



- (51) **International Patent Classification:**
G06F 15/163 (2006.01) *G06F 9/46* (2006.01)
- (21) **International Application Number:**
PCT/US2015/023174
- (22) **International Filing Date:**
27 March 2015 (27.03.2015)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (71) **Applicant:** HEWLETT PACKARD ENTERPRISE DEVELOPMENT LP [US/US]; 11445 Compaq Center Drive West, Houston, TX 77070 (US).
- (72) **Inventor:** KEAVENY, Thomas A; 8000 Foothills Blvd., Roseville, California 95747 (US).
- (74) **Agents:** KOGAN, Daniel et al.; Hewlett Packard Enterprise, 3404 E. Harmony Road, Mail Stop 79, Fort Collins, CO 80528 (US).
- (81) **Designated States** (*unless otherwise indicated, for every kind of national protection available*): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM,

DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, 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.

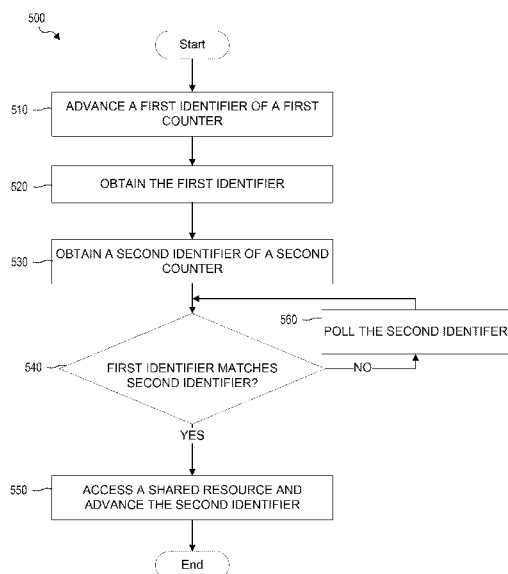
- (84) **Designated States** (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, 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).

Declarations under Rule 4.17:

— *as to the identity of the inventor (Rule 4.17(i))*

Published:

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

(54) Title: ACCESSING A SHARED RESOURCE**FIG. 5**

(57) **Abstract:** Examples disclosed herein relate, in one aspect, to a processor. In one aspect, the processor may advance a first counter coupled to the processor to a first value, obtain a second value of a second counter coupled to the processor, and determine whether the first value corresponds to the second value. The processor may also, if the first value does not correspond to the second value, wait until the first value corresponds to the second value, and when the first value corresponds to the second value, access a resource, and advance the second counter.

ACCESSING A SHARED RESOURCE

BACKGROUND

[0001] Many computing systems today allow multiple processes (or threads) to run concurrently and access shared resources, such as common memory areas, input/output resources, and the like. Concurrent execution of multiple processes may increase the system's performance, but it may also cause undesirable effects. For example, if one process modifies a memory area that is being read by another process, a phenomenon called data tearing may occur. To prevent this, a computing system may employ various synchronization mechanisms. For example, before accessing a shared resource, each of the concurrently running processes may "take" or "lock" a synchronization object (e.g., a mutual-exclusion lock (mutex) or a binary semaphore) associated with that resource. After completing the access, the process may "release" or "unlock" the synchronization object. The computing system may prevent a process from locking a synchronization object that has already been locked by another process, and thus prevent the process from accessing the shared resource until the other process finishes its use of the shared resource and unlocks the synchronization object.

BRIEF DESCRIPTION OF THE DRAWINGS

- [0002] The following detailed description references the drawings, wherein:
- [0003] FIG. 1 is a block diagram of an example electronic device;
- [0004] FIG. 2 is an example time diagram;
- [0005] FIG. 3 is another example time diagram;
- [0006] FIG. 4 is yet another example time diagram;
- [0007] FIG. 5 is a flowchart of an example method; and
- [0008] FIG. 6 is another block diagram of an example electronic device.

DETAILED DESCRIPTION

[0009] Examples disclosed herein describe, among other things, an electronic device. The electronic device may include, among other things, a processor. The processor may, in some examples, advance a first counter coupled to the processor to a first value, obtain a second value of a second counter coupled to the processor, and determine whether the first value corresponds to the second value. The processor may also, if the first value does not correspond to the second value, wait until the first value corresponds to the second value, and when the first value corresponds to the second value, access a resource, and advance the second counter.

[0010] FIG. 1 is a block diagram of an example electronic device 100. Electronic device 100 may include any type of electronic devices or combinations thereof. For example, electronic device 100 may be a server, a desktop computer, a notebook computer, a tablet, a mobile phone, a smartphone, a gaming device, a printing device, and the like.

[0011] In some examples, electronic device 100 may include a plurality of (e.g., at least two) processors 130. Each processor 130 may include one or more processors, such as central processing units (CPUs), semiconductor-based microprocessors, hardware state machines, graphics processing units (GPUs), field-programmable gate arrays (FPGAs), or other electronic circuitry, which may be integrated in a single device or distributed across devices. Each processor 130 may also be implemented as a combination of hardware and programming, as will be further discussed below. Any two processors 130 may be similar or different in terms of their type, function, processing power, and other characteristics. In some examples, some or all processors 130 may be running and processing data simultaneously. In some examples, some or all processors 130 may be different cores of the same multi-core processor. In some examples, instead of a plurality of processors 130, electronic device 100 may include a single processor (e.g., one single-core processor) running multiple threads, e.g., in a time-sharing manner.

[0012] Electronic device 100 may also include a memory 140. Memory 140 may include any combination of volatile and non-volatile memory, such as a random-access memory (RAM), flash memory, hard drive, and so forth. In some examples, memory 140 may include a non-transitory machine-readable storage medium.

[0013] Electronic device 100 may also include a first counter 110 and a second counter 120. First counter 110 and second counter 120 may each be implemented as any combination of hardware (e.g., digital circuits) and programming (e.g., software or firmware). For example, each counter may include a number of flip-flops connected in cascade. Each counter may hold a value and each counter may be able to advance the value either forward, backward, or both. Accordingly, “advancing” the counter as used in the examples described herein may refer to incrementing the counter or decrementing the counter, depending on the counter’s type and/or the system configuration.

[0014] In some examples, any processor 130 may advance a value associated with either of the counters 110 and 120 by issuing one or more signal or commands to the respective counter. For example, processor 130 may issue a signal or command causing a counter (e.g., 110 or 120) to advance (e.g., increment or decrement) by a certain step. Depending on the counter’s type and configuration, in some examples, processor 130 may also issue a signal or command to reset the counter to a predefined value (e.g., 0, 0xFF, etc.). Also, depending on the counter’s type and configuration, processor 130 may or may not be able to issue a signal or command to set the timer to an arbitrary value. In some examples, when a counter is advanced beyond its limit (e.g., incremented above its maximum value or decremented below its minimal value) the timer may automatically wrap around. For example, if a counter having a minimum value of 0, a maximum value of 255, and a current value of 253, is advanced by +5, the counter may wrap around and set its value to 2.

[0015] In some examples, each counter, in response to receiving a signal or command from processor 130 to advance by a certain step, may send back to

processor 130 a signal or command indicating the new (advanced) value of the counter. In some examples, processor 130 may also issue a command to receive the counter's current value without advancing the value. In some examples, in order to obtain the current value of a counter without advancing it, processor 130 may issue a signal or command to advance the counter by 0. In some examples, issuing a signal or command to advance a counter and receiving the new value from the counter may be performed atomically, such that no other processor may change the value of the counter between these events.

[0016] In some examples, the value of first counter 110 may represent a first identifier and the value of second counter 120 may represent a second identifier. In some examples, each counter may use its entire value (i.e., all of its bits) to represent its respective identifier. In other examples, the counter may include two or more fields, where one of the fields may represent the identifier, and one or more other fields may represent additional information, such as a status of the shared resource, a descriptor of the processor using the shared resource, or any other information associated with the shared resource, one or more processors 130, or any arbitrary information. Thus, while in some examples the value of the identifier may be equal to the value of the counter, in other examples, the value of the identifier may be determined based on the value of the identifier by applying some mathematical and/or logical operations (e.g., a masking "AND" operation, a shifting operation, and the like).

[0017] In some examples, the value of the first identifier (represented by counter 110) may be associated with the last processor 130 that requested the shared resource. In some examples, the second identifier (represented by counter 120) may be associated with or indicate the processor 130 that is currently allowed to access the shared resource. As discussed above, the shared resource may include any resource that may be simultaneously accessed (e.g., read from, written to, used, etc.) by at least two different processors 130. For example, the shared

resource may include an area in a memory (e.g., an area in memory 140) that is accessible to at least two processors 130.

[0018] In some examples, each processor 130, or at least two processors 130, before accessing a shared resource, may advance first counter 110 such that the first identifier represented by first counter 110 is advanced by a predefined step. For example, if the predefined step is +1 and the first identifier is represented by one or more least significant bits of first counter 110, processor 130 may advance (e.g., increase) the first identifier by 1 by incrementing first counter 110 by 1. As another example, if the predefined step is -1 and the first identifier is represented by bits 8:2 of first counter 110 (where bits 1:0 represent some other value(s)), processor 130 may decrease the first identifier by 1 (without affecting the other value(s)) by decrementing first counter 110 by 4.

[0019] In some examples, after advancing the first counter 110, processor 130 may obtain the value of the advanced first counter 110 and the value of the advanced first identifier represented by that value. As discussed above, in some examples, processor 130 may obtain the value of the advanced first counter 110 in response to the advancement signal or command. In other examples, processor 130 may obtain the value of the advanced first counter 110 by issuing another signal or command, such as a command to advance first counter 110 by 0. As discussed above, in some examples, processor 130 may advance first counter 110 and obtain the value of first counter 110 atomically. After obtaining the value of first counter 110, processor 130 may perform any necessary mathematical and/or logical operations to extract from it the value of the first identifier. In some examples, processor 130 may then temporarily store the value of the first identifier in a volatile or non-volatile memory such as memory 140 or a local register.

[0020] In some examples, after obtaining the first identifier, processor 130 may obtain the second identifier without advancing second counter 120. For example, processor 130 may issue a command to advance second counter 120 by 0, in response receive the current value of second counter 120, and if necessary,

perform mathematical and/or logical operations to extract the value of the second identifier from the value of second counter 120.

[0021] In some examples, after obtaining the values of the first and second identifiers, processor 130 may compare the values and determine whether or not the second identifier matches the first identifier, i.e., whether the two identifiers are equal. If processor 130 determines that the second identifier matches the first identifier, processor 130 may access the shared resource. However, if processor 130 determines that the second identifier does not match the first identifier, processor 130 may not access the shared resource. Instead, in some examples, processor 130 may wait until the second identifier matches the first identifier. For example, processor 130 may poll the second identifier by periodically (e.g., at given intervals of time or at random times) obtaining the value of second counter 120, until the second identifier represented by second counter 120 matches the first identifier. Thus, in some examples, processor 130 may proceed to accessing the shared resource safely (e.g., without other processors 130 being able to access it at the same time) only when the second identifier represented by second counter 120 matches the first identifier represented by first counter 110.

[0022] After determining that the second identifier matches the first identifier, processor 130 may advance second counter 120 such that the second identifier is advanced by a predefined step (e.g., +1) which, in some examples, may be the same predefined step as the predefined step that was used to advance first counter 110. As illustrated in the examples discussed herein, advancing the second identifier after accessing the shared resource may allow another processor 130 to access the shared resource.

[0023] FIG. 2 shows an example time diagram involving three processors 130 (referred to as processors A, B, and C) interacting with first counter 110 (referred to as a “requestor counter”), second counter 120 (referred to as a “service counter”), and a shared resource in parallel. In FIG. 2, the first identifier value as determined

by each processor 130 is indicated as “My ID,” and the second identifier value as determined by each processor 130 is indicated as “Service ID.”

[0024] As illustrated in FIG. 2, in some examples, first counter 110 and second counter 120 may be initialized (e.g., when electronic device 100 is powered up) to different values. In some examples, one value may be higher than the other value by the above-discussed predefined step (e.g., 1). For example, if the predefined step is +1, second counter 120 may be initialized to a value that is higher by 1 than that of first counter 110, and if the predefined step is -1, second counter 120 may be initialized to a value that is lower by 1 than the value of first counter 110. In the example of FIG. 2, the predefined step is +1, and counters 110 and 120 are initialized to 0 and 1, respectively.

[0025] In the example of FIG. 2, the entire value of first counter 110 and 120 represents the first identifier and the second identifier, respectively, meaning that no additional mathematical or logical operations may be needed to extract the identifiers based on their respective counters' values. In the example of FIG. 2, processor A, before accessing a shared resource, advances (205) first counter 110 by +1, and thereby also advances the first identifier by +1. In response, processor A obtains the advanced value (0x01) of first counter 110, determines the first identifier to be equal to 1, and (optionally) temporarily stores this value as its “My ID” value.

[0026] Next, processor B also decides to access the shared resource. Accordingly, processor B advances (210) counter 110 by +1, receives in response the advanced value (0x02) of first counter 110, determines the value of the first identifier to be 2, and (optionally) temporarily stores this value as processor B's “My ID.”

[0027] Next, processor A attempts to determine whether it can now safely access the shared resource by determining the value of the second identifier. Accordingly, processor A advances (215) second counter 120 by 0, receives in response the value (0x01) of second counter 120, and determines the value of the second identifier to be 1. Processor A then determines whether or not the second

identifier matches the first identifier that was last obtained by processor A and temporarily stored as its “My ID.” In this case, processor A determines that the second identifier matches the first identifier, and based on this determination, processor A determines that it may safely access the shared resource, and proceeds to access (220) the shared resource.

[0028] Next, while processor A is accessing the shared resource, processor B determines whether it can safely access the shared resource. Accordingly, processor B advances (225) second counter 120 by 0, obtains the value (0x01) of second counter 120, and based on that value determines the value of the second identifier to be 1. Processor B then compares the second identifier to its “My ID” value and determines that the two identifier do not match. Accordingly, processor B determines that it may not yet safely access the shared resource.

[0029] Next, processor C also decides to access the shared resource. Accordingly, processor C advances (230) first counter 110 by +1 and obtains its “My ID” value of 3. Next, processor A determines that it no longer needs the shared resource. Accordingly, processor A advances (235) second counter 120 by +1, thereby also advancing the second identifier represented by second counter 120 by +1.

[0030] Next, processor B, having previously determined that its “My ID” did not match the second identifier, polls the second identifier again by advancing (240) second counter 120 by 0 and receiving in response the value (0x02) of second counter 120. Processor B then determines that the second identifier now matches its “My ID” value. Accordingly, processor B determines that it can safely access the shared resource, and proceeds to access (245) the shared resource.

[0031] Next, processor C also determines whether it can safely access the shared resource. Processor C obtains the value of the second identifier by advancing (250) second counter 120, determines that the second identifier does not match its “My ID” value, and therefore does not access the shared resource at this time.

[0032] Next, processor B releases the shared resource by advancing (255) second counter 120 (and the second identifier represented thereby) by +1. Next, processor C checks the value of the second identifier by advancing (260) second counter 120 by 0, determines that the second identifier matches its “My ID” value, and proceeds to safely access (265) the shared resource.

[0033] Next, processor A decides to access the shared resource again, and therefore obtains a new “My ID” value by advancing (270) first counter 110 by +1. It is appreciated that the processors may continue “locking,” accessing, and “releasing” the shared resource in a similar manner indefinitely. As discussed above, while each processor 130 may advance any of the two counters at any time, in some examples, one processor may not advance a given counter after another processor has advanced that and before the other processor has obtained the advanced counter value. In other words, in some examples, advancing a counter and obtaining the counter’s new value may include one or more operations performed atomically.

[0034] FIG. 3 illustrates another example, in which the most significant bits of counters 110 and 120 represent the first and second identifiers, respectively, and the least significant bit represents a status of the shared resource. For example, the least significant bit may be set to 1 (“busy”) when the shared resource is being accessed by one of the processors, and 0 (“free”) when the resource is not being used or accessed by any of the processors.

[0035] In this example, in order to advance the first identifier or the second identifier by +1 when the resource is free, processor 130 may advance (305, 310, 335) the respective counter by +2, and in order to advance the first identifier or the second identifier by +1 when the resource is busy, the processor may advance (340) the respective counter by +1. Additionally, in this example, when a processor obtains a value of one of the counters, in order to obtain the value of the identifier represented by the counter’s value, processor 130 may shift the value to the right by one bit (i.e., divide the value by two). Additionally, in this example, after a

processor determines that its “My ID” value matches the value of the second identifier, meaning that it can safely access the shared resource, the processor may advance (320, 350) second counter 120 by +1 in order to set its status bit to 1 (“busy”) without affecting the value of the second identifier. Additionally, in this example, when processor 130 finishes accessing the shared resource, it may advance (340, 365) second counter 120 by +1, thereby both advancing the second identifier by +1 and resetting the status bit to 0 (“free”).

[0036] FIG. 4 illustrates another example, in which each counter includes three fields: an identifier field (bits 8 and above), a status field (bit 7), and a descriptor field (bits 6:0). The descriptor field may identify a processor that currently accesses (“owns”) the shared resource. The descriptor may be, for example, a descriptive name, a serial number, a bus address, or any other descriptor identifying (e.g., uniquely) a particular processor 130 from the plurality of processors 130. In the example of FIG. 4, the descriptor is a seven-bit ASCII code, and processors A, B, and C are described by ASCII codes “A”, “B”, and “C”, respectively.

[0037] In FIG. 4, processor A advances (405) first counter 110 by +0x100, thereby advancing the first identifier by 1 and keeping the other fields unchanged. In response, processor A obtains the advanced value (0x100) of first counter 110 and based on that value determines the value of the first identifier. In this example, to determine the value of the first identifier, processor A shifts the value of first counter 110 to the right by eight bits. Processor A then determines the second identifier by advancing (410) second counter 120 by 0, and then determines that the second identifier matches the first identifier, which means that processor A can access the shared resource safely. Based on this determination, processor A advances second counter 120 by a sum of the ASCII value of the character “A” and the number 0x80. This causes the status bit of second counter 120 to be set to 1 (“busy”), the descriptor field to be set to “A”, and the second identifier to remain unchanged. Processor A then accesses the shared resource. After finishing using the shared resource, processor A releases it by advancing second counter 120 by

63, which is the value that cause the status field and the descriptor field to be set to 0, and the second identifier to be advanced by +1. In some examples, when processor A releases the shared resource, processor may advance second counter 120 by a value that will cause the status field to 0 ("free") but will keep the descriptor field unchanged, so that it keeps describing the last processor to have access to the shared resource. For reasons of brevity, FIG. 4 only illustrates processor A's interactions with the counters and the shared resource, but it is appreciated that processors B and C may also try to simultaneously access the shared resource, as illustrated in the above examples.

[0038] Based on the above examples, it is appreciated that while counter 110 may include additional fields such as status or descriptor, in some examples, these fields may not be utilized. In some examples, for computational efficiency, the first and the second identifier may occupy the same bit range (e.g., 15:8) in both counters 110 and 120, but the least significant bits (e.g., 7:0) of first counter 110 may not be used and may be left at their default value (e.g., 0). In some examples, if a counter (e.g., second counter 120) does represent additional information (e.g., status, descriptor, etc.), the additional information may be stored in the counter's least significant bit(s), and the identifier may be stored in the most significant bits. In such examples, the counter may automatically wrap around when it reaches its limit (minimum or maximum) without disrupting the methods discussed herein.

[0039] While some examples discussed herein referred to multiple processors running in parallel, it is appreciated that the systems and methods discussed herein may also be applicable to a single processor (e.g., one single-core processor) that executes a plurality of threads in a time-sharing manner.

[0040] As mentioned above, in some examples, one or more processors 130 may be implemented as any combination of hardware and programming. The programming may include processor-executable instructions stored on a tangible, non-transitory computer readable medium and the hardware may include a processing resource for executing those instructions. The processing resource, for

example, may include one or multiple processors (e.g., central processing units (CPUs), semiconductor-based microprocessors, graphics processing units (GPUs), field-programmable gate arrays (FPGAs) configured to retrieve and execute instructions, or other electronic circuitry), which may be integrated in a single device or distributed across devices. The computer readable medium can be said to store program instructions that when executed by the processor resource implement the functionality of the respective component. The computer readable medium may be integrated in the same device as the processor resource or it may be separate but accessible to that device and the processor resource. In one example, the program instructions can be part of an installation package that when installed can be executed by the processor resource to implement the corresponding component. In this case, the computer readable medium may be a portable medium such as a CD, DVD, or flash drive or a memory maintained by a server from which the installation package can be downloaded and installed. In another example, the program instructions may be part of an application or applications already installed, and the computer readable medium may include integrated memory such as a hard drive, solid state drive, or the like.

[0041] FIG. 5 is a flowchart of an example method 500 for changing a frequency of a clock signal. Method 500 may be described below as being executed or performed by a computing device a processor (e.g., processor 130), or another suitable device or module. Method 500 may be implemented in the form of executable instructions stored on at least one non-transitory machine-readable storage medium of the computing device and executed by at least one processor of the client device. Alternatively or in addition, method 500 may be implemented in the form of electronic circuitry (e.g., hardware). In alternate examples of the present disclosure, one or more or blocks of method 500 may be executed substantially concurrently or in a different order than shown in FIG. 5. In alternate examples of the present disclosure, method 500 may include more or less blocks than are shown in

FIG. 5. In some examples, one or more of the blocks of method 500 may, at certain times, be ongoing and/or may repeat.

[0042] At block 510, method 500 may, before accessing a shared resource, advance, by a first processor (e.g., one of processors 130 from the plurality of processors of device 100), a first identifier represented by a first counter (e.g., 110). At block 520, the method may obtain the first identifier (after it has been advanced) by the first processor. At block 530, the method may obtain, by the first processor, a second identifier represented by a second counter (e.g., 120). As discussed above, this may include “advancing” the second counter by a zero step (e.g., issuing a “+0” command). At block 540, the method may compare the first identifier and the second identifier to determine if they match. If they match, the method may proceed to block 550, where the method may access the shared resource by the first processor, and advance the second identifier by the first processor. If the two identifier do not match, however, the method may proceed to block 560, where the method may poll, by the first processor. The method may then proceed to block 540 again.

[0043] As discussed above, in some examples, the second counter may include, among other things, a first field representing the second identifier and a second field indicating whether the shared resource is busy. In these examples, method 500 may also, before accessing the shared resource, set the second field to a value indicating that the shared resource is busy, and after accessing the shared resource, set the second field to a value indicating that the shared resource is free.

[0044] As discussed above, in some examples, the second counter may include, among other things, a first field representing the second identifier and a second field identifying a processor that currently has access to the shared resource (or that was last to have access to the shared resource). In these examples, method 500 may also, before accessing the shared resource, set the second field to a value identifying the first processor.

[0045] FIG. 6 is a block diagram of an example electronic device 600. Electronic device 600 may be similar to electronic device 100 of FIG. 1. In the example of FIG. 6, client device 600 includes a processor 610 and a non-transitory machine-readable storage medium 620. Although the following descriptions refer to a single processor and a single machine-readable storage medium, it is appreciated that multiple processors and multiple machine-readable storage mediums may be anticipated in other examples. In such other examples, the instructions may be distributed (e.g., stored) across multiple machine-readable storage mediums and the instructions may be distributed (e.g., executed by) across multiple processors.

[0046] Processor 610 may be similar to processor 130 of FIG. 1, and may include one or more central processing units (CPUs), microprocessors, and/or other hardware devices suitable for retrieval and execution of instructions stored in non-transitory machine-readable storage medium 620. In the particular example shown in FIG. 6, processor 610 may fetch, decode, and execute instructions 622, 624, 626, 628, or any other instructions (not shown for brevity). As an alternative or in addition to retrieving and executing instructions, processor 610 may include one or more electronic circuits comprising a number of electronic components for performing the functionality of one or more of the instructions in machine-readable storage medium 620. With respect to the executable instruction representations (e.g., boxes) described and shown herein, it should be understood that part or all of the executable instructions and/or electronic circuits included within one box may, in alternate examples, be included in a different box shown in the figures or in a different box not shown.

[0047] Non-transitory machine-readable storage medium 620 may be similar to memory 140, and may include any electronic, magnetic, optical, or other physical storage device that stores executable instructions. Thus, medium 620 may be, for example, Random Access Memory (RAM), an Electrically-Erasable Programmable Read-Only Memory (EEPROM), a storage drive, an optical disc, and the like. Medium 620 may be disposed within computing device 600, as shown in FIG. 5. In

this situation, the executable instructions may be “installed” on computing device 500. Alternatively, medium 520 may be a portable, external or remote storage medium, for example, that allows client device 510 to download the instructions from the portable/external/remote storage medium. In this situation, the executable instructions may be part of an “installation package”. As described herein, medium 520 may be encoded with executable instructions for generating report(s).

[0048] Referring to FIG. 6, instructions 622, when executed by a processor (e.g., 610), may cause the processor to advance a first counter coupled to the processor to a first value. Instructions 624, when executed by a processor, may cause the processor to obtain a second value of a second counter coupled to the processor. Instructions 626, when executed by a processor, may cause the processor to determine whether the first value corresponds to the second value and if the first value does not correspond to the second value, wait until the first value corresponds to the second value. As discussed above, in some examples, the first counter’s value may represent at least a first identifier and the second counter’s value may represent at least a second identifier. In these examples, determining whether the first value corresponds to the second value may include determining whether the first identifier is equal to the second identifier. As discussed above, in some examples, the second counter’s value may include a first field including the second identifier and a second field including information associated with the shared resource, where the first field may include a most significant bit of the second counter. Instructions 628, when executed by a processor, may cause the processor to, when the first value corresponds to the second value, access a resource, and advance the second counter.

CLAIMS

1. An electronic device comprising:
a plurality of processors;
a first counter, the first counter's value representing at least a first identifier;
and
a second counter, the second counter's value representing at least a second identifier,
wherein at least a first processor from the plurality of processors is to:
advance the first counter such that the first identifier is advanced by a predefined step,
obtain the second identifier,
determine whether the second identifier matches the first identifier,
based on a determination that the second identifier matches the first identifier, access the shared resource, and advance the second counter such that the second identifier is advanced by the predefined step.
2. The electronic device of claim 1, wherein the first processor is further to:
based on a determination that the second identifier does not match the first identifier, poll the second identifier until the second identifier matches the first identifier.
3. The electronic device of claim 1, wherein:
the first counter and the second counter are to automatically wrap around upon reaching their respective limits.
4. The electronic device of claim 1, wherein the electronic device further comprises a memory, and wherein accessing the shared resource comprising

accessing an area of the memory that is accessible to the first processor and to at least a second processor from the plurality of processors.

5. The electronic device of claim 1, wherein the second counter's value further represents a status of the shared resource.

6. The electronic device of claim 6, wherein the first processor is further to:
after the determination that the second identifier matches the first identifier, advance the second counter such that the status indicates that the shared resource is busy, and
after accessing the shared resource, advance the second counter such that the status indicates that the shared resource is not busy.

7. The electronic device of claim 1, wherein the second counter's value further represents a descriptor of a processor, from the plurality of processors, that has access to the shared resource.

8. The electronic device of claim 7, wherein the first processor is further to:
after the determination that the second identifier matches the first identifier, advance the second counter such that the descriptor corresponds to the first processor.

9. A method comprising:
before accessing a shared resource, advance, by a first processor, a first identifier represented by a first counter;
after advancing the first identifier, obtain, by the first processor, the first identifier;
obtain, by the first processor, a second identifier represented by a second counter;
compare, by the first processor, the first identifier and the second identifier;

if the first identifier matches the second identifier, access the shared resource by the first processor, and advance the second identifier by the first processor; and

if the first identifier does not match the second identifier, poll, by the first processor, the second identifier until the second identifier is advanced by a second processor such that the second identifier matches the first identifier, and when the second identifier matches the first identifier, access the shared resource by the first processor and advance the second identifier by the first processor.

10. The method of claim 9, wherein obtaining the second identifier comprises advancing the second counter by a zero step.

11. The method of claim 9, wherein the second counter comprises at least a first field representing the second identifier and a second field indicating whether the shared resource is busy, the method further comprising:

before accessing the shared resource, setting the second field to a value indicating that the shared resource is busy; and

after accessing the shared resource, setting the second field to a value indicating that the shared resource is free.

12. The method of claim 9, wherein the second counter comprises at least a first field representing the second identifier and a second field identifying a processor, the method further comprising:

before accessing the shared resource, setting the second field to a value identifying the first processor.

13. A non-transitory machine-readable storage medium encoded with instructions executable by at least one processor to cause the processor to:

advance a first counter coupled to the processor to a first value;

obtain a second value of a second counter coupled to the processor;

determine whether the first value corresponds to the second value;
if the first value does not correspond to the second value, wait until the first value corresponds to the second value; and
when the first value corresponds to the second value, access a resource, and advance the second counter.

14. The non-transitory machine-readable storage medium of claim 13, wherein the first counter's value represents at least a first identifier, the second counter's value represents at least a second identifier, and wherein determining whether the first value corresponds to the second value comprises determining whether the first identifier is equal to the second identifier.

15. The non-transitory machine-readable storage medium of claim 14, wherein the second counter's value comprises a first field comprising the second identifier and a second field comprising information associated with the shared resource, wherein the first field includes a most significant bit of the second counter.

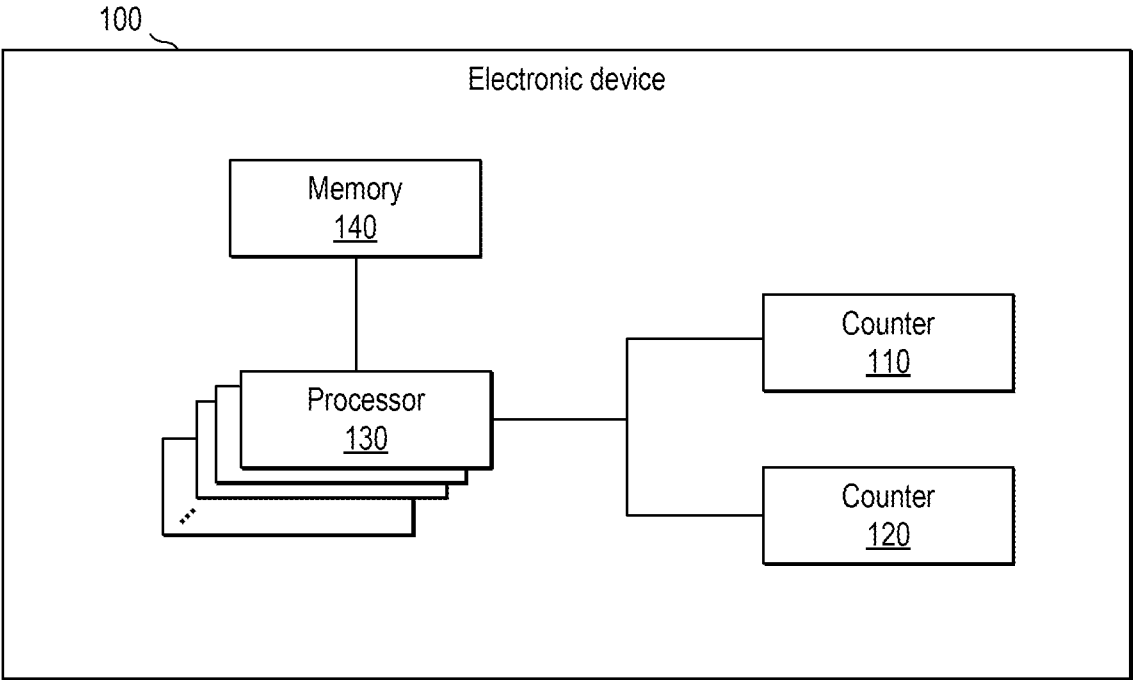
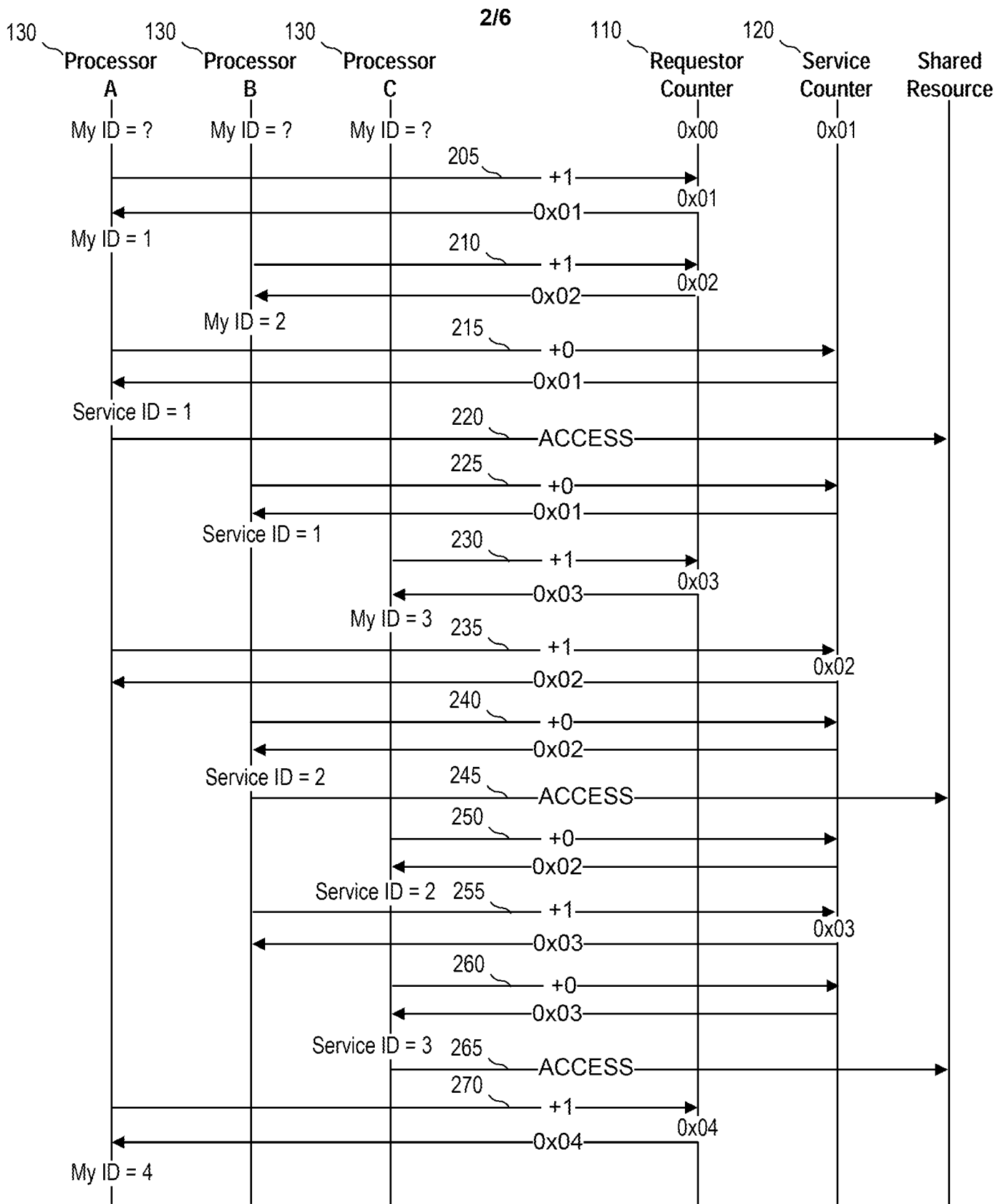
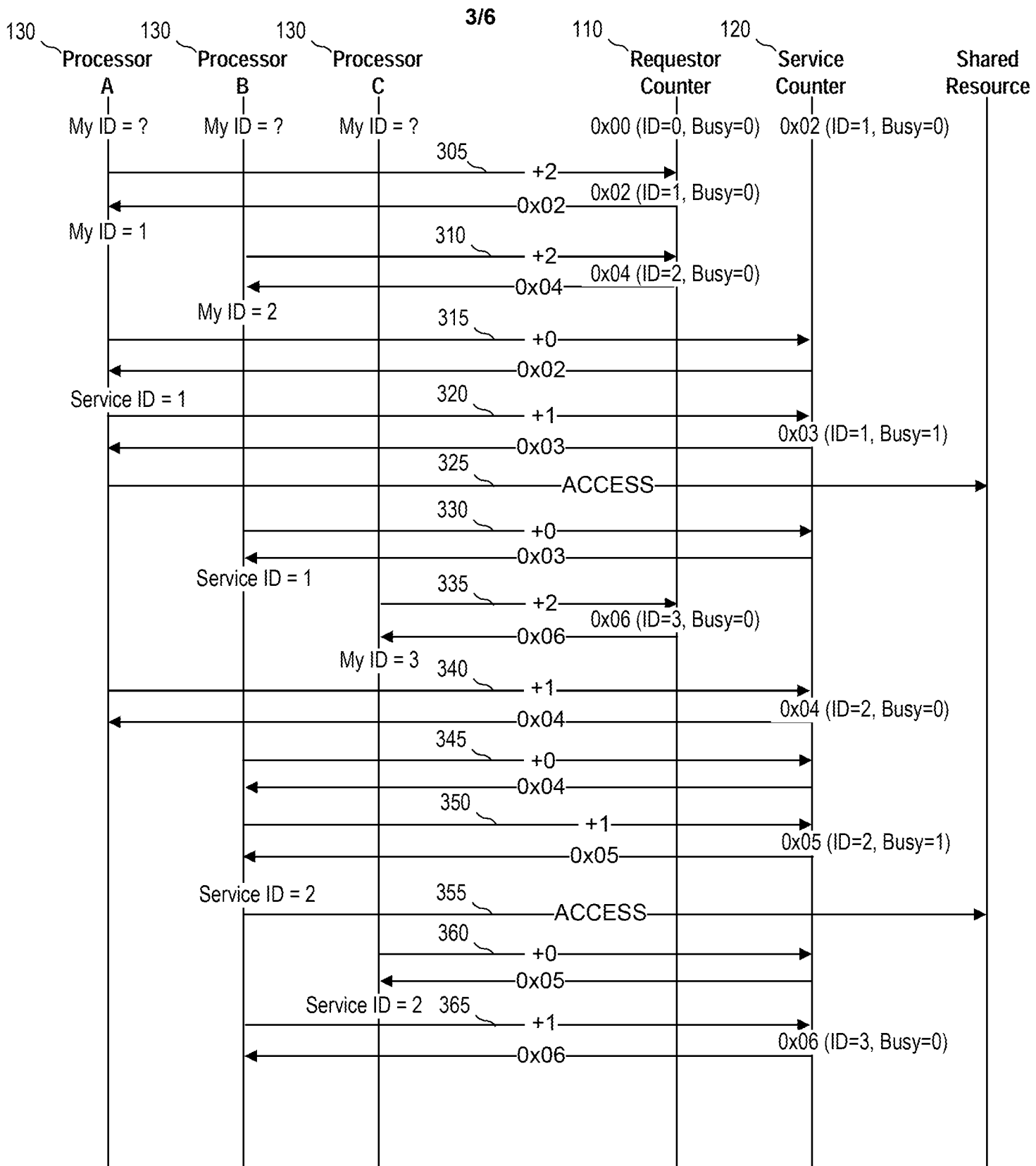
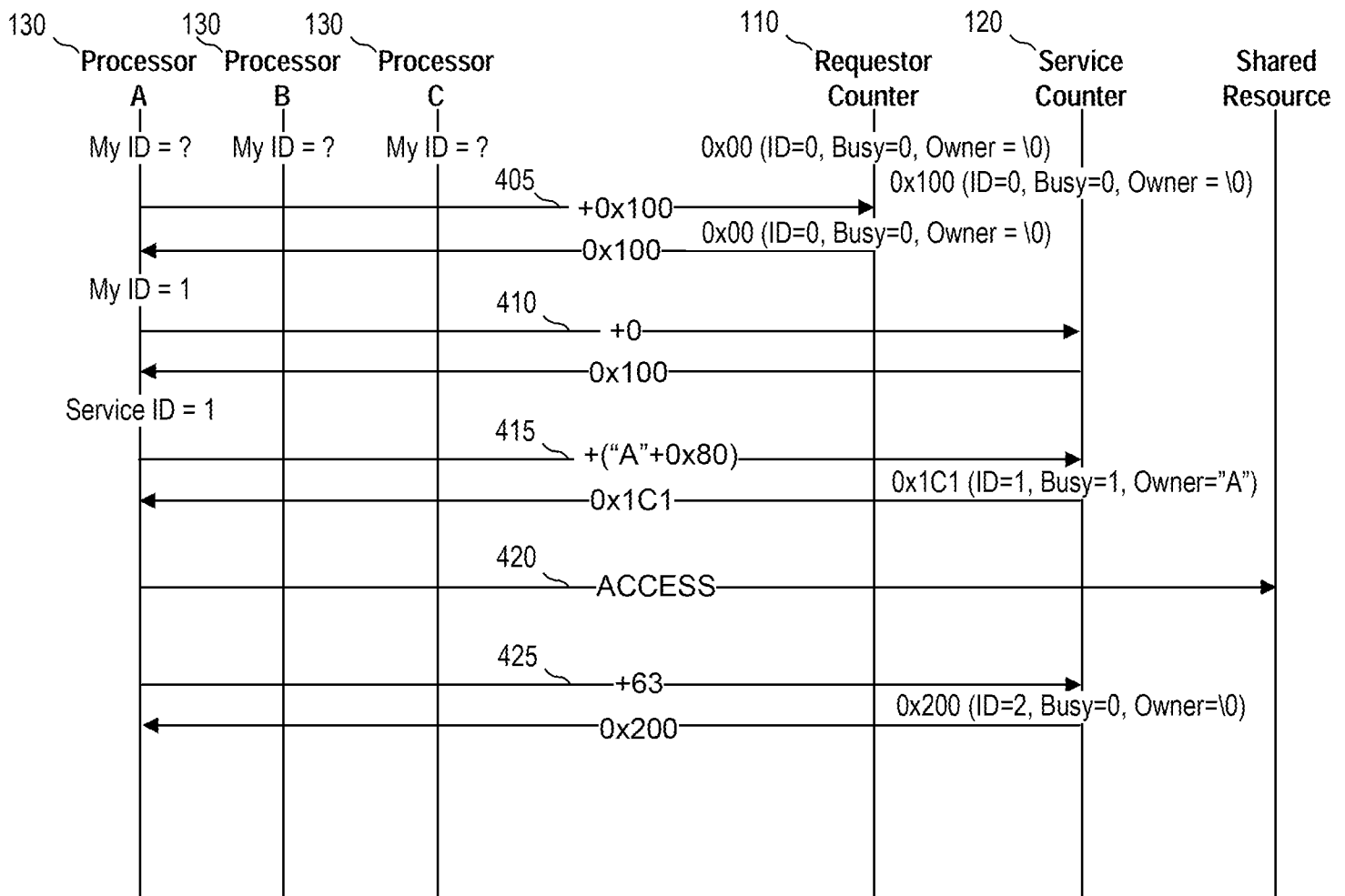


FIG. 1

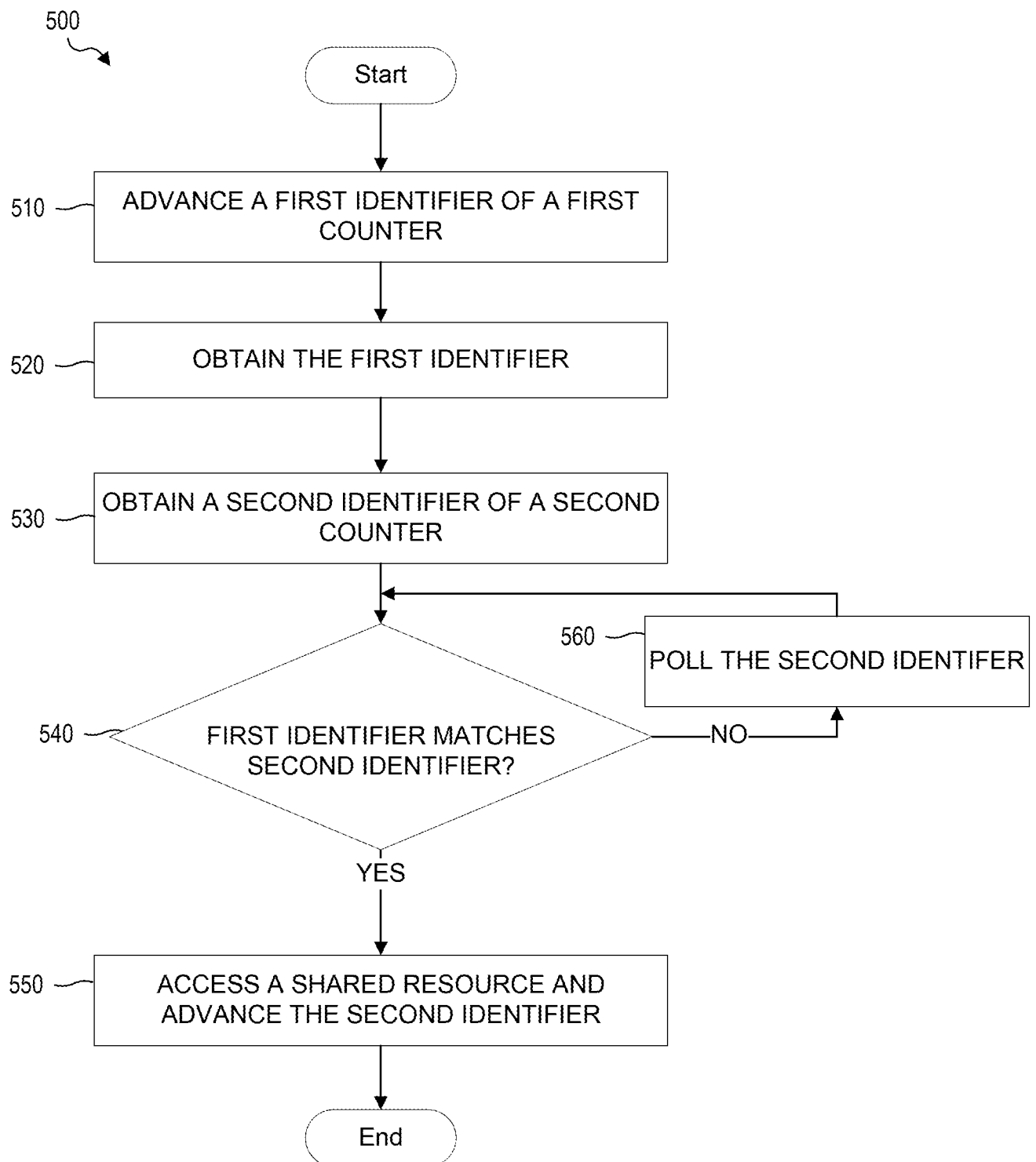
**FIG. 2**

**FIG. 3**

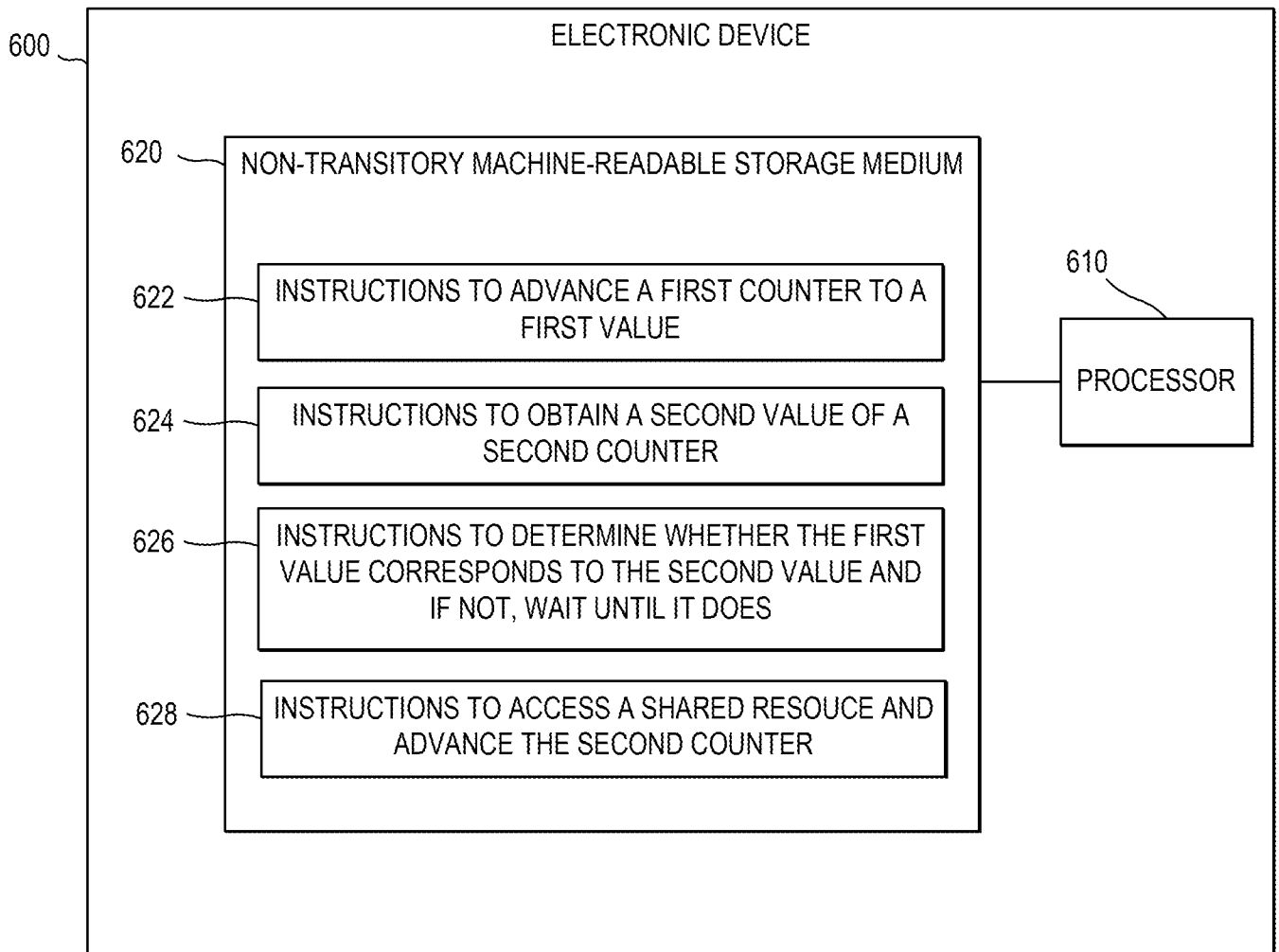
4/6

**FIG. 4**

5/6

**FIG. 5**

6/6

**FIG. 6**

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2015/023174**A. CLASSIFICATION OF SUBJECT MATTER****G06F 15/163(2006.01)i, G06F 9/46(2006.01)i**

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F 15/163; G06F 9/46; G06F 12/14; G06F 12/00; G06F 9/00

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

Korean utility models and applications for utility models

Japanese utility models and applications for utility models

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

eKOMPASS(KIPO internal) & Keywords: shared resources, processors, counters

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2009-0328041 A1 (SUDZILOUSKI ULADZISLAU) 31 December 2009 See abstract; paragraphs [0015]-[0016], [0029]-[0030], [0034]-[0047]; claims 1-9 and figures 1-4.	1-15
A	US 2009-0144742 A1 (SUBHRAVETI DINESH et al.) 04 June 2009 See abstract; paragraphs [0012]-[0014], [0068]-[0080]; claims 1-10 and figures 4-5.	1-15
A	US 2004-0215858 A1 (ARMSTRONG TROY DAVID et al.) 28 October 2004 See abstract; paragraphs [0021]-[0023], [0029]-[0044]; claims 1-12 and figure 2.	1-15
A	US 2003-0145035 A1 (JEREMY S. DE BONET) 31 July 2003 See abstract; paragraphs [0046]-[0051] and claims 1-14.	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

"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

23 December 2015 (23.12.2015)

Date of mailing of the international search report

24 December 2015 (24.12.2015)

Name and mailing address of the ISA/KR

International Application Division

Korean Intellectual Property Office

189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea

Facsimile No. +82-42-472-7140

Authorized officer

HONG, Kyoung Ah

Telephone No. +82-42-481-5668



INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2015/023174

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2009-0328041 A1	31/12/2009	US 8938738 B2	20/01/2015
US 2009-0144742 A1	04/06/2009	CN 101446909 A	03/06/2009
		CN 101446909 B	28/12/2011
		US 8370841 B2	05/02/2013
US 2004-0215858 A1	28/10/2004	CA 2522096 A1	04/11/2004
		CN 1774699 A	17/05/2006
		CN 1774699 B	26/05/2010
		EP 1620801 A2	01/02/2006
		JP 04-410795 B2	03/02/2010
		JP 2006-524381 A	26/10/2006
		KR 10-0843548 B1	04/07/2008
		KR 2006-0004919 A	16/01/2006
		TW I265412 B	01/11/2006
		US 7047337 B2	16/05/2006
		WO 2004-094863 A2	04/11/2004
		WO 2004-094863 A3	24/03/2005
US 2003-0145035 A1	31/07/2003	AU 2002-367516 A1	02/09/2003
		AU 2003-205150 A1	30/07/2003
		AU 2003-205150 A8	30/07/2003
		AU 2003-205209 A1	02/09/2003
		AU 2003-205209 A8	02/09/2003
		AU 2003-207561 A1	30/07/2003
		AU 2003-207561 A8	30/07/2003
		AU 2003-209028 A1	04/09/2003
		AU 2003-209249 A1	02/09/2003
		AU 2003-209249 A8	02/09/2003
		AU 2003-210522 A1	22/09/2003
		AU 2003-214841 A1	30/07/2003
		AU 2003-214841 A8	30/07/2003
		AU 2003-216062 A1	30/07/2003
		AU 2003-216062 A8	30/07/2003
		AU 2003-224610 A1	02/09/2003
		AU 2003-224610 A8	02/09/2003
		AU 2003-236530 A1	02/09/2003
		AU 2003-236530 A8	02/09/2003
		AU 2003-267220 A1	08/04/2004
		AU 2003-303102 A1	23/08/2004
		AU 2003-303102 A8	23/08/2004
		EP 1474762 A2	10/11/2004
		EP 1493102 A2	05/01/2005
		EP 1502207 A2	02/02/2005
		EP 1546911 A1	29/06/2005
		US 2003-0167257 A1	04/09/2003
		US 2003-0167326 A1	04/09/2003
		US 2003-0172172 A1	11/09/2003
		US 2003-0177197 A1	18/09/2003

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2015/023174Patent document
cited in search reportPublication
datePatent family
member(s)Publication
date

US 2003-0177284 A1	18/09/2003
US 2003-0182403 A1	25/09/2003
US 2003-0187869 A1	02/10/2003
US 2006-0021026 A1	26/01/2006
US 2006-0200562 A1	07/09/2006
US 2006-0253461 A1	09/11/2006
US 2007-0192772 A1	16/08/2007
US 7073178 B2	04/07/2006
US 7117506 B2	03/10/2006
US 7130872 B2	31/10/2006
US 7246360 B2	17/07/2007
US 7469300 B2	23/12/2008
US 7818758 B2	19/10/2010
WO 03-060704 A2	24/07/2003
WO 03-060704 A3	06/11/2003
WO 03-060711 A2	24/07/2003
WO 03-060711 A3	13/11/2003
WO 03-060712 A2	24/07/2003
WO 03-060724 A2	24/07/2003
WO 03-062991 A2	31/07/2003
WO 03-062993 A2	31/07/2003
WO 03-062995 A2	31/07/2003
WO 03-063103 A1	31/07/2003
WO 03-067852 A2	14/08/2003
WO 03-069475 A2	21/08/2003
WO 03-077127 A2	18/09/2003
WO 2003-060712 A3	15/04/2004
WO 2003-060724 A3	28/10/2004
WO 2003-062991 A3	14/10/2004
WO 2003-062993 A3	04/03/2004
WO 2003-062995 A3	04/11/2004
WO 2003-067852 A3	10/09/2004
WO 2003-069475 A3	19/08/2004
WO 2003-077127 A3	13/05/2004
WO 2004-027638 A1	01/04/2004
WO 2004-068277 A2	12/08/2004
WO 2004-068277 A3	29/12/2004