlling a clock signal for a central processing unit (CPU) of a computer in a cloud infrastructure for a radio access network (RAN), the method comprising:

receiving (210) information indicative of a transmission pattern of radio data to be processed;

estimating (220) an execution time for co-processing the radio data by an accelerator associated with the CPU; and

controlling (230) a frequency of the clock signal provided to the CPU based on the transmission pattern of the radio data, the estimated execution time for co-processing the radio data, and a ramping time of the clock signal.

2. The method (200) of claim 1, wherein receiving information indicative of a transmission pattern of the radio data to be processed comprises receiving information indicative of a periodicity of the radio data.

3. The method (200) of claim 2, wherein the information comprises a numerology used for the transmission of the radio data.

4. The method (200) of any one of claims 1 - 3, further comprising determining a processing period for processing the radio data based on information indicative of the transmission pattern.

5. The method (200) of any one of claims 1 - 4, wherein estimating an execution time for co-processing the radio data comprises estimating the execution time based on historical data.

6. The method (200) of claim 5, wherein estimating the execution time based on historical data comprises:

measuring the execution time for co-processing the radio data over a plurality of processing periods to obtain measurement data; and

estimating the execution time based on the measurement data.

7. The method (200) of claim 6, wherein estimating the execution time based on the measurement data comprises computing a running average over a predetermined number of the processing periods.

8. The method (200) of claim 6, wherein estimating the execution time based on the measurement data comprises:

generating a statistical distribution of the execution time over the plurality of the processing periods; and

computing an estimated execution time based on the statistical distribution.

9. The method (200) of any one of claims 1 - 8, wherein controlling the frequency of the clock signal comprises:

determining a start time of a processing period for processing the radio data based on the received information; and

increasing the frequency of the clock signal during a first ramping period before the start time of the processing period, wherein the first ramping period is determined based on the ramping time of the clock signal.

10. The method (200) of any one of claims 1 - 9, further comprising:

switching the CPU to an idle state and decreasing the frequency of the clock signal while radio data is co-processed by the accelerator;

switching the CPU to an active state when the co-processing is completed based on the estimated execution time; and

increasing a frequency of the clock signal during a second ramping period prior to the return of the CPU to the active state, wherein the second ramping period is determined based on the ramping time.

11. The method of claim 10, further comprising:

completing, by the CPU, processing of the radio data in the active state; and

switching the CPU to the idle state when the processing of the radio data is completed.

12. A clock frequency governor (300) for controlling a frequency of a clock signal for a central processing unit (CPU) in a cloud radio access network (RAN), the clock frequency governor being configured to:

receive information indicative of a transmission pattern of radio data to be processed;

estimate an execution time for co-processing the radio data by an accelerator associated with the CPU; and

control the frequency of the clock signal provided to the CPU based on the transmission pattern of the radio data, the estimated execution time for co-processing the radio data, and a ramping time of the clock signal.

13. The clock frequency governor (300) of claim 12, further configured to perform the method of any one of claim 2 - 11.

14. A computing system (400) for a cloud radio access network, the computing device comprising:

network interface circuitry (405) for communicating radio data with other network devices;

a central processing unit (CPU) (410) and accelerator (420) configured to process the radio data;

a clock frequency governor (300) for controlling a frequency of a clock signal provided to the CPU, the clock frequency governor being configured to:

receive information indicative of a transmission pattern of radio data to be processed;

estimate an execution time for co-processing the radio data by an accelerator associated with the CPU; and

control the frequency of the clock signal provided to the CPU based on the transmission pattern of the radio data, the estimated execution time for co-processing the radio data, and a ramping time of the clock signal.

15. The computing system of claim 14, wherein the clock frequency governor (300) is further configured to perform the method of any one of claim 2 - 11.

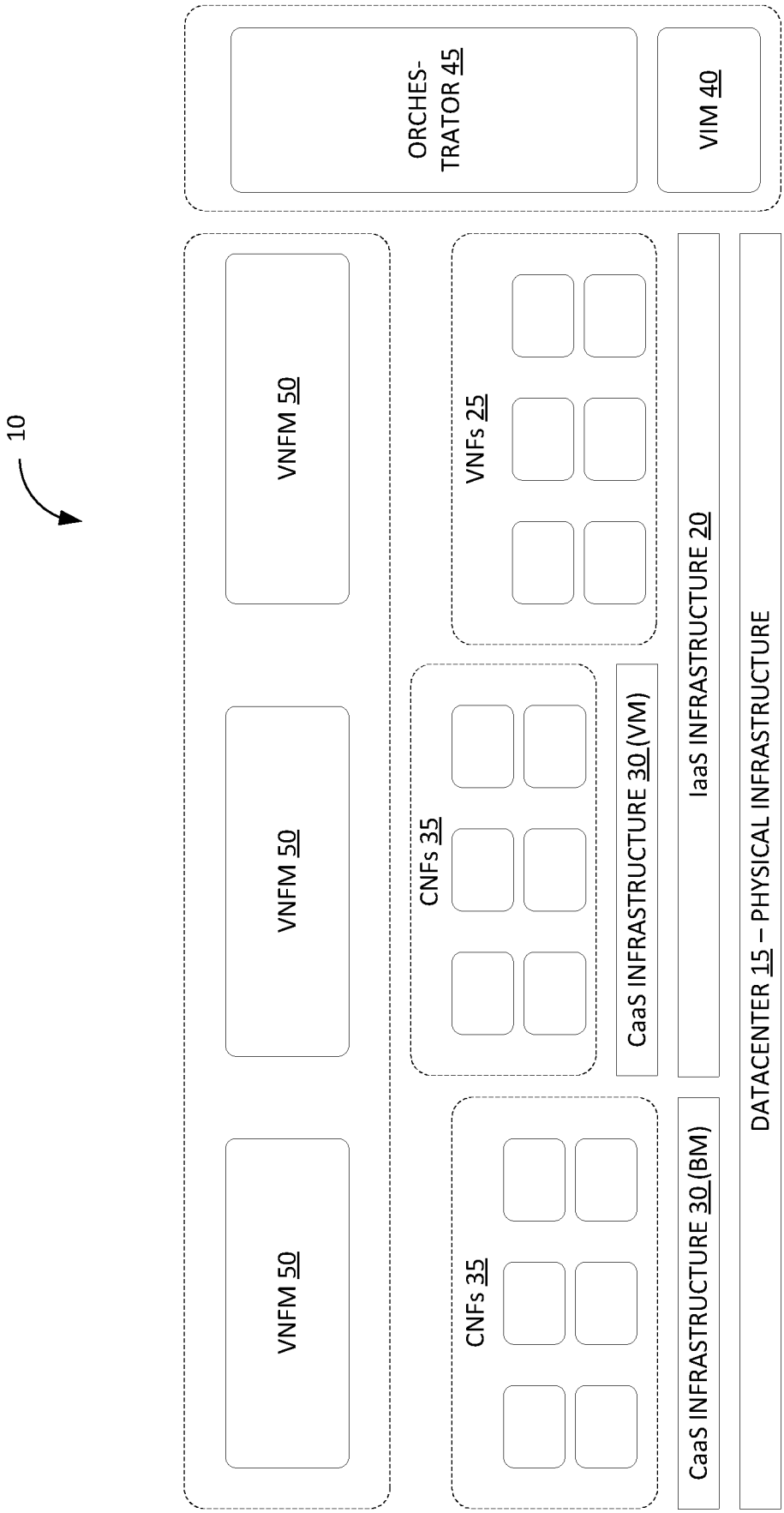
10
VNFM 50
VNFM 50
VNFM 50
ORCHES-
TRATOR 45
VIM 40
VNFs 25
CNFs 35
CaaS INFRASTRUCTURE 30 (VM)
IaaS INFRASTRUCTURE 20
CNFs 35
CaaS INFRASTRUCTURE 30 (BM)
DATACENTER 15 – PHYSICAL INFRASTRUCTURE

FIG. 1

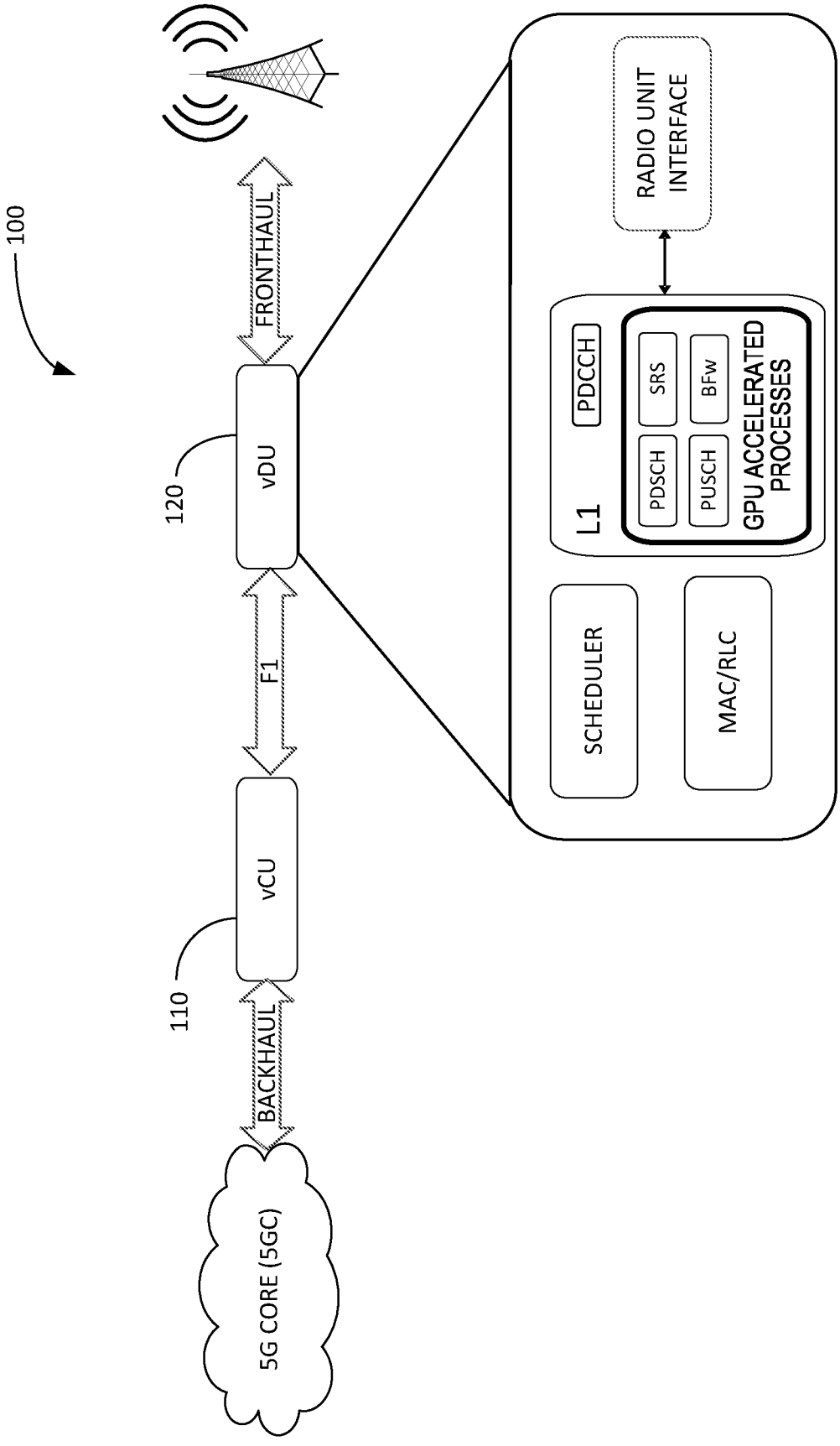
100
110
120
vCU
vDU
F1
BACKHAUL
FRONTHAUL
5G CORE (5GC)
SCHEDULER
MAC/RLC
L1
PDCCH
PDSCH
PUSCH
SRS
BFw
GPU ACCELERATED PROCESSES
RADIO UNIT INTERFACE

FIG. 2

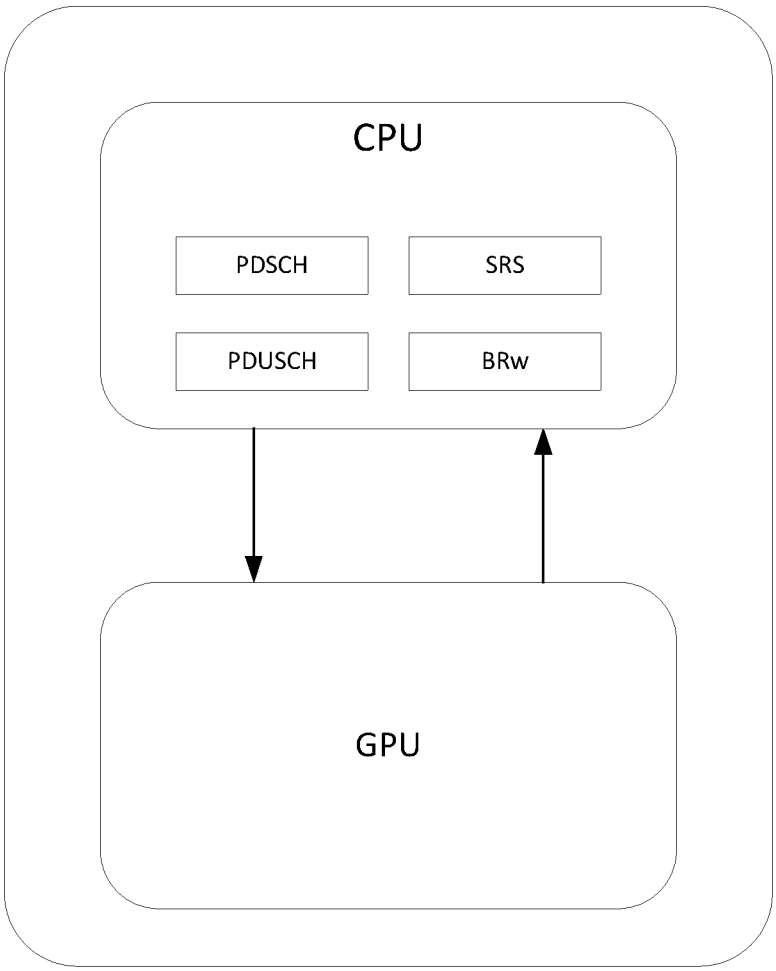

FIG. 3

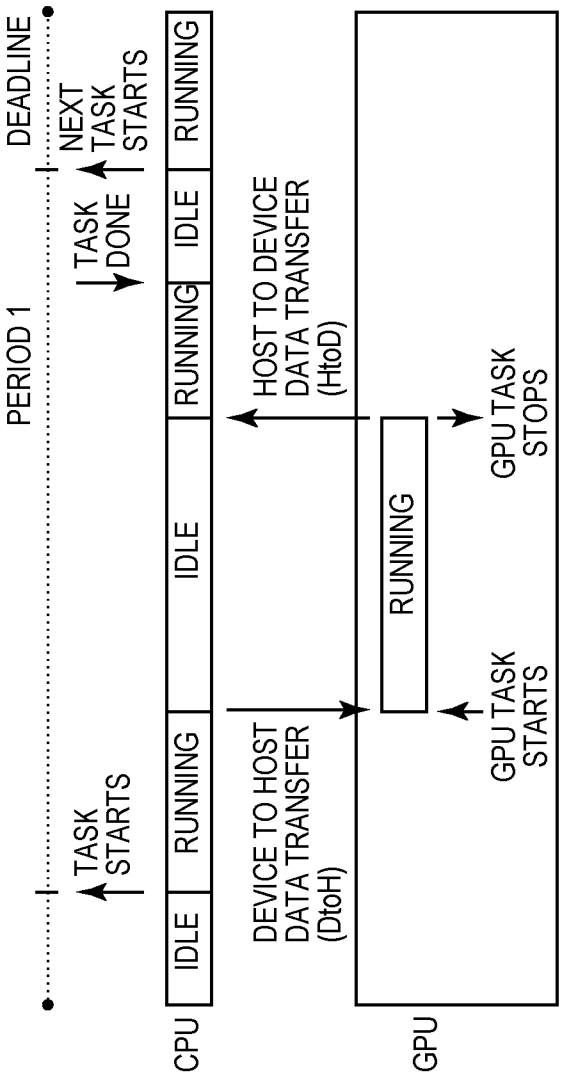
PERIOD 1
DEADLINE
TASK STARTS
TASK DONE
NEXT TASK STARTS
CPU
IDLE
RUNNING
IDLE
RUNNING
IDLE
RUNNING
DEVICE TO HOST DATA TRANSFER (DtoH)
HOST TO DEVICE DATA TRANSFER (HtoD)
GPU
GPU TASK STARTS
GPU TASK STOPS

FIG. 4

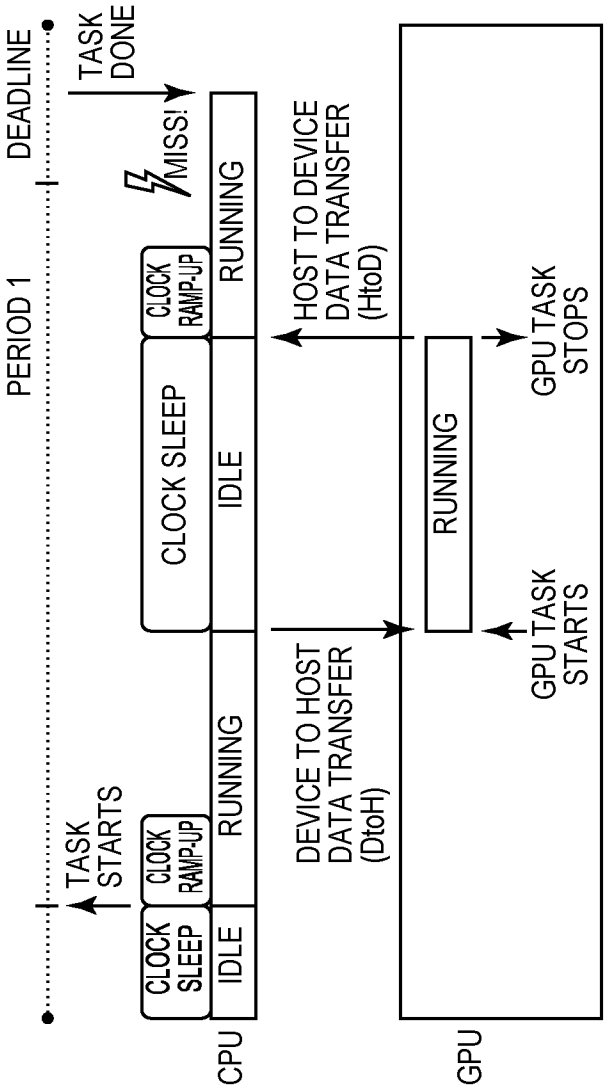
PERIOD 1
DEADLINE
TASK DONE
MISS!
TASK STARTS
CLOCK SLEEP
CLOCK RAMP-UP
IDLE
RUNNING
CLOCK SLEEP
CLOCK RAMP-UP
RUNNING
HOST TO DEVICE DATA TRANSFER (HtoD)
DEVICE TO HOST DATA TRANSFER (DtoH)
GPU TASK STARTS
GPU TASK STOPS
CPU
GPU

FIG. 5

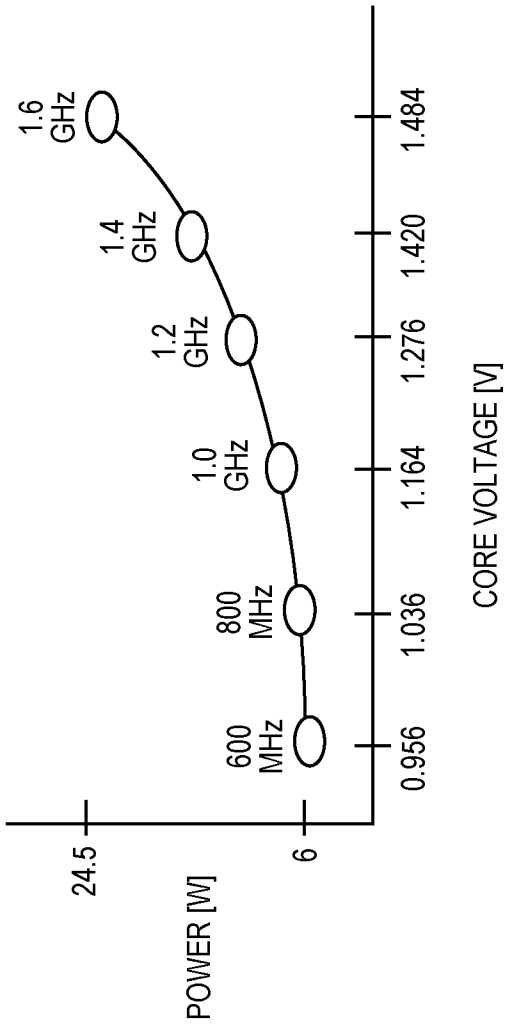
1.6 GHz
1.4 GHz
1.2 GHz
1.0 GHz
800 MHz
600 MHz
0.956
1.036
1.164
1.276
1.420
1.484
CORE VOLTAGE [V]
6
24.5
POWER [W]

FIG. 6

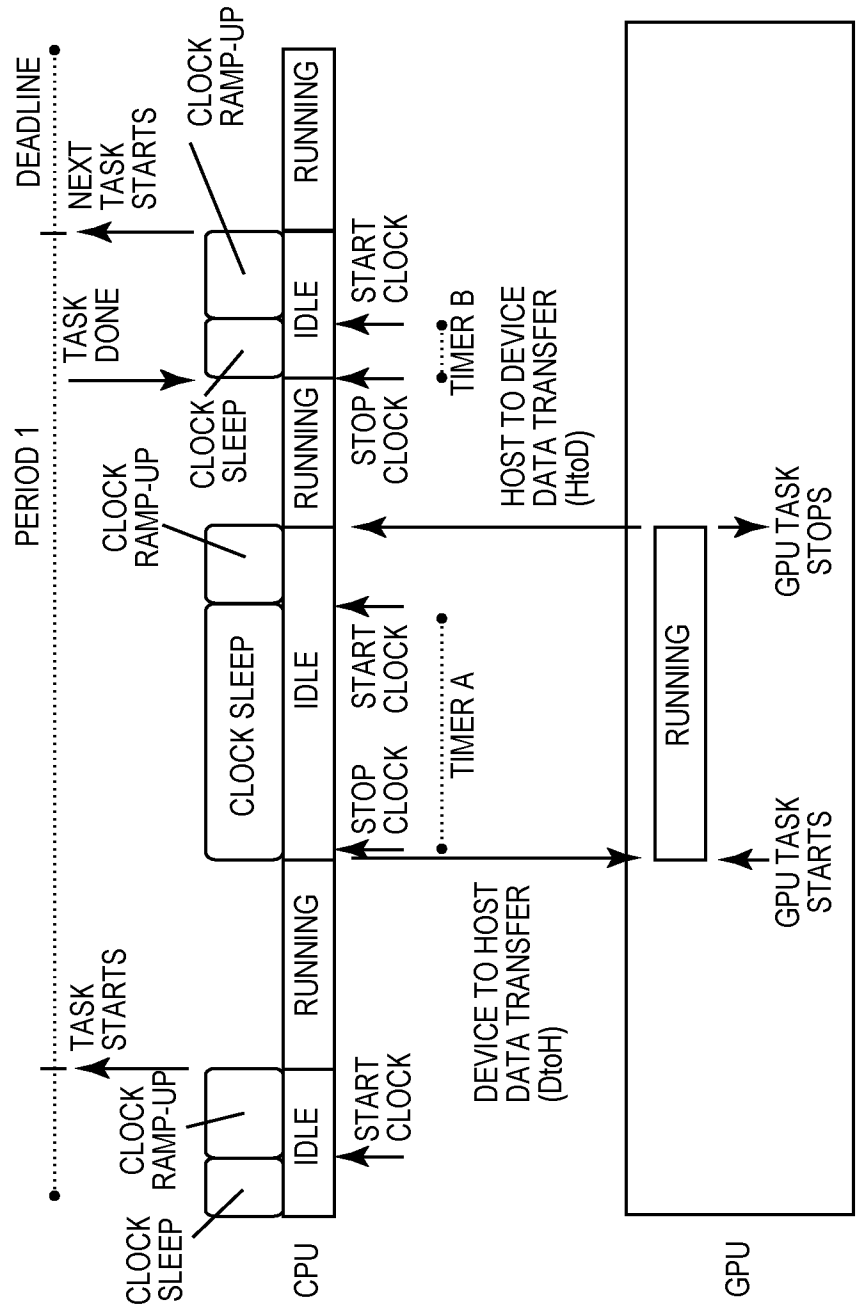
CPU
GPU
CLOCK SLEEP
CLOCK RAMP-UP
IDLE
RUNNING
TASK STARTS
PERIOD 1
DEADLINE
CLOCK SLEEP
CLOCK RAMP-UP
IDLE
RUNNING
TASK DONE
NEXT TASK STARTS
START CLOCK
STOP CLOCK
TIMER A
DEVICE TO HOST DATA TRANSFER (DtoH)
GPU TASK STARTS
RUNNING
GPU TASK STOPS
CLOCK SLEEP
CLOCK RAMP-UP
IDLE
RUNNING
START CLOCK
STOP CLOCK
TIMER B
HOST TO DEVICE DATA TRANSFER (HtoD)

FIG. 7

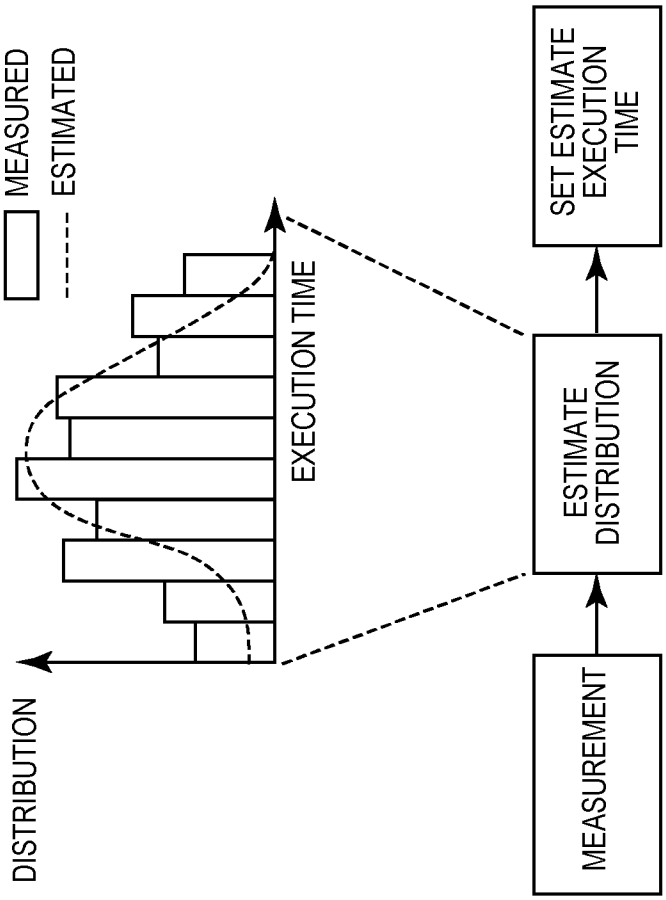
MEASURED
ESTIMATED
DISTRIBUTION
EXECUTION TIME
MEASUREMENT
ESTIMATE DISTRIBUTION
SET ESTIMATE EXECUTION TIME

FIG. 8

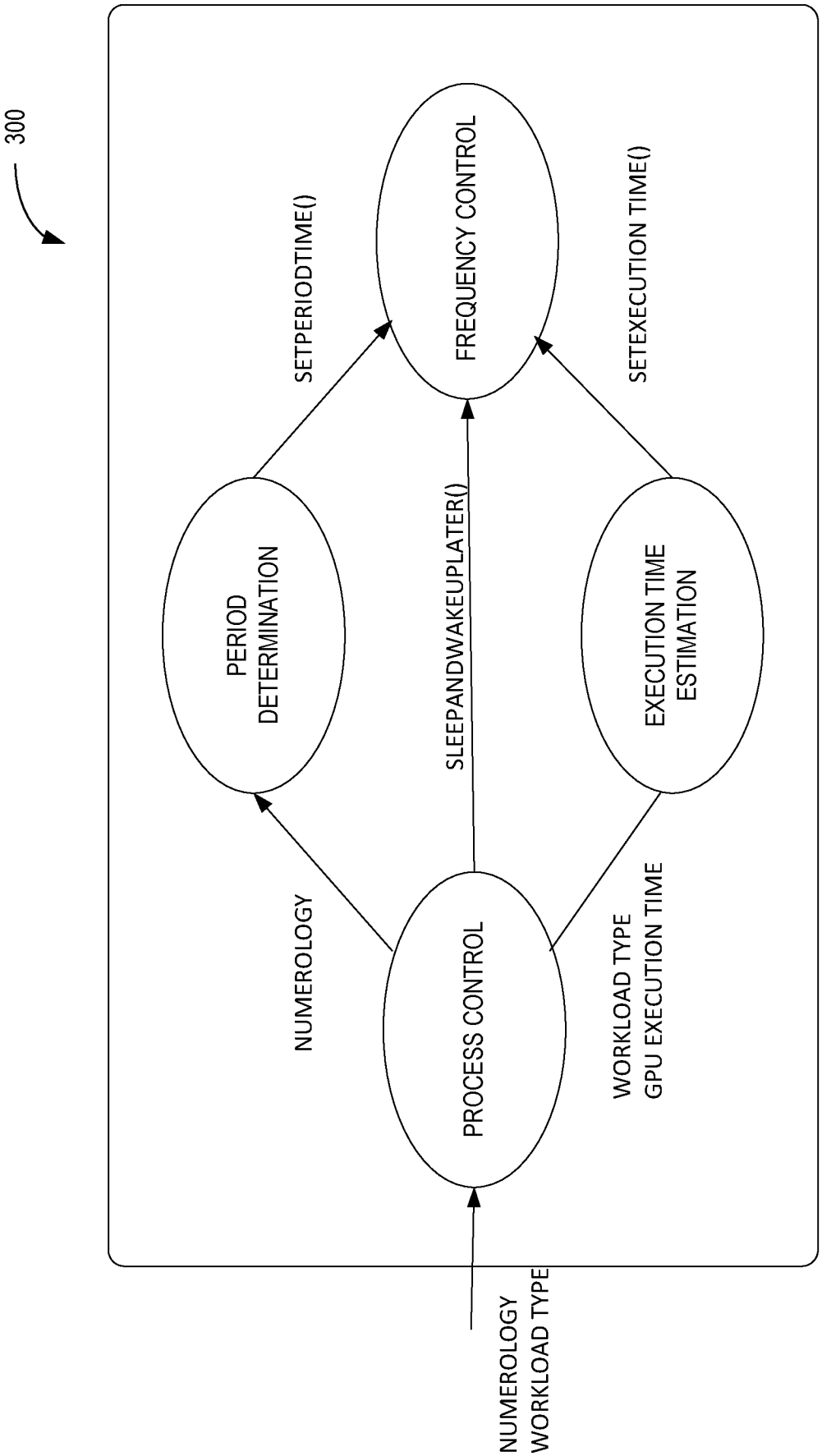
300
NUMEROLOGY
WORKLOAD TYPE
PROCESS CONTROL
NUMEROLOGY
PERIOD DETERMINATION
SETPERIODTIME()
WORKLOAD TYPE
GPU EXECUTION TIME
EXECUTION TIME ESTIMATION
SETEXECUTION TIME()
SLEEPANDWAKEUPLATER()
FREQUENCY CONTROL

FIG. 9

```
/* Pseudo code inside the CPU Frequency Governor */
void sleepAndWakeUp(string type){
  double timer = 0;

  if (type == waitForGPU) {  // Wait for the GPU
    timer = execTime – rampUpTime;
  } else if (type == waitForNewPeriod){ // Wait for the new period to start
    timer = tNewPeriod – rampUpTime;
  }

  // if there is time to ramp-up the clock,
  // set a timer for when to do so go to sleep.
  if (timer > 0){
    startClockRampUp(tNow + timer);
    sleep();
  }
}
```

FIG. 10

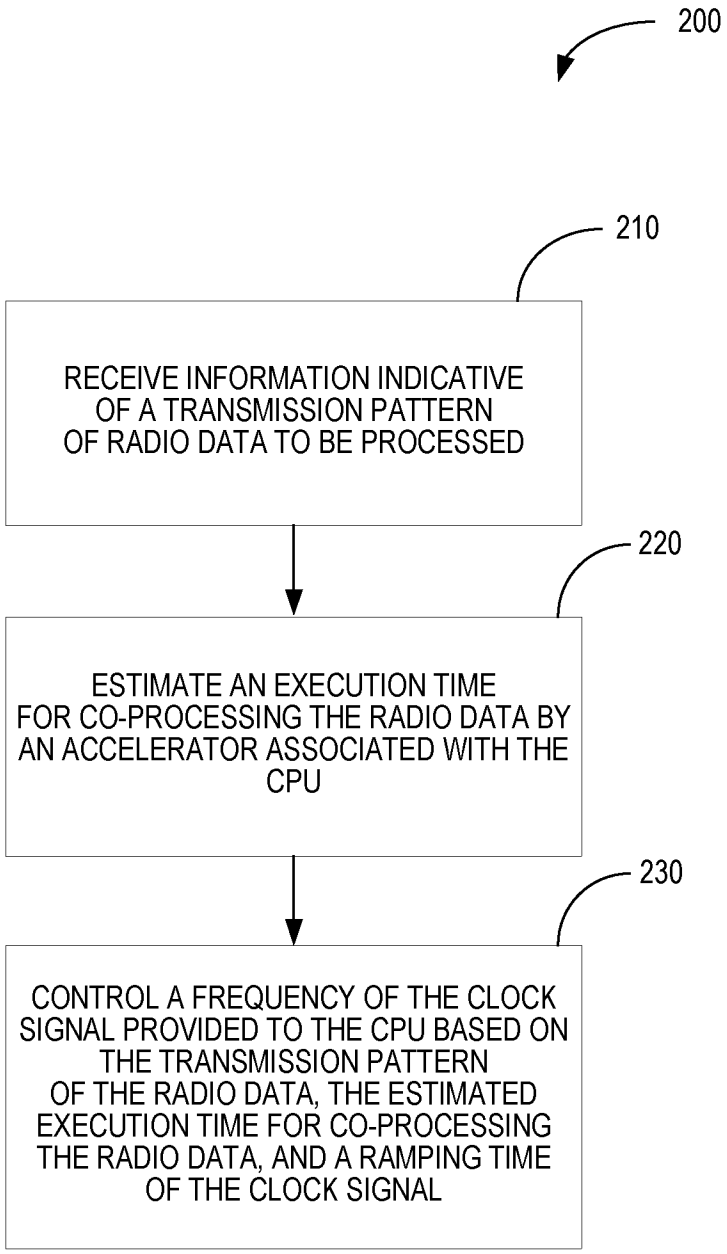
200
210
RECEIVE INFORMATION INDICATIVE OF A TRANSMISSION PATTERN OF RADIO DATA TO BE PROCESSED
220
ESTIMATE AN EXECUTION TIME FOR CO-PROCESSING THE RADIO DATA BY AN ACCELERATOR ASSOCIATED WITH THE CPU
230
CONTROL A FREQUENCY OF THE CLOCK SIGNAL PROVIDED TO THE CPU BASED ON THE TRANSMISSION PATTERN OF THE RADIO DATA, THE ESTIMATED EXECUTION TIME FOR CO-PROCESSING THE RADIO DATA, AND A RAMPING TIME OF THE CLOCK SIGNAL

FIG. 11

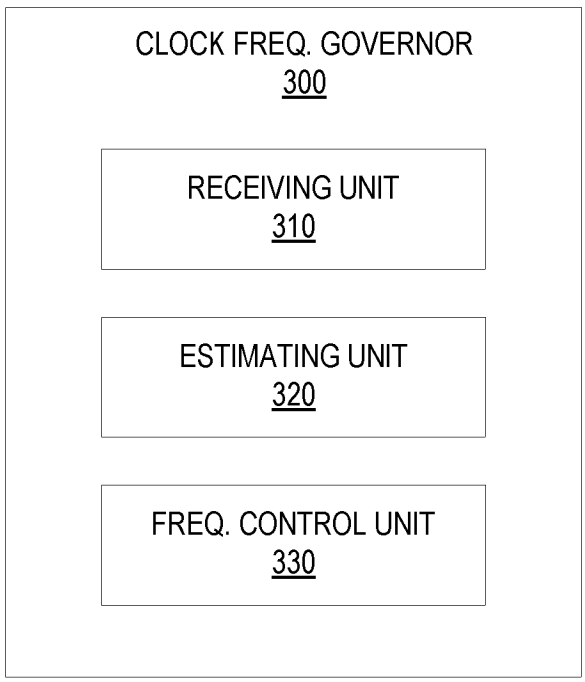
CLOCK FREQ. GOVERNOR
300
RECEIVING UNIT
310
ESTIMATING UNIT
320
FREQ. CONTROL UNIT
330

FIG. 12

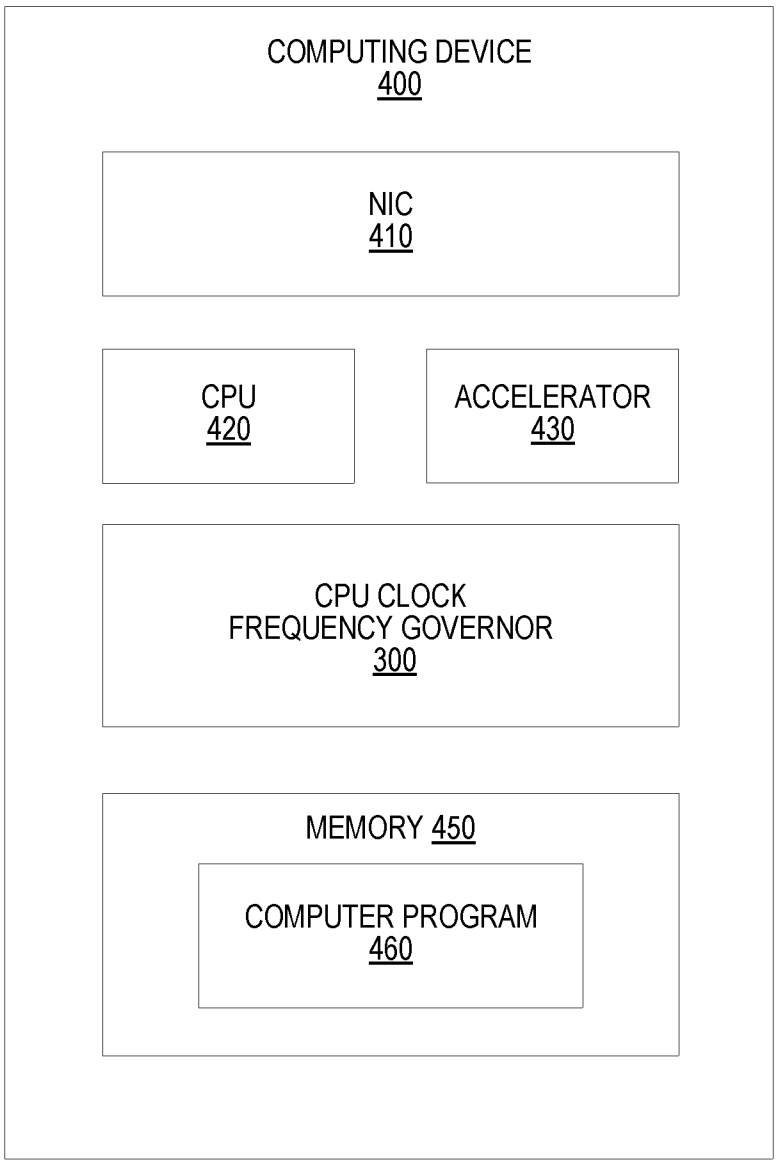
COMPUTING DEVICE
400
NIC
410
CPU
420
ACCELERATOR
430
CPU CLOCK
FREQUENCY GOVERNOR
300
MEMORY 450
COMPUTER PROGRAM
460

FIG. 13

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SE2022/050214

A. CLASSIFICATION OF SUBJECT MATTER

IPC: see extra sheet

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: G06F, H04W

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

SE, DK, FI, NO classes as above

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

EPO-Internal, PAJ, WPI data, COMPENDEX, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	WO 2022026044 A1 (MICROSOFT TECHNOLOGY LICENSING LLC), 3 February 2022 (2022-02-03); abstract --	1-15
A	US 20190272002 A1 (SEENAPPA VIKRAM ET AL), 5 September 2019 (2019-09-05); abstract --	1-15
A	US 20200201797 A1 (VU CHUONG ET AL), 25 June 2020 (2020-06-25); abstract --	1-15
A	US 20210184795 A1 (IBARS CASAS CHRISTIAN ET AL), 17 June 2021 (2021-06-17); abstract --	1-15

☒ 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

"D" document cited by the applicant in the international application

"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	Date of mailing of the international search report
10-11-2022	10-11-2022
Name and mailing address of the ISA/SE Patent- och registreringsverket Box 5055 S-102 42 STOCKHOLM Facsimile No. + 46 8 666 02 86	Authorized officer Lars Magnusson Telephone No. + 46 8 782 28 00

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SE2022/050214

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 8621253 B1 (BROWN JEFF ET AL), 31 December 2013 (2013-12-31); abstract --	1-15
A	US 20210365288 A1 (PARK HEE JUN), 25 November 2021 (2021-11-25); abstract --	1-15
A	US 11157311 B2 (GUIM BERNAT FRANCESC ET AL), 26 October 2021 (2021-10-26); abstract -- --------	1-15

International application No.
PCT/SE2022/050214

Continuation of: second sheet

International Patent Classification (IPC)

G06F 1/324 (2019.01)
G06F 1/3206 (2019.01)
G06F 1/3234 (2019.01)
H04W 52/00 (2009.01)

WO	2022026044	A1	03/02/2022	US	20220035665	A1	03/02/2022
US	20190272002	A1	05/09/2019	US	10579093	B2	03/03/2020
				US	20200159281	A1	21/05/2020
				US	11119529	B2	14/09/2021
US	20200201797	A1	25/06/2020	US	10713196	B1	14/07/2020
US	20210184795	A1	17/06/2021	AU	2020404901	A1	11/08/2022
				CN	114731163	A	08/07/2022
				DE	112020006125	T5	03/11/2022
				GB	2603867	A	17/08/2022
				KR	20220084327	A	21/06/2022
				WO	2021126718	A1	24/06/2021
US	8621253	B1	31/12/2013	NONE			
US	20210365288	A1	25/11/2021	TW	202147071	A	16/12/2021
				WO	2021236262	A1	25/11/2021
US	11157311	B2	26/10/2021	CN	111953725	A	17/11/2020
				EP	3734452	A1	04/11/2020
				US	20220138003	A1	05/05/2022
				US	20220318064	A1	06/10/2022
				US	11436051	B2	06/09/2022
				US	11416295	B2	16/08/2022
				US	11334382	B2	17/05/2022
				US	20200026575	A1	23/01/2020
				US	20190394096	A1	26/12/2019
				US	20190391971	A1	26/12/2019
				US	20190391855	A1	26/12/2019