A lock circuit for a multi-core processor

Strøm, Torur Biskopstø

Publication date:
2015

Document Version
Publisher's PDF, also known as Version of record

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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
11 September 2015 (11.09.2015)

WIPO | PCT

(10) International Publication Number
WO 2015/132293 A1

(51) International Patent Classification:
G06F 9/48 (2006.01) *G06F 9/52* (2006.01)
G06F 15/173 (2006.01)

(21) International Application Number:
PCT/EP2015/054491

(22) International Filing Date:
4 March 2015 (04.03.2015)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
14157808.8 5 March 2014 (05.03.2014) EP

(71) Applicant: **DANMARKS TEKNISKE UNIVERSITET** [DK/DK]; Anker Engelunds Vej 101 A, DK-2800 Kgs. Lyngby (DK).

(72) Inventor: **STRØM, Tórur Biskopstø**; Skodsborgvej 190 1107, DK-2850 Nærum (DK).

(74) Agent: **ZACCO DENMARK A/S**; Arne Jacobsens Allé 15, DK-2300 Copenhagen S (DK).

(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).

Published:

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

(54) Title: A LOCK CIRCUIT FOR A MULTI-CORE PROCESSOR

(57) Abstract: An integrated circuit comprising a multiple processor cores and a lock circuit that comprises a queue register with respective bits set or reset via respective, connections dedicated to respective processor cores, whereby the queue register identifies those among the multiple processor cores that are enqueued in the queue register. Furthermore, the integrated circuit comprises a current register and a selector circuit configured to select a processor core and identify that processor core by a value in the current register. A selected processor core is a prioritized processor core among the cores that have a bit that is set in the queue register. The processor cores are connected to receive a signal from the current register. Correspondingly: a method of synchronizing access to software and/or hardware resources by a core of a multi-core processor by means of a lock circuit; a multi-core processor configured with an integrated circuit; and a silicon die configured with an integrated circuit.

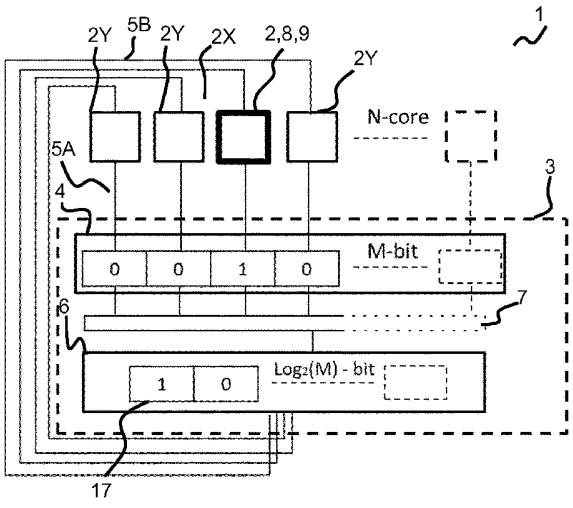

Fig. 1

WO 2015/132293 A1

A lock circuit for a multi-core processor.

RELATED PRIOR ART

A lock circuit is typically assigned e.g. by software to a predetermined resource such as a block of memory or a communications port. A very common issue in computing arises when two or more threads, or processes, need to modify the same resource. If the modification is not atomic, the state of the resource can become corrupt. Several synchronization mechanisms exist to alleviate this, such as mutexes, semaphores, etc. Typically, these locks are implemented on top of some low level atomic operations, such as the compare-and-swap, which requires memory access. This can be fast on uni-core systems, however on multi-core systems memory arbitration is necessary and the latency grows with the increasing numbers of cores.

US 2013/0290584 A1 discloses an apparatus comprising a multi-core processor communicating via a communications bus with off-chip components, such as a memory unit or a lock mechanism. The lock mechanism is configured for scheduling and/or synchronizing multiple threads operating on the multi-core processor requesting access to common data in a memory or a storage unit. The lock mechanism comprises multiple locks, wherein the lock mechanism receives a lock request corresponding to a requested action and assigns a scheduled lock to the requesting thread and the action may be performed.

It is a disadvantage that the multi-core processer and the lock mechanism communicates via a communication bus since a communication delay would most likely occur when multiple cores requesting multiple locks simultaneously, or when multiple actions is dedicated to the same communication bus at the same time.

It is a further disadvantage that the lock mechanism administered the assignment of a lock to a thread/core, since reduced locking speed may occur when multiple threads request multiple locks simultaneously.

SUMMARY

It is an object of the present invention to provide an integrated circuit comprising a multi-core processor and one or more lock circuits, wherein the integrated circuit provides an improved speed and concurrent access of a processor core to a critical resource, i.e. a communication port, a memory or any other component which is located outside the processor core. Furthermore, the object of the present invention is to avoid or reduce any arbitration, such as delays or waiting time, during assignment of a processor core to a lock circuit. Furthermore, the object of the present invention is to avoid any communication between a processor core and a memory or a communication bus during assignment of the processor core to a lock circuit, since that will reduce the speed of a processor core accessing a critical resource.

According to the present invention, the integrated circuit comprises multiple processor cores and a lock circuit that comprises a queue register with respective bits set or reset via respective, connections dedicated to respective processor cores, whereby the queue register identifies those among the multiple processor cores that are enqueued in the queue register. Furthermore, the integrated circuit comprises a current register and a selector circuit configured to select a processor core and identify that processor core by a value in the current register. A selected processor core is a prioritized processor core among the cores that have a bit that is set in the queue register. The processor cores are connected to receive a signal from the current register.

It is an advantage of the present invention that the respective processor cores are directly and exclusively connected to the queue register of the respective lock circuit since that provides no waiting time for a processor core accessing a critical resource. Thereby, a faster allocation of the respective lock circuit (i.e. a critical resource) to a respective processor core is provided.

It is a further advantage of the present invention that a faster allocation of the respective lock circuit to a respective processor core improves the power efficiency of the processor core.

The use of the queue register, where only the respective cores can modify their respective bits in the queue register, serves multiples purposes. It simplifies the enqueing process, so much that requesting a lock, i.e. a lock circuit, and releasing it can be as low as a single clock cycle.

In one or more embodiments, the multiple processor cores may be a central processing unit (CPU) with two or more processor cores which may run multiple instructions at the same time. The multiple processor cores may be a multi-core processor.

In one or more embodiments the queue register with respective bits are set or reset exclusively via respective connections dedicated exclusively to respective processor cores.

In one or more embodiments, the queue register may be a processor register with multiple bits, i.e. N-bits. The queue register may be a data register, an address register, a bit-register and etc. The queue register for a lock circuit is configured with a single bit per processor core or with multiple bits per processor core.

In one or more embodiments, the current register may be a processor register with multiple bits. The queue register may be a data register, an address register, a bit-register and etc. Alternatively, the current register may be a processor register with $\log_2(N)$-bits configured to the respective lock circuit. For example, the queue register may be a 4-bit register and the current register may then be a $\log_2(4)$-bit register, i.e. a 2-bit register.

In one or more embodiments, the connections may be wired connections, pattern connections, high-speed connections, standard connections and non-bus connections transferring information.

In one or more embodiments, the selector circuit may be a selector with a number of outputs and with a number of inputs equal the number of outputs times a factor X, wherein X may be an integer value of two or more. The respective inputs are dedicated to the respective bits in the queue register and allocated with a priority level. A respective input may be active when the respective bit is set by the respective processor core. The selector circuit's output corresponds to the currently active input which has the highest priority. So when an input with a higher priority is present all other active inputs with a lower priority will be ignored. The at least one output of the selector circuit writes the value identifying the selected processor core to the corresponding current register. The selector is configured with a counter, e.g. a circular counter or an up-down counter.

In one or more embodiments, the selected processor core corresponds to a processor core which has set its respective bit in the queue register and has the highest priority in the selector circuit. The respective processor core can be programmed to memorize that it has set the respective bit in the queue register and while receiving a signal from the current register, e.g. a 'low'-bit signal, the respective processor core registers that it has been selected by

the respective lock circuit, and the selected processor core may use the critical resource dedicated to the respective lock circuit.

In one or more embodiments, the current register may be configured with the value identifying the selected processor core. The value may be represented by a single bit or multiple bits.

In one or more embodiments, the selected processor core may receive the value from the respective current register of the respective lock circuit via the connection dedicated to the selected processor core. The processor core knows when it has set or reset its respective bit in the queue register, and thereby, the processor core registers that it has been selected by the respective lock circuit when receiving a signal configured with a value, being a 'low' bit value, from the respective lock circuit, while it has set the respective bit in the queue register.

In one or more embodiments, the lock circuit may comprise a decoder having multiple inputs and multiple outputs. The multiple-inputs receive the value from the current register and decode the value into a signal which is addressed to the respective outputs connected exclusively to the respective processor cores.

In one or more embodiments, the current register may convert the value identifying the selected processor core into a signal which the connected processor cores receive.

According to the present invention, a computer implemented method of synchronizing access to resources by a processor core of a multi-core processor by means of a lock circuit; the method comprising the following steps:

1) Setting or resetting respective bits in a queue register via respective connections dedicated to respective processor cores;

2) Identifying, by the queue register, that the multiple processor cores are enqueued in the queue register;

3) selecting a processor core by a selector circuit configured to select the processor core and identify that processor core by a value in the current register; wherein a selected processor core is a prioritized processor core among the processor cores that have a bit that is set in the queue register;

4) transmitting a signal from the current register to the respective processor cores.

In one or more embodiments, the signal may be a single bit signal or a multibit signal.

In one or more embodiments, the processor core may enqueue for any lock without waiting for other processor cores and in direct response thereto receives free/locked information on a dedicated port. The dedicated port doesn't encode the identity of the lock circuit, only whether a lock circuit enqueued for is granted or not. This reduces the number of wired-connections between the processor cores and the lock circuits.

In one or more embodiments, the resource may be a software resource or a hardware resource.

In one or more embodiments, the respective processor core may enqueuing for a first lock circuit by writing a first multibit signal to a first physical communications port configured to provide instantaneous access from the processor core to a first queue register with a but set or reset by the processor core.

In one or more embodiments, the respective processor core may be configured to receive the signal from a second physical communication port, dedicated to instantaneously indicate whether the processor core got a lock circuit or not.

In one or more embodiments, the communication ports may be single bit ports or multi-bit ports.

In one or more embodiments, the signal may be a single-bit signal received on a single terminal. Alternatively, the signal may be a multi-bit signal.

In one or more embodiments, the processor core may be configured to write a multi-bit signal to a first physical communication port, wherein the multi-bit signal may be encoded to address a predetermined lock circuit.

In one or more embodiments, the steps of enqueuing and reading are performed by programme code run by the processor core.

In some embodiments, the programme code may execute in consecutive the respective assemble code instructions within N clock cycles, e.g. within 5 to 10 clock cycles, 5 to 30 clock cycles and 10 to 20 clock cycles. The assembler code may be configured to enqueuing for a first lock, requesting receiving of a signal from a second physical port.

The integrated circuit comprises selector circuit which is configured to maintain a selected processor core selected while that selected processor core maintains its bit in the queue register set, and select a prioritized processor core in response to the event that the selected processor core resets its bit in the queue register; wherein the prioritized processor core is a processor core with its bit set in the queue register.

In this specific example, “maintain” means that the processor core is selected until its respective bit in the queue register is reset.

In this specific example, “selected” means that a resource or a lock circuit has been allocated to the processor core.

In one or more embodiments, a processor core resigns from the lock circuit by resetting the respective bit in the respective queue register. The processor core keeps the lock circuit, i.e. a resource, assigned to the processor core as long as it needs it.

In one or more embodiments, the selected processor core is selected firstly and then secondly, in response to the event that the selected processor core resets its bit in the queue register, a prioritized processing core is selected.

It is an advantage of the present invention that any conflict between the selected processor core and the other processor cores, trying to assessing the respective lock circuit, is avoided.

It is a further advantage of the present invention that a processor, comprising more than one processor core, would perform more reliable since the selected processor core to a respective lock circuit avoids any conflict from other processor cores trying to access the same respective lock circuit.

The integrated circuit comprises a selector circuit which is configured with a priority encoder.

In one or more embodiments, the priority encoder is configured to simplify the integrated circuit and/or a lock circuit by reducing the number of connections between a respective queue register and a current register, via the respective priority encoder.

In one or more embodiments, the priority encoder is configured with a number of inputs and one or more outputs. The respective inputs are dedicated to the respective bits in the queue register and allocated with a

priority level. A respective input may be active when the respective bit is set by the respective processor core. The priority encoder's output corresponds to the currently active input which has the highest priority. So when an input with a higher priority is present all other active inputs with a lower priority will be ignored. The at least one output of the selector circuit writes the value identifying the selected processor core to the corresponding current register.

Subsequently, the processor core resets its respective bit in the queue register, and the priority encoder ignores the processor core and selects a prioritized processor core which has set its respective bit in the queue register, and so on.

In one or more embodiments, the priority encoder may be a circular priority encoder, a round-robin priority encoder, Up-and-Down priority encoder or any other priority encoder.

It is an advantage of the present invention that the selected circuit is configured with a priority encoder since priority encoder improves the speed of allocating respective lock circuit, i.e. a critical resource, to a respective processor core.

It is a further advantage of the present invention that a faster allocation of the respective lock circuit to a respective processor core would improve the way a multiple-core processor would handle multiple instructions and/or complicated instructions.

The integrated circuit comprises multiple of said lock circuits, wherein processor cores connect to queue registers of respective multiple lock circuits, and wherein the processor cores are connected to receive the values in the current registers of respective lock circuits.

Thereby, multiple critical resources may be assigned or locked to a processor core, and/or multiple respective processor cores may be assigned to respective multiple critical resources.

It is advantage of the present invention that multiple processor cores dedicated to respective lock circuits may perform parallel operations or threads dedicated to the respective lock circuits without any conflict between the processor cores.

The integrated circuit comprises multiple of said lock circuits, and a blocking circuit with multiple input ports connected via respective multi-bit connections to a current register of a respective lock circuit, and multiple output ports connected via respective connections dedicated to respective processor cores, such that a processor core is connected to receive a signal from a current register in response to the processor core being selected by that current register. Furthermore, the blocking circuit encodes the signal to a predefined processor core with a transition in response to the value of any of the current registers indicating the identity of that predefined core.

It is an advantage of the present invention that the blocking circuit improves the simplicity of the multi-core processor by reducing the number of wiring going back to the processor cores, i.e. between one or more respective current register of the respective lock circuits and the processor cores.

For example, the multi-core processor comprises 4 cores and 64 lock circuits the blocking unit has four outputs each connected exclusively to the respective processor cores. If the blocking unit was not present, each processor cores may then need one input port for each connection to the respective lock circuits, i.e. in this particular example, each core may need 64 input ports.

In one or more embodiments, the blocking circuit may comprise multiple inputs and multiple outputs, and respective multiple encoders dedicated to the respective processor cores. The respective encoder is configured to encode the signals, received by one or more lock circuits, to the respective processor core with a transition in response to the value of any of the current

registers, wherein the response indicates the identity of that predefined processor core.

In one or more embodiments, the predefined processor core may be the respective processor core connected to the respective encoder of the blocking unit.

In one or more embodiments, the transition may be a pulse event, a clock event or a high/low event etc.

In one or more embodiments, the lock information may disclose the value that identifies the selected processor core. Alternatively, the lock information may disclose a bit-signal, wherein a 'high'-bit signal or a 'low'-bit signal indicates an unselected or a selected processor core, respectively.

In one or more embodiments the connections may be wired connections. In one or more embodiments the respective connections are dedicated exclusively to respective processing cores.

In one or more embodiments, the lock circuit outputs by default a reset bit (not asserted) signal to a processor core; then, in case the lock circuit receives a set bit (i.e. request) signal from the processor core and the lock circuit is free, the bit signal to the processor core remains not asserted; whereas in case the lock circuit is not free, the bit signal to the processor core is set (asserted) until the processor core withdraws its request in the queue register (by resetting its bit there) or until the moment the lock circuit becomes free.

In one or more embodiments, the lock circuit outputs a level transition with a predefined duration or any un-determined duration to a processor core in the case the lock circuit receives a set bit from the processor core and is unassigned to any other processor core, i.e. the lock circuit is free. For example, the processor core is dedicated to a lock circuit when receiving a level transition with duration of period T, or the processor core is dedicated to

more than one lock circuit when receiving a level transition with duration of above period T.

In one or more embodiments, a processor core may be configured with one or more programming strategies utilizing the level transition signal or the bit signal to the processor core in different ways. Both signals may be used by the processor core for using a critical resource, to synchronize its work with more than one critical resource, to prepare the processor core for the forthcoming use of the critical resource and etc.

The integrated circuit comprising respective connections from the multiple output ports to respective processor cores are single-wire connections.

In one or more embodiments, a processor core may receive a single-terminal signal, via a single-wire connection, which may be encoded to indicate whether a lock device is granted or not.

It is an advantage of the present invention that the integrated circuit comprising single-wire connections allows extremely fast communication to the processor core.

In one or more embodiments, the blocking circuit may be configured to communicate a single-bit signal to the processing cores via the respective output ports. The single-bit signal is ‘low’ when a lock circuit is free and becomes ‘high’ when the lock circuit is locked and remains ‘high’ until the lock circuit gets free or the processor core withdraws from the queue register by resetting a dedicated bit there.

The integrated circuit comprises one or more of said lock circuits, wherein the processor cores are configured with a processor identification value and are connected to access the value of one or more current registers.

The processor identification value may be stored in the respective processor core identifying the processor core to other components.

In one or more embodiments, a predetermined processor core has access to a current register of a predetermined lock circuit and may from its identification value determine whether the value of a current register indicates that the predetermied processor core is granted access to a ressource assigned to the predetermined lock.

Alternatively, the processor core may be loaded with a programmable unit that is configured to compare the identification value with the received value from the current register of the respective lock circuit.

Furthermore, the identification value may be stored locally in a processor core.

The integrated circuit comprises multiple of said lock circuits, and wherein multiple processor cores are connected to respective multiple decoder circuits. Furthermore, the decoder circuit is connected to multiple lock circuits, such that a processor core can address a predefined lock circuit by means of an encoded multibit signal. Additionally, the decoder circuit is configured to set or reset a bit in the queue register.

The processor core may address a predefined lock circuit by transmitting an encoded multibit signal to the respective lock circuit.

It is an advantage of the present invention that the decoder is configured to reduce the number of wired-connections between a processor core and plurality of lock circuits.

It is a further advantage of the present invention that the decoder improves the simplicity of the integrated circuit.

The integrated circuit comprising a multibit communications bus through which the processor cores communicate with off-chip components when access to the multibit communications granted, and wherein said wired connections dedicated exclusively to respective processing cores are separated from the multibit communications bus.

The advantage of separating the multibit communciations bus and the wired connections is an improved speed of the processor core to request a lock circuit, i.e. a critical resource.

In one or more embodiments a conventional communicaiton bus provides access to storage memory, communications ports, such as USB, PCI, or other ports.

A multi-core processor is configured with an integrated circuit.

A silicon die is configured with an integrated circuit.

In one or more embodiments the silicon die, i.e. a semiconductor die, is configured with an intra-die communications bus with an intra-die bus controller and/or an inter-die communications bus such as a PCI or a PCIe bus with an inter-die bus controller to connect to provide communication with off-die devices, such as I/O devices, RAM or external storage unit(s).

In one or more embodiments the semiconductor die comprises a cache memory which is a block of high-speed memory for temporary data storage located on the same silicon die as the CPU.

The computer-implemented method, wherein it is evaluated whether the processor core got the first lock circuit and wherein the processor core performs the work required at least first lock circuit and then release the at least first lock circuit.

The computer-implemented method, wherein it is evaluated whether the processor core got the first lock circuit, and wherein the step of enqueuing for the second lock circuit is performed on the condition that the processor core got the first lock circuit.

The computer-implemented method, wherein it is evaluated whether the processor core got the first lock circuit and wherein the processor core performs the work required at least first lock circuit and then release the at least first lock circuit.

The computer-implemented method, wherein the steps designated 1) and 2) and/or the steps designated 3) and 4) of the method are encoded in machine instructions arranged in a consecutive order.

The computer-readable medium encoded with a programme to perform the computer-implemented method as set forth in any of the method claims above.

According to the present invention, a computer implemented method of synchronizing access to resources by a processor core of a multi-core processor by means of lock circuits; the method comprising the following steps:

1) enqueuing for a first lock circuit by writing a first multibit signal to a first physical communications port configured to provide instantaneous access from the processor core to a first queue register with a bit set or reset by the processor core;

2) receiving a signal from a second physical communication single-bit port, dedicated to instantaneously indicate whether the processor core got a lock circuit or not; processing this signal as an indication that the processor core got the first lock circuit;

3) enqueuing for a second lock circuit by writing a second multibit signal to the first physical communications port of the processor core connected for instantaneous access also to a second queue register with a bit set or reset by the processor core;

4) receiving again a signal from the second physical communication single-bit port; processing this signal as an indication that the processor core got the second lock circuit.

In one or more embodiments, the processor core may enqueue for any lock without waiting for other processor cores and in direct response thereto receives free/locked information on a dedicated port. The dedicated port doesn't encode the identity of the lock circuit, only whether a lock circuit enqueued for is granted or not. This reduces the number of wired-connections between the processor cores and the lock circuits.

In one or more embodiments, the resource may be a software resource or a hardware resource.

In one or more embodiments, the bit may be set or reset exclusively by the processor core, and the second physical communication port may be a single-bit or a multi-bit port.

In one or more embodiments, the signal may be a single-bit signal received on a single terminal. Alternatively, the signal may be a multi-bit signal.

In one or more embodiments, the at least first multibit signal is encoded to address a predetermined lock circuit.

In one or more embodiments, the pairwise steps of enqueuing and reading are performed by programme code run by the processor core.

In some embodiments, the programme code may execute in consecutive the respective assemble code instructions within N clock cycles, e.g. within 5 to 10 clock cycles, 5 to 30 clock cycles and 10 to 20 clock cycles. The assembler code may be configured to enqueuing for a first lock, requesting receiving of a signal from a second physical port, enqueuing for a second lock, requesting receiving again of a signal from the second physical single-bit port and etc.

According to the present invention, a computer implemented method of synchronizing access to resources by a processor core of a multi-core processor by means of lock circuit comprises following steps:

1) enqueuing for a first lock by writing a first multibit signal to a first communications port configured to provide instantaneous access from the processor core to a first queue register with a bit set or reset by the processor core;

2) receiving a signal from a second physical port, dedicated to instantaneously indicate whether the processor core got a lock or not; processing this signal as an indication that the processor core got the first lock.

In one or more embodiments, the signal may be a single bit signal or a multibit signal.

In one or more embodiments, the processor core may enqueue for any lock without waiting for other processor cores and in direct response thereto receives free/locked information on a dedicated port. The dedicated port doesn't encode the identity of the lock circuit, only whether a lock circuit enqueued for is granted or not. This reduces the number of wired-connections between the processor cores and the lock circuits.

In one or more embodiments, the resource may be a software resource or a hardware resource.

In one or more embodiments, the bit may be set or reset exclusively by the processor core, and the second physical communication port may be a single-bit or a multi-bit port.

In one or more embodiments, the signal may be a single-bit signal received on a single terminal. Alternatively, the signal may be a multi-bit signal.

In one or more embodiments, the at least first multibit signal is encoded to address a predetermined lock circuit.

In one or more embodiments, the steps of enqueuing and reading are performed by programme code run by the processor core.

In some embodiments, the programme code may execute in consecutive the respective assemble code instructions within N clock cycles, e.g. within 5 to 10 clock cycles, 5 to 30 clock cycles and 10 to 20 clock cycles. The assembler code may be configured to enqueuing for a first lock, requesting receiving of a signal from a second physical port.

Another object of the present invention is to provide an integrated circuit comprising multiple processor cores and a lock circuit. Furthermore, the lock circuit comprises a queue register with respective bits set or reset via respective, connections dedicated to respective processor cores, whereby the queue register identifies those among the multiple processor cores that are enqueued in the queue register. Additionally, the lock circuit comprises a current register, and a selector circuit, being a priority encoder, is configured to select a processor core and identify that processor core by a value in the current register; wherein a selected processor core is a prioritized processor core among the cores that have a bit that is set in the queue register; and wherein the processor cores are connected to receive a signal from the current register.

By having the queue register and the selector circuit, being a priority encoder, preferable a circular priority encoder, prevents a processor core from never being selected to access a lock circuit requested by the processor core. Thereby, the combination of the queue register and the round-robin priority encoder results in a starvation free lock circuit.

The combination of the queue register, the current register, and the round-robin priority encoder (CPE) creates an efficient round-robin queue, which means that switching between cores in queue during a single clock cycles is possible.

The round-robin queue can be implemented in a multitude of ways, even in software.

Whilst the CPE is efficient, it can suffer from using a lot of hardware resources. Instead of a CPE a round-counter could be used. The round-counter works as following; the current register starts at 0 and counts up to N, after which it wraps around counting from 0 again. If the current register is at processor core C, and core C's queue bit is set, the current register remains at C until its queue bit goes low.

An advantage of using the round-counter is that the locking circuit may be able to operate at a different clock rate than the processor cores.

In one or more embodiments the clock rate of the lock circuit may be much higher than the processor cores clock rate. For example, if a processor core runs at 100 MHz, the locking circuit can run at 1 GHz. Switching between the cores then takes a 10th of the time if the lock circuit ran at the same rate as the processor cores.

In order to be able to operate at a different clock rate the connections going in (request lines from processor cores) and out (response lines to processor cores) the lock circuit are configured to two or more clock domains. In one or more embodiments, each connection going in and out the lock circuit has at

least one Flip-Flops (FF) clocked at a rate. A first Flip-Flop of the connection going from a processor core and into the lock circuit is clocked differently compared to a second Flip Flop of connection going out from the same lock circuit and into the same processor core.

In one or more embodiments, the first Flip-Flop may be clocked at a higher rate than the second Flip-Flop.

A further advantage of the lock circuit having a round-counter with a single Flip-Flop connected to each connection going in and out from the lock circuit is that the size of the lock circuit is much smaller than a lock circuit having a CPE instead of the round-counter.

In one or more embodiments it is possible to combine the round-counters, including the Flip-Flop connected to each connection going in and out from the lock circuit, with CPEs. The advantage of the combination is that the switching between processor cores requesting respective lock circuits is both single cycled and at the clock rate of the lock circuit (i.e. clock rate configured to the First Flip-Flop).

In one or more embodiments, the locking circuit is real-time compatible. This requires the locking circuit to have well defined bounds on processing time (potentially constant and single cycle in the CPE) but also requires the queueing to avoid starvation. In real-time system starvation is the situation where a task (or processor core in our case) is prevented from accessing a resource because of scheduling/timing/priority issues.

The current register not only allows the efficient CPE integration, but also gives several options for notifying the processor cores of lock statuses by the value in the current register. In one or more embodiments, that can be done by simply expose M*log(N) wires, where each set of log(N) wires indicates the current owner of the respective lock and M is an integer value. Individual processor cores can then use a multiplexer when reading the value in the current register. This means that there are N multiplexers, one for each

processor core, but each processor core only needs log(M)+log(N) wires to its multiplexer, where log(M) are the lock status select lines, and log(N) are the selected lock's current owner.

Alternatively, that can be done by using M*N comparators, one for each core for each lock circuit. The output from each comparator is a single output line (N*M output lines in total) indicating whether a given processor core is the owner of a given lock circuit. This is compared to the processing core's queue bit in the respective queue register of the respective lock circuit, and the resulting output line indicates whether a given core is enqueued and if so, whether it is the current owner of the respective lock circuit. We could direct these output line to the respective processing cores, meaning each processing cores receives M output lines. In a preferred solution the number of output lines can be reduced by, e.g., AND all these output lines, resulting in a single input line for each processing core, i.e., N input lines in total. This input line indicates whether a given core is enqueued in any lock circuits and if so, if it owns all of them.

In one or more embodiments the value in the current register is a bit-signal, wherein a ‘high’-bit signal or a ‘low’-bit signal indicates an unselected or a selected processor core, respectively.

Alternatively, according to the invention, a computer implemented method of synchronizing access to resources by a processor core of a multi-core processor by means of a lock circuit; the method comprising the following steps:

1) Setting or resetting respective bits in a queue register via respective connections dedicated to respective processor cores;

2) Identifying, by the queue register, that the multiple processor cores are enqueued in the queue register;

3) selecting a processor core by a selector circuit, being a priority encoder, configured to select the processor core and identify that processor core by a value in the current register; wherein a selected processor core is a prioritized processor core among the processor cores that have a bit that is set in the queue register;

4) transmitting a signal from the current register to the respective processor cores.

BRIEF DESCRIPTION OF THE FIGURES

A more detailed description follows below with reference to the drawing, in which:

fig. 1 shows an integrated circuit comprising multiple processor cores and at least a lock circuit,

fig. 2 shows a timing diagram of processor cores receiving a signal from a current register of a respective lock circuit of the integrate circuit,

fig. 3 shows an integrated circuit comprising multiple processor cores and multiple lock circuits,

fig. 4 shows an integrated circuit comprising multiple processor cores, multiple lock circuits and a blocking circuit,

fig. 5 shows a timing diagram of processor cores receiving a signal from a current register in response to the processor core being selected by the current register, and wherein the blocking unit encode the signal to a predefined processor core.

fig. 6 shows an integrated circuit comprising multiple processor cores, multiple lock circuits and a blocking circuit, and wherein the multiple processor cores are connected to respective multiple decoder circuits,

fig. 7 shows an integrated circuit comprising a multibit communications bus separated from wired connections dedicated exclusively to respective processing cores,

fig. 8 shows a computer-implemented method of synchronizing access to resources by a processor core of a multi-core processor comprising a lock circuit,

fig. 9 shows a further computer-implemented method of synchronizing access to resources by a processor core of a multi-core processor comprising a lock circuit,

fig. 10 shows an alternatively computer-implemented method of synchronizing access to resources by a processor core of a multi-core processor comprising multiple lock circuits,

fig. 11 shows a further alternatively computer-implemented method of synchronizing access to resources by a processor core of a multi-core processor comprising multiple lock circuits and a block circuit.

fig. 12 shows a further alternatively computer-implemented method of synchronizing access to resources by a processor core of a multi-core processor comprising multiple lock circuits.

DETAILED DESCRIPTION

Item	**No**
Integrated circuit	1
Processor core	2
Multiple processor cores	2X
Other processor cores	2Y
Lock circuit	3

First lock circuit	3A
Second lock circuit	3B
Lock circuits	3X
Queue register	4
First queue register	4A
Second queue register	4B
Wired connection	5
Connection (Connection from a processor core)	5A
Connection (Connection to a processor core)	5B
Current register	6
Current registers	6X
Selector circuit	7
Selected processor core	8
Prioritized processor core	9
Priority encoder	11
Blocking circuit	12
Input ports	13A
Output ports	13B
Multi-bit connections	14
Transition	15
First transition	15A
Second transition	15B
Third transition	15C
Fourth transition	15D
Fifth transition	15E
Value (indicating the ID of the predefined core)	17
Decoder circuit	18
Decoder circuits	18X

Off-chip components	21
Integrated circuit	22
Silicon die	23
Multi-core processor	26
clock	27
Event	28
Computer-implemented method	80
Further computer-implemented method	90
Alternatively computer-implemented method	100
Second alternatively computer-implemented method	200
Third alternatively computer-implemented method	300

Fig. 1 shows an integrated circuit 1 comprising a multiple processor cores 2X, and a lock circuit 3 that comprises a queue register 4 with respective bits set or reset via respective, connections 5A dedicated to respective processor cores 2X, whereby the queue register 4 identifies those among the multiple processor cores 2X that are enqueued in the queue register 4. Furthermore, the lock circuit 3 comprises a current register 6, and a selector circuit 7 configured to select a processor core 2 and identify that processor core 2 by a value 17 in the current register 6. Furthermore, a selected processor core 8 is a prioritized processor core 9 among the cores 2X that have a bit that is set in the queue register 4; and wherein the processor cores 2X are connected 5B to receive a signal from the current register 6.

In this particular example the integrated circuit 1 comprises four processor cores 2X connected via respective connections 5A dedicated to the respective bits in a queue register 4. A respective bit in the queue register 4 is set by the respective processor core 2, and the remaining bits are not set by the respective other processor cores 2Y. The processor core 2 has been selected by a selector circuit 7 to become the prioritized processor core 9,

and the current register 6 identifies the prioritized processor core 9 by a value 17. The processor cores 2X receive a signal from the current register 6 by the connections 5B, wherein the prioritized processor core 9 becomes the selected processor core 8.

Fig. 2 shows a timing diagram which illustrates a selector circuit 7 configured to maintain a selected processor core 8 selected while that selected processor core 8 maintains its bit in the queue register 4 set. Furthermore, the selector circuit 7 may select a prioritized processor core 9 in response to an event 28 that the selected processor core 8 resets its bit in the queue register 4, and wherein the prioritized processor core 9 is a processor core 2 with its bit set in the queue register 4.

In this particular example, four processor cores (C1 – C4) are connected 5A exclusively to respective bits in the queue register 4 of a respective lock circuit 3. The respective processor cores (C1 – C4) receive via the dedicated connections 5B a respective signal (C1-signal to C4-signal) from the respective current register 6. The respective signals (C1-signal to C4-signal) may be configured with an event 28,

A signal may be configured to a 'low'- event 28 when the respective bit is set in the queue register 4 and the respective processor core 2 is selected by the selector circuit 7 or when the respective bit is reset (i.e. not asserted).

A signal may be configured to a 'high'-event 28 when the respective bit is set in the queue register 4 and the respective processor core 2 is not selected.

In first clock cycle 27, none of the bits are set and the processor cores (C1-C4) receive via the respective connections 5B a respective 'low'-event signal 28.

In second clock cycle 27, the second processor core C2 sets its respective bit in the queue register 4, and while none of the other processor cores (C1, C3 and C4) are enqueued in the queue register 4, the second processor core

(C2,8) is selected by the selector circuit 7. All four processor cores (C1 – C4) receive a respective 'low'-event signal 28 via the respective connections 5B.

In third clock cycle 27, all four processor cores (C1-C4) set the respective bits in the queue register 4, and the enqueued processor cores (C1, C3, and C4) receive a 'high'-event signal 28 via the respective connections 5B. In this particular example the selector circuit is configured to a round-robin prioritization, and thereby, the processor core C3 becomes the prioritized processor core (C3, 9).

The selector may also be configured to any other kind of a prioritization scheme.

In fourth clock cycle 27, the previous selected processor core (C2, 8) becomes unselected since the processor core C2 resets its respective bit in the queue register 4. The previous prioritized processor core (C3, 9) becomes the selected processor core (8,C3) while its respective bit is kept asserted in the queue register 4. Again, according to the round-robin prioritization, the processor core C4 becomes the prioritized processor core (C4, 9) while its respective bit is kept asserted in the queue register 4. The enqueued processor cores (C1, C4) receive a 'high'-event signal 28 and the selected processor core C3 receives a 'low'-event signal 28 via the respective dedicated connection 5B.

In fifth clock cycle 27, the previous selected processor core C3 resets its respective bit in the queue register 4 and the previous prioritized processor core C4 becomes the selected processor core (C4, 8). The processor core C1 becomes the prioritized processor core (C1, 9) while its respective bit is kept set in the queue register 4.

In sixth clock cycle 27, the configuration of the bits in the queue register 4 has not changed.

In seventh clock cycle 27, the previous selected processor core C4 resets its respective bit in the queue register 4 and the previous prioritized processor core C1 becomes the selected processor core (C1,8). The processor core C2 becomes the prioritized processor core (C2, 9) while its respective bit is kept set in the queue register 4. The enqueued processor core C2 receives a 'high'-event signal 28, and the selected processor core C1 and the non-enqueued processor cores (C3, C4) receive a respective 'low'-event signal 28.

In eight clock cycle 27, the configuration of the bits in the queue register 4 has not changed.

Fig. 3 shows an integrated circuit 1 comprising multiple of said lock circuits 3X wherein the respective processor cores 2X are connected 5A to queue registers 4 of respective multiple lock circuits 3X, and the processor cores 2X are connected 5B to receive the values 17 from respective lock circuits 3X.

The connections 5A and 5B may be single-bit connections dedicated exclusively to each processor cores 2X. The value 17 may be configured to a single bit signal or a multi-bit signal.

Fig. 4 shows an integrated circuit 1 comprising multiple of said lock circuits 3X, and a blocking circuit 12 with multiple input ports 12A, wherein the respective input ports are connected to respective current registers 6 of respective lock circuits 3X via respective multi-bit connections 14. Furthermore, the blocking circuit 12 comprises multiple output ports 12B connected to respective processor cores 2X via respective connections 5B, such that a processor core 2 is connected to receive a signal from a current register 6 in response to the processor core 8 being selected by that current register 6. Additionally, the blocking circuit 12 encodes the signal to a predefined processor core with a transition 15 in response to the value 17 of any of the current registers 6X indicating the identity of that predefined core.

Fig. 5 shows a timing diagram which illustrates the blocking circuit 12 receiving a signal from a respective current register 6 of a respective lock circuit 3 via the respective multi-bit connections 14. The blocking circuit 12 is configured to encode the signal to a predefined processor core 2 with a transition 15 in response to the value 17 of any of the current registers 6X indicating the identity of that predefined core 2.

In first transition 15A, the blocking circuit receives a signal from a first and a second lock circuit (3A, 3B), and in this particular example, the blocking circuit encodes the respective received signals to the first processor core C1. Thereby, both lock circuits (3A, 3B) are dedicated to the first processor core C1.

Alternatively, the time period of the first transition 15A may be equal to XT, where X is an integer value going from 1 to 100, 1 to 50, 1 to 25 and 1 to 10, and T is a time value in micro seconds, milliseconds, seconds or minutes. The integer value X may correspond to the number of lock circuits which are dedicated to one or more processor cores.

Furthermore, the respective connections 5B from the multiple output ports 12B to respective processor cores 2X are single-wire connections, single high-speed connections, single- bit connections or trace connections.

Furthermore, the processor cores 2X are configured with a processor identification value and are connected to access the value 17 of one or more current registers.

In fig. 5, in second transition 15B, the received signals from the first and the second lock circuit (3A, 3B) includes a value which indicates that the third processor core C3 is assigned to both lock circuits (3A, 3B).

In third transition 15C, the received signals from the first and the second lock circuit disclose a value 17 which identifies that the second and the first

processor core are assigned to the first and the second lock circuit (3A, 3B), respectively.

In fourth transition 15D, the received signal from the first lock circuit 3A discloses a value 17 which identifies that the third processor core C3 is dedicated to the first lock circuit 3A.

In fifth transition 15E, the received signal from the second lock circuit 3B discloses a value 17 which identifies that the fourth processor core C4 is dedicated to the second lock circuit 3B.

Fig. 6 shows an integrated circuit 1comprising multiple of said lock circuits 3X and multiple processor cores 2X that are connected to respective multiple decoder circuits 18X. The decoder circuit 18 is connected to multiple lock circuits 3X, such that a processor core 2 can address a predefined lock circuit by means of an encoded multibit signal. Thereby, the respective processor core 2 is configured to set or reset a bit in the respective queue register 4 of the respective lock circuit 3 via the respective decoder circuit 18.

Fig. 7 shows a multi-core processor 26 configured with an integrated circuit 1 comprising a multibit communications bus 20 through which the processor cores 2X communicate with off-chip components 21 when access to the multibit communications granted. The said wired connections 5 dedicated exclusively to respective processing cores are separated from the multibit communications bus 20.

In this particular example the integrated circuit 1 is configured to a silicon die 23.

Fig. 8 shows a further computer-implemented method 80 of synchronizing access to a resource by a processor core 2 of a multi-core processor 26 by means of a lock circuit 3. In first step 80A, the respective processor cores 2X set or reset respective bits in a queue register 4 via respective connections 5A dedicated to the respective processor cores 2X.

In second step 80B, the queue register 4 identifying that the multiple processor cores (2X) are enqueued in the queue register (4).

In third step 80C, includes selecting a processor core (2) by a selector circuit (7) configured to select the processor core (2) and identify that processor core (2) by a value (17) in the current register (6). A selected processor core (8) is a prioritized processor core (9) among the cores (2X) that have a bit that is set in the queue register (4).

In fourth step 80D, includes transmitting a signal from the current register (6) to the respective processor cores (2X).

Alternatively, in fifth step 80E, includes evaluating whether the processor core (2) got the first lock circuit (3A) and wherein the processor core (2) performs the work required at least first lock circuit (3A) and then release the at least first lock circuit (3A).

Fig. 9 shows a further computer-implemented method 90 of synchronizing access to a resource by a processor core 2 of a multi-core processor 26 by means of a lock circuit 3. In first step 90A, a processor core 2 enqueuing for a first lock circuit 3A by writing a first multibit signal to a first communications port configured to provide instantaneous access from the processor core 2 to a first queue register 4A with a bit set or reset by the processor core 2.

In second step 90B, the processor core 2 is receiving a signal from a second physical port, dedicated to instantaneously indicate whether the processor core 2 got a lock circuit 3 or not. Thereby, the processor core 2 is processing the signal as an indication that the processor core 2 got the first lock circuit (3A , 90D).

Alternatively, the processing of the signal 90D may only occur if an evaluation indicates that the processor core 2 is dedicated to the first lock circuit (3A, 90C). If the evaluation indicates that the processor core 2 is not assigned to the first lock circuit 3A, the processor core 2 receives then again

a signal from the second physical port 90B and performs a new evaluation 90C.

Alternatively, the processor core processes the signal 90D and then perform the work required the first lock circuit (3, 90E). Furthermore, the processor core 2 releases the first lock circuit 3A when the work including the resource dedicated to the first lock circuit (3A, 90F) is finalized.

The steps above can be performed sequentially, concurrently, repeatedly, or in a different order, or various steps can be omitted.

Fig. 10 shows an alternatively computer-implemented method 100 of synchronizing access to resources by a processor core 2 of a multi-core processor 26 by means of lock circuits 3X. The method comprises a first step 100 A, wherein the processor core 2 enqueues for a first lock circuit 3A by writing a first multibit signal to a first communications port configured to provide instantaneous access from the processor core 2 to a respective first queue register 4A with a bit set or reset by the processor core 2.

Then, in second step 100B the processor core 2 receives a signal from a second physical port, dedicated to instantaneously indicate whether the processor core 2 got a lock circuit 3 or not. Thereby, the processor core 2 is processing the signal as an indication that the processor core 2 got the first lock circuit (3A,100D).

Alternatively, the processing of the signal 100D may only occur if an evaluation indicates that the processor core 2 is dedicated to the first lock circuit (3A, 100C).

The processor core 2 is not assigned to the first lock circuit 3A, the processor core 2 receives then again a signal from the second physical port 100B and performs a new evaluation 100C. Then, the step of enqueuing for the second lock circuit (3B,100E) is performed on the condition that the processor core 2 got the first lock circuit 3A.

In step 100E, the processor core 2 further enqueues for the second lock circuit 3B by writing a second multibit signal to the first physical communications port of the processor core 2 connected for instantaneous access to a second queue register 4B with a bit set or reset by the processor core 2.

In step 100F, the processor core 2 receives again a signal from the second physical single-bit port and processing the signal as an indication that the processor core 2 got the second lock circuit 3B.

Alternatively, the processor core processes the signal 100F and then performing the work required the first lock circuit 3A and the second lock circuit (3B, 100G). Furthermore, the processor core 2 releases the first lock circuit 3A and the second lock circuit 3B, in a random or systematic order, when finalized the work including the resources dedicated to both lock circuits (3A, 3B, 100H).

The steps above can be performed sequentially, concurrently, repeatedly, or in a different order, or various steps can be omitted.

Fig. 11 shows a second alternatively computer-implemented method 200 of synchronizing access to resources by a processor core 2 of a multi-core processor 26 by means of lock circuits 3X and a blocking unit 12, wherein the first step 200A involves enqueuing for a first and at least a second lock circuit (3A, 3B) by writing a first multibit signal and at least a second multibit signal, respectively, to a first physical communication port.

In step 200B, the processor core 2 receives a single bit signal with a single event 28 from a second physical communication port.

In this particular example, the blocking unit 12 is configured to transmit the single bit signal with a ‘high’-event 28 to a second physical communication port of a respective processor core 2 requesting one or more lock circuits 3X. The blocking unit 12 is further configured to transmit a single bit signal with a

single 'low'-event 28 to the second physical communication port of the enqueuing processor core 2 dedicating the processor core 2 to one or more lock circuits 3X.

In step 200C, the processor core 2 evaluates whether it got the first and the second lock circuit (3A, 3B). Step 200B is repeated if the processor core 2 is not dedicated to both lock circuits (3A, 3B), i.e. the received single bit signal is configured with a 'high'-event.

The processor core 2 receives the single bit signal with a 'low'-event and begins processing the signal as an indication that the processor core (2) got the second lock circuit (3B, 200D). The work requiring both lock circuits (3A, 3B, 200E) is executed by the processor core (2,200E). After finalizing the work both lock circuits (3A, 3B) are released by the respective processor core (2, 200F).

The steps above can be performed sequentially, concurrently, repeatedly, or in a different order, or various steps can be omitted.

Fig. 12 shows a third alternatively computer-implemented method 300 of synchronizing access to resources by a processor core 2 of a multi-core processor 26 by means of lock circuits 3X and a blocking unit 12, wherein the first step 300A involves enqueuing for a first lock circuit 3 by writing a first multibit signal to a first physical communication port.

In step 300B the processor core 2 receives a signal from a second physical port, dedicated to instantaneously indicate whether the processor core 2 got the first lock circuit 3A or not.

In step 300C, the processor core 2 evaluates whether it got the first lock circuit (3A).

The processing of the signal 300D may only occur if an evaluation indicates that the processor core 2 is dedicated to the first lock circuit (3A, 300C). The processor core 2 is not assigned to the first lock circuit 3A and may then

request a second lock circuit 3B by writing a second multibit signal to the first physical communication port 300H.

The processor core 2 is dedicated to the first lock circuit (3A, 300C) and may perform the work required by the first lock circuit (3A, 300E). The processor core 2 releases the first lock circuit 3A after finalizing the work 300F. The processor core 2 may decide to request a second lock circuit (3B, 300G), by this means the processor core 2 enqueues for a second lock circuit 3B by writing a first multibit signal to a first physical communication port 300H.

The processor core 2 is enqueued for the second lock circuit 3B, and thereby, the processor core 2 receives a signal from a second physical port, dedicated to instantaneously indicate whether the processor core 2 got the second lock circuit 3 or not 300I.

In step 300J, the processor core 2 evaluates whether it got the second lock circuit 3B or not.

The processing of the signal 300K may occur if the processor core 2 is dedicated to the second lock circuit (3B, 300J).

The processor core 2 may execute the work required by the second lock circuit (3A, 300E) and releases the second lock circuit (3B, 300F) when finalized the work.

In step 300 N, the processor core 2 may decide to continue to step 300B, and repeat the processes of requesting a new lock circuit 3.

The steps above can be performed sequentially, concurrently, repeatedly, or in a different order, or various steps can be omitted.

CLAIMS

1. An integrated circuit (1); comprising:

multiple processor cores (2X);

a lock circuit (3) that comprises:

a queue register (4) with respective bits set or reset via respective, connections (5A) dedicated to respective processor cores (2X), whereby the queue register (4) identifies those among the multiple processor cores (2X) that are enqueued in the queue register (4);

a current register (6); and

a selector circuit (7), being a priority encoder (11), is configured to select a processor core (2) and identify that processor core (2) by a value (17) in the current register (6); wherein a selected processor core (8) is a prioritized processor core (9) among the cores (2X) that have a bit that is set in the queue register (4); and wherein the processor cores (2X) are connected (5B) to receive a signal from the current register (6).

2. An integrated circuit (1) according to claim 1, wherein the selector circuit (7) is configured to:

- maintain a selected processor core (8) selected while that selected processor core (8) maintains its bit in the queue register (4) set; and

- select a prioritized processor core (9) in response to the event (28) that the selected processor core (8) resets its bit in the queue register (4); wherein the prioritized processor core (9) is a processor core (2) with its bit set in the queue register (4).

3. An integrated circuit (1) according to any of the above claims, wherein the value (17) in the current register (6) is a bit-signal, wherein a 'high'-bit signal or a 'low'-bit signal indicates an unselected or a selected processor core, respectively.

4. An integrated circuit (1) according to any of the above claims, comprising multiple of said lock circuits (3X);

wherein processor cores (2X) connect (5A) to queue registers (4) of respective multiple lock circuits (3X); and

wherein the processor cores (2X) are connected (5B) to receive the values (17) in the current registers (6) of respective lock circuits (3X).

5. An integrated circuit (1) according to any of the above claims, comprising:

multiple of said lock circuits (3X); and

a blocking circuit (12) with multiple input ports (12A) connected via respective multi-bit connections (14) to a current register (6) of a respective lock circuit (3), and multiple output ports (12B) connected via respective connections (5B) dedicated to respective processor cores (2X) such that a processor core (2) is connected to receive a signal from a current register (6) in response to the processor core (8) being selected by that current register (6);

wherein the blocking circuit (12) encodes the signal to a predefined processor core with a transition (15) in response to the value (17) of any of the current registers (6X) indicating the identity of that predefined core.

6. An integrated circuit (1) according to claim 5, wherein the respective connections (5B) from the multiple output ports (12B) to respective processor cores (2X) are single-wire connections.

7. An integrated circuit (1) according to any of the above claims, comprising:

one or more of said lock circuits (3X); wherein the processor cores (2X) are configured with a processor identification value and are connected to access the value (17) of one or more current registers (6X).

8. An integrated circuit (1) according to any of the above claims, comprising:

multiple of said lock circuits (3X); and

wherein multiple processor cores (2X) are connected to respective multiple decoder circuits (18X); and

wherein the decoder circuit (18) is connected to multiple lock circuits (3X), such that a processor core (2) can address a predefined lock circuit (3X) by means of an encoded multibit signal (20); and

wherein the decoder circuit (18) is configured to set or reset a bit in the queue register (4).

9. An integrated circuit (1) according to any of the above claims, comprising a multibit communications bus (20) through which the processor cores (2X) communicate with off-chip components (21) when access to the multibit communications granted; wherein said wired connections 5 dedicated exclusively to respective processing cores are separated from the multibit communications bus (20).

10. A multi-core processor (26) configured with an integrated circuit (1) according to any of the above claims.

11. A silicon die (23) configured with an integrated circuit (1) according to any of the above claims.

12. A computer-implemented method (80) of synchronizing access to resources by a processor core (2) of a multi-core processor (26) by means of a lock circuit (3); the method comprising the following steps:

1) Setting or resetting respective bits in a queue register (4) via respective connections (5A) dedicated to respective processor cores (2X);

2) Identifying, by the queue register (4), that the multiple processor cores (2X) are enqueued in the queue register (4);

3) selecting a processor core (2) by a selector circuit (7), being a priority encoder (11), configured to select the processor core (2) and identify that processor core (2) by a value (17) in the current register (6); wherein a selected processor core (8) is a prioritized processor core (9) among the processor cores (2X) that have a bit that is set in the queue register (4);

4) transmitting a signal from the current register (6) to the respective processor cores (2X).

13. A computer-implemented method (80) according to claim 12, wherein it is evaluated whether the processor core (2) got the first lock circuit (3A) and

wherein the processor core (2) performs the work required at least first lock circuit (3A) and then release the at least first lock circuit (3A).

14. A computer-implemented method (80) according to claim 12, wherein the steps designated 1) and 2) of the method (80) are encoded in machine instructions arranged in a consecutive order.

15. A computer-readable medium encoded with a programme to perform the computer-implemented method (90) as set forth in any of the method claims 12 to 14.

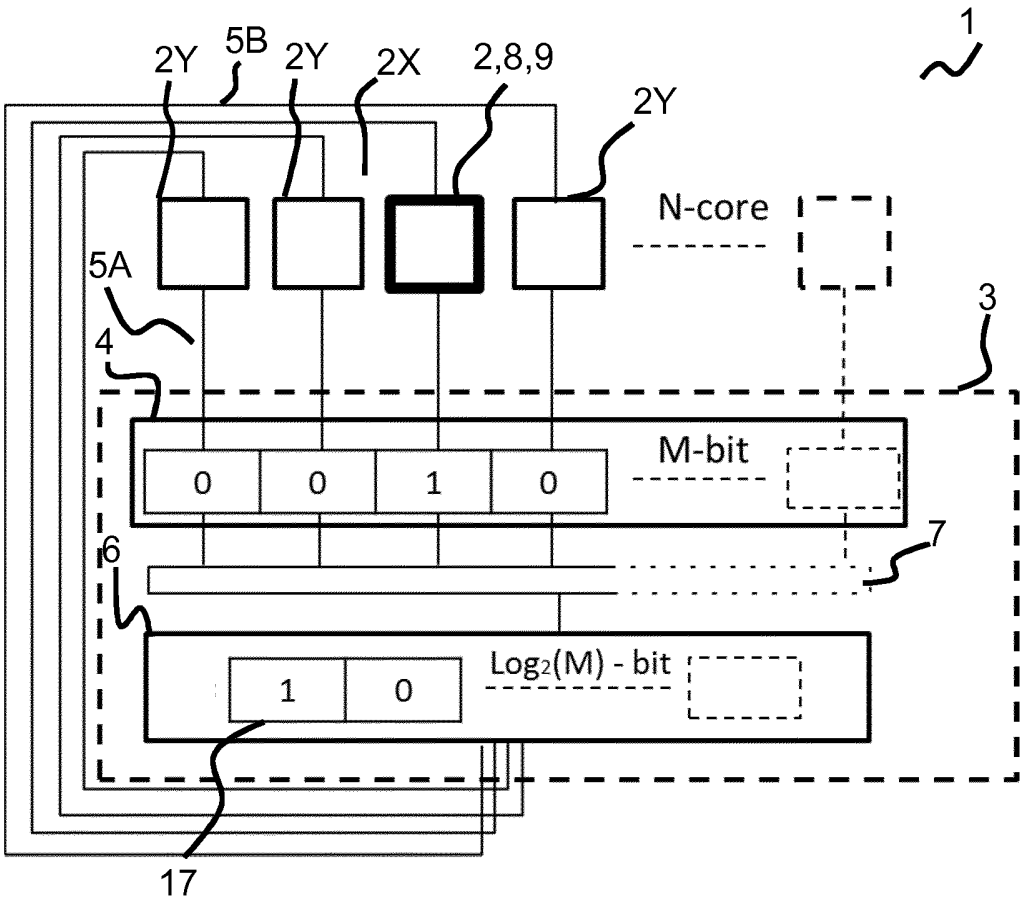

Fig. 1

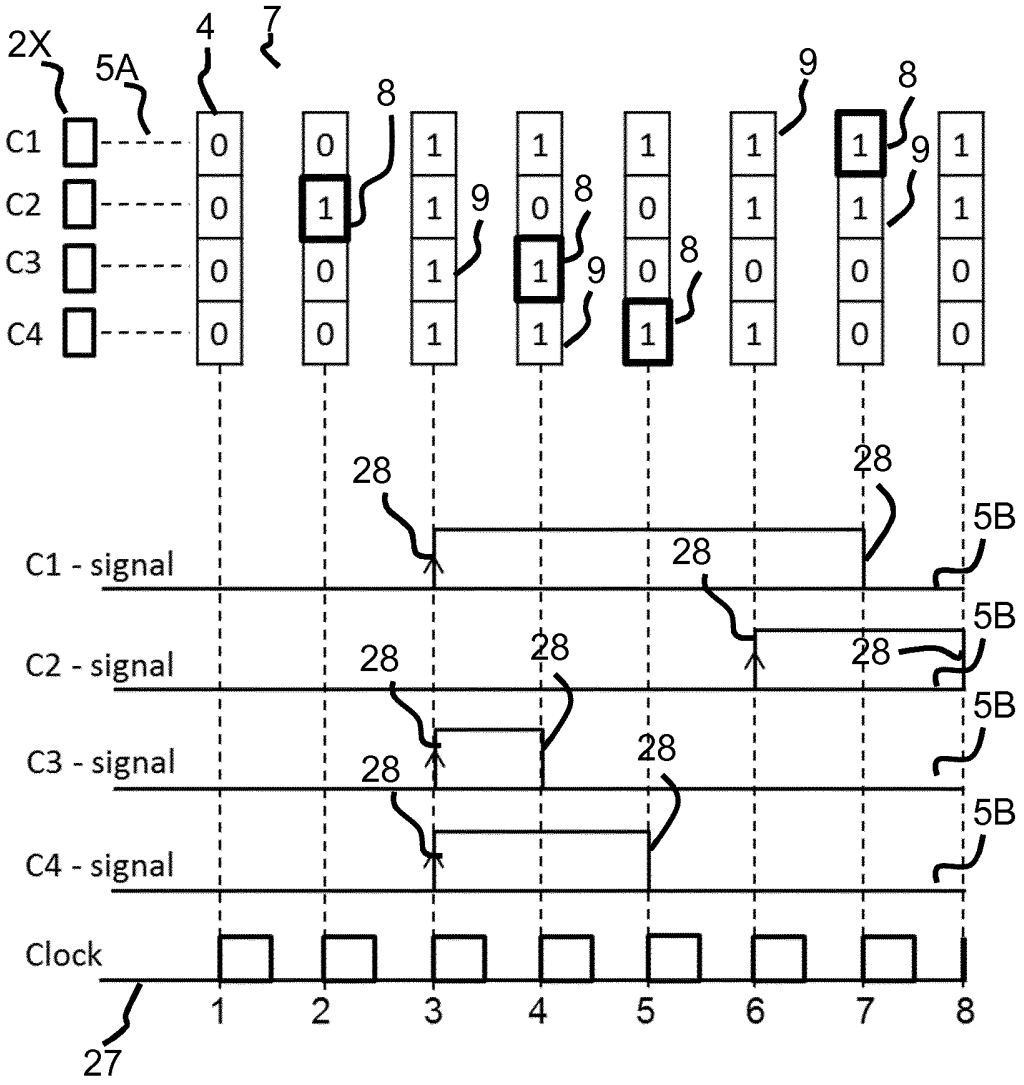
2X
5A
4
7
C1
C2
C3
C4
8
9
28
5B
C1 - signal
C2 - signal
C3 - signal
C4 - signal
Clock
1
2
3
4
5
6
7
8
27

Fig. 2

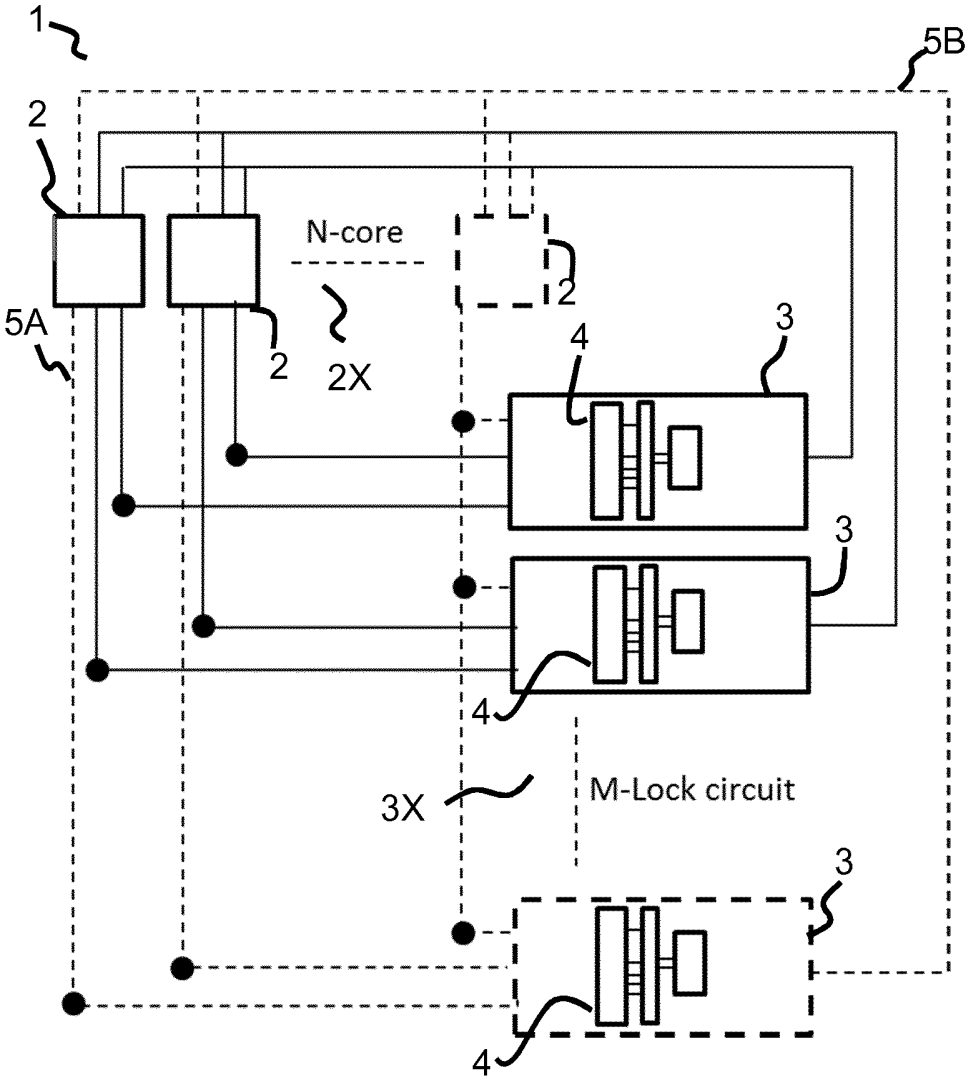

Fig. 3

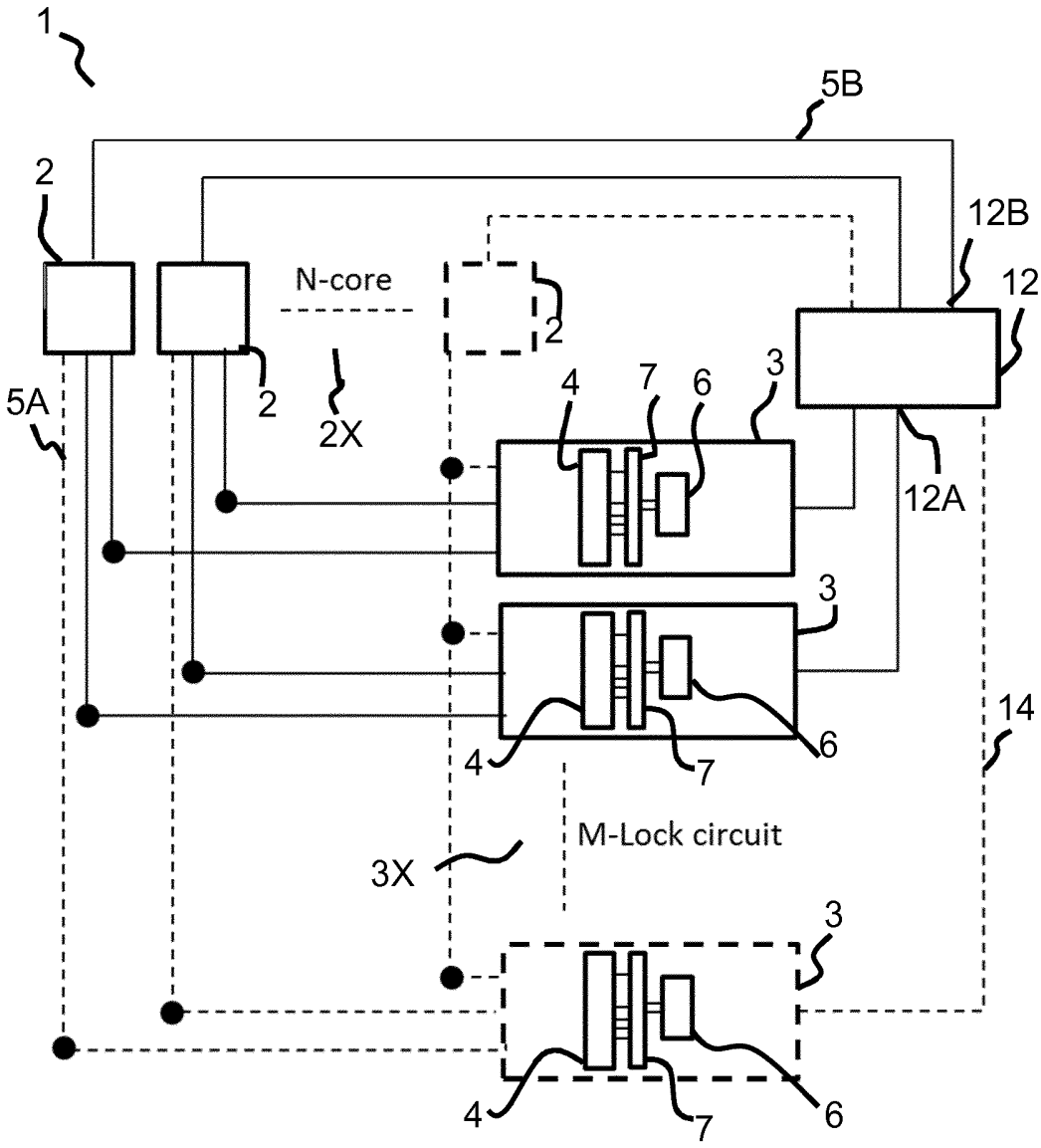
1
5B
2
12B
12
N-core
2
2
2X
5A
4
7
6
3
12A
3
14
4
7
6
M-Lock circuit
3X
3
4
7
6

Fig. 4

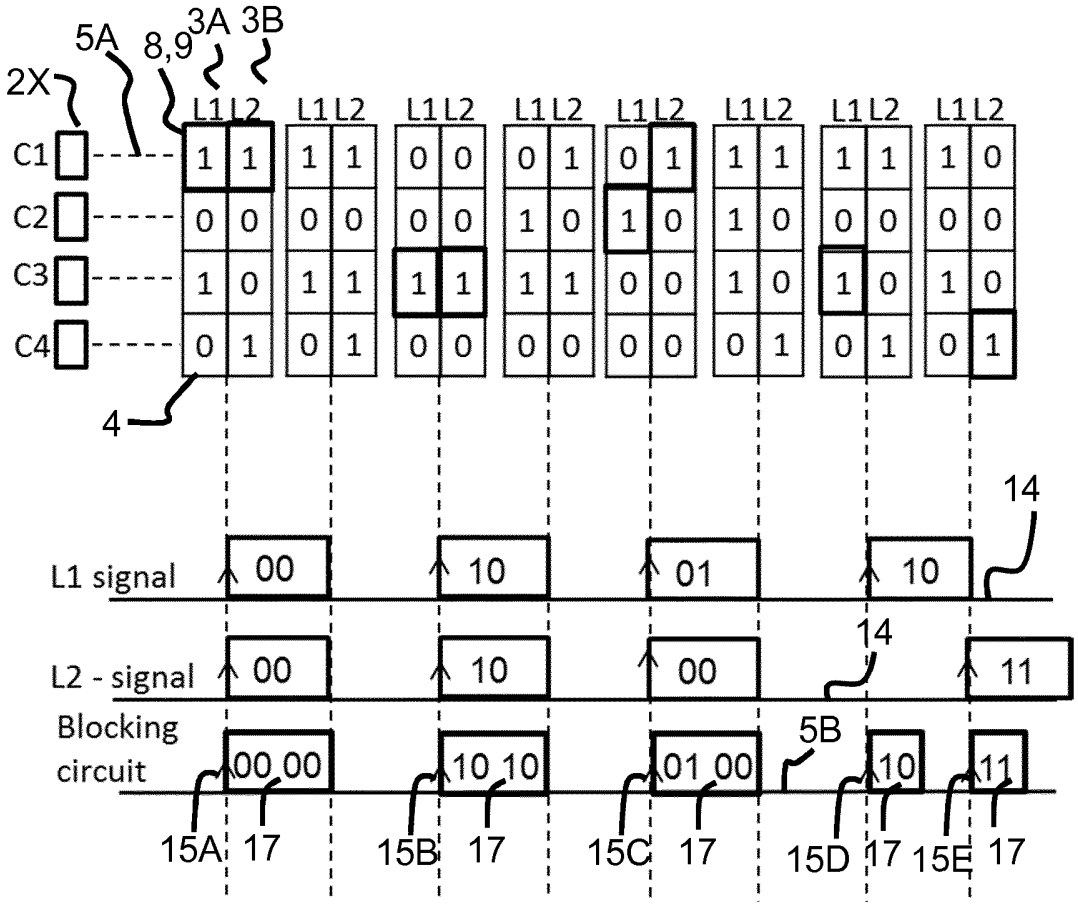
5A
8,9
3A
3B
2X
L1 L2
C1
C2
C3
C4
4
14
L1 signal
00
10
01
10
L2 - signal
00
10
00
14
11
Blocking circuit
00 00
10 10
01 00
5B
10
11
15A
17
15B
17
15C
17
15D
17
15E
17

Fig. 5

6/12

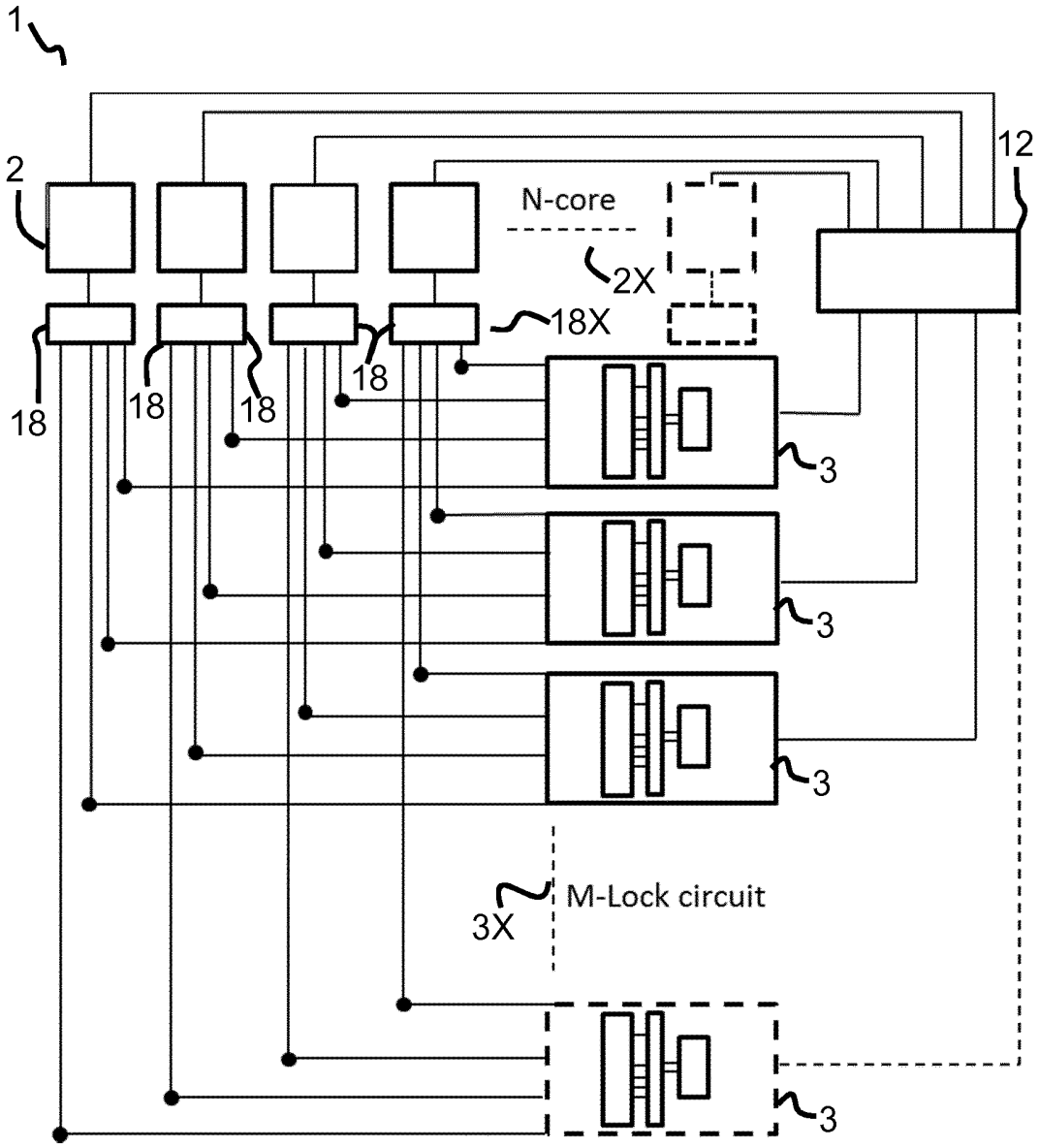

Fig. 6

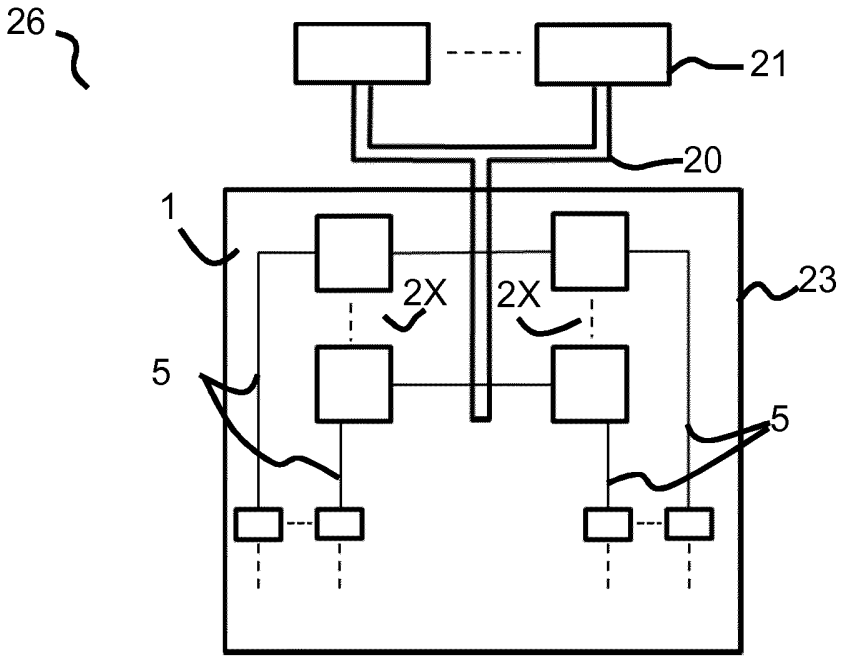

Fig. 7

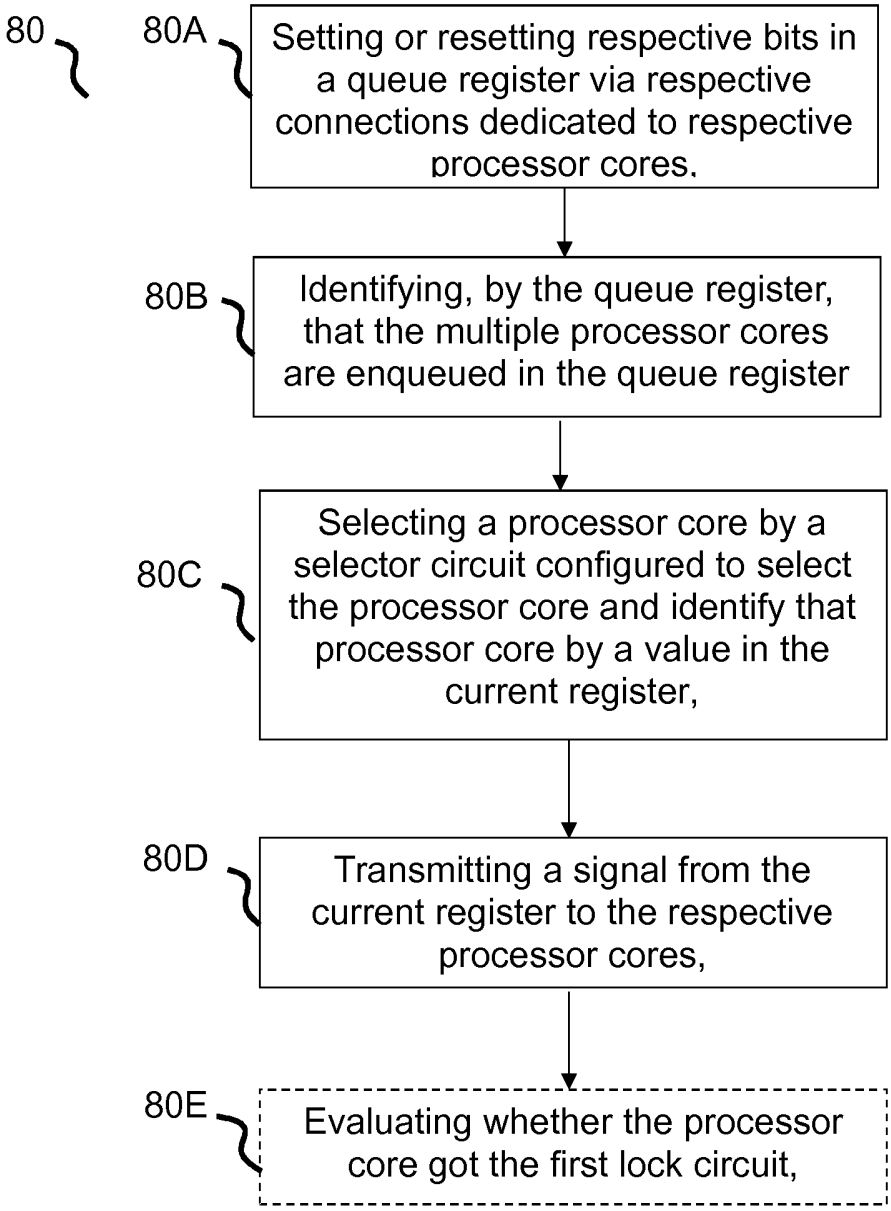
80
80A
Setting or resetting respective bits in a queue register via respective connections dedicated to respective processor cores.
80B
Identifying, by the queue register, that the multiple processor cores are enqueued in the queue register
80C
Selecting a processor core by a selector circuit configured to select the processor core and identify that processor core by a value in the current register,
80D
Transmitting a signal from the current register to the respective processor cores,
80E
Evaluating whether the processor core got the first lock circuit,

Fig. 8

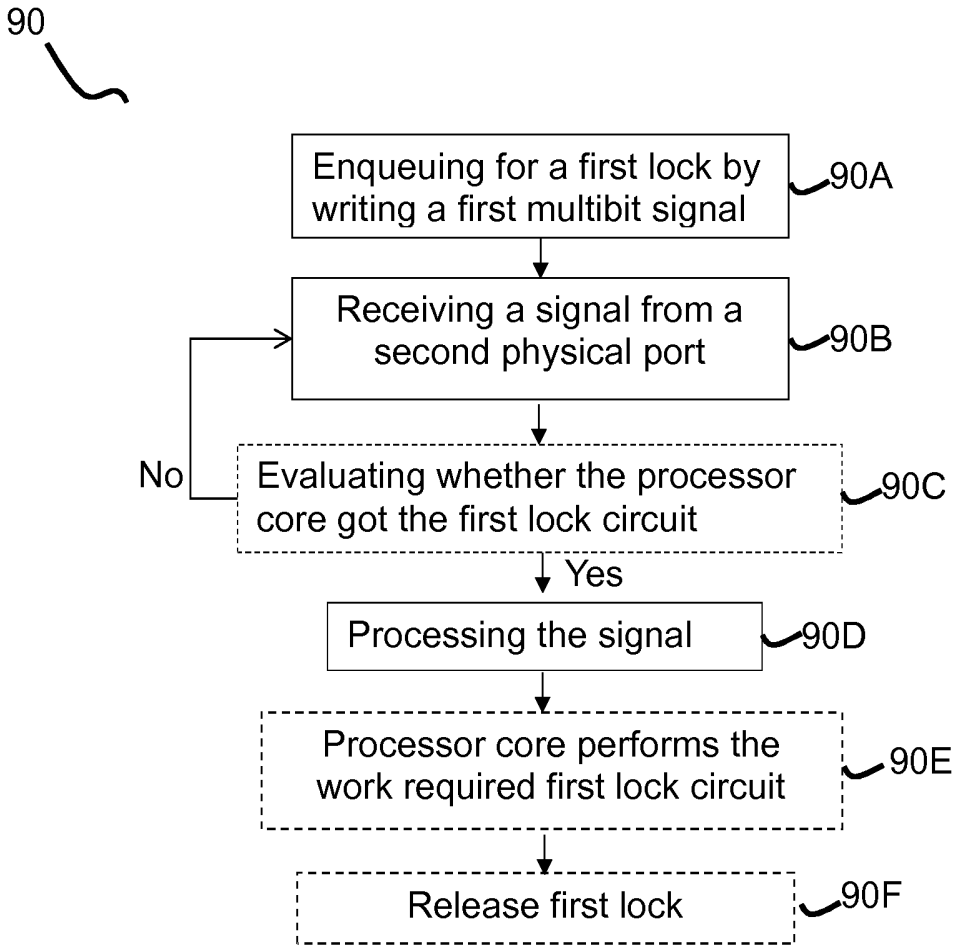
90
Enqueuing for a first lock by writing a first multibit signal
90A
Receiving a signal from a second physical port
90B
No
Evaluating whether the processor core got the first lock circuit
90C
Yes
Processing the signal
90D
Processor core performs the work required first lock circuit
90E
Release first lock
90F

Fig. 9

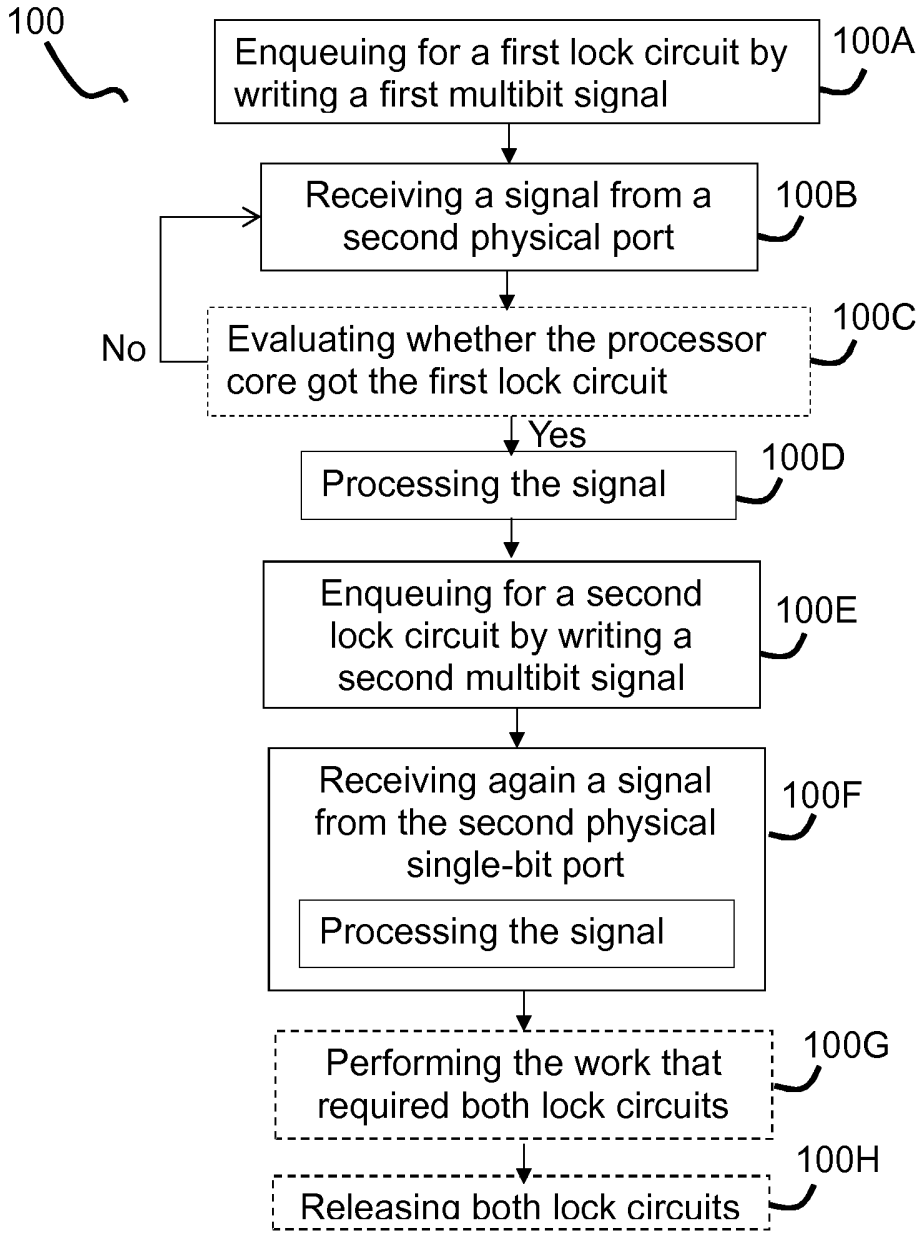
100
Enqueuing for a first lock circuit by writing a first multibit signal
100A
Receiving a signal from a second physical port
100B
No
Evaluating whether the processor core got the first lock circuit
100C
Yes
Processing the signal
100D
Enqueuing for a second lock circuit by writing a second multibit signal
100E
Receiving again a signal from the second physical single-bit port
Processing the signal
100F
Performing the work that required both lock circuits
100G
Releasing both lock circuits
100H

Fig. 10

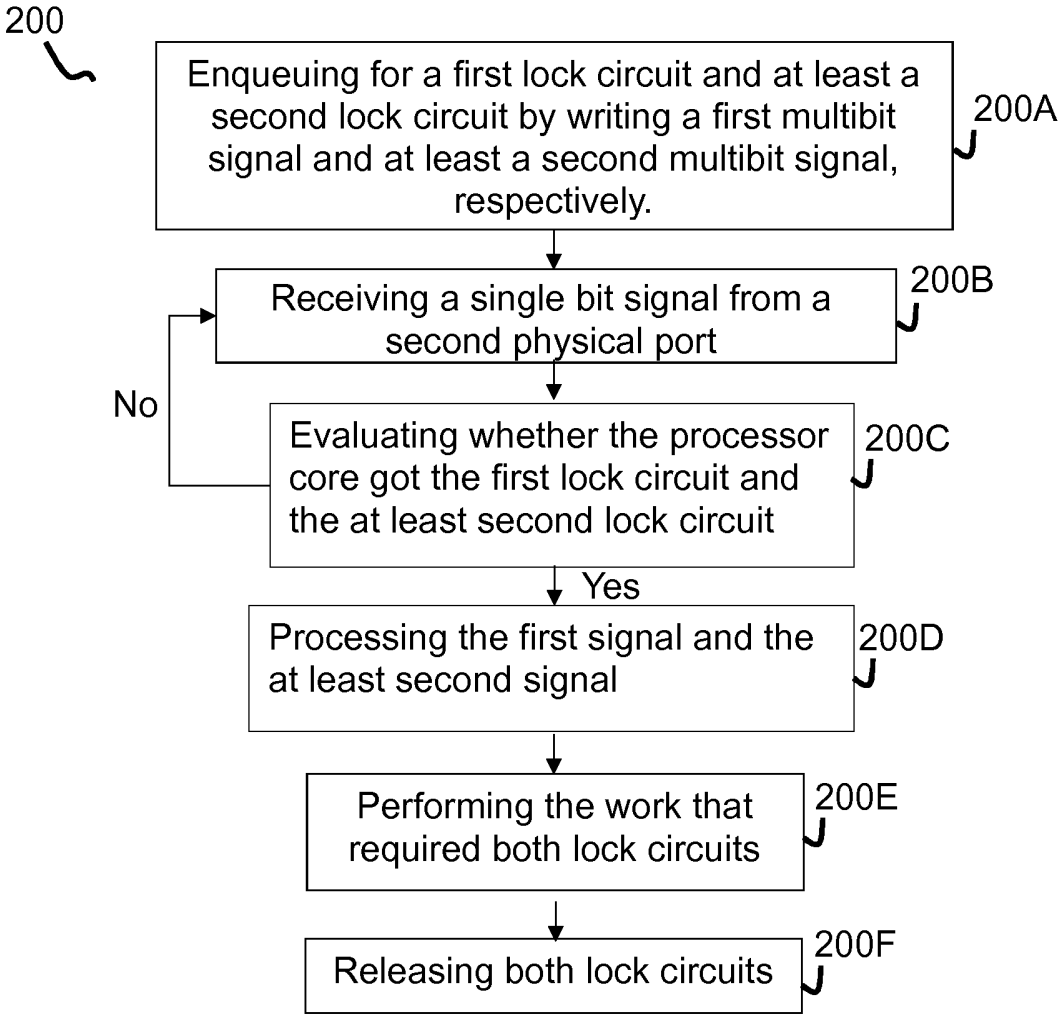
200
Enqueuing for a first lock circuit and at least a second lock circuit by writing a first multibit signal and at least a second multibit signal, respectively.
200A
Receiving a single bit signal from a second physical port
200B
No
Evaluating whether the processor core got the first lock circuit and the at least second lock circuit
200C
Yes
Processing the first signal and the at least second signal
200D
Performing the work that required both lock circuits
200E
Releasing both lock circuits
200F

Fig. 11

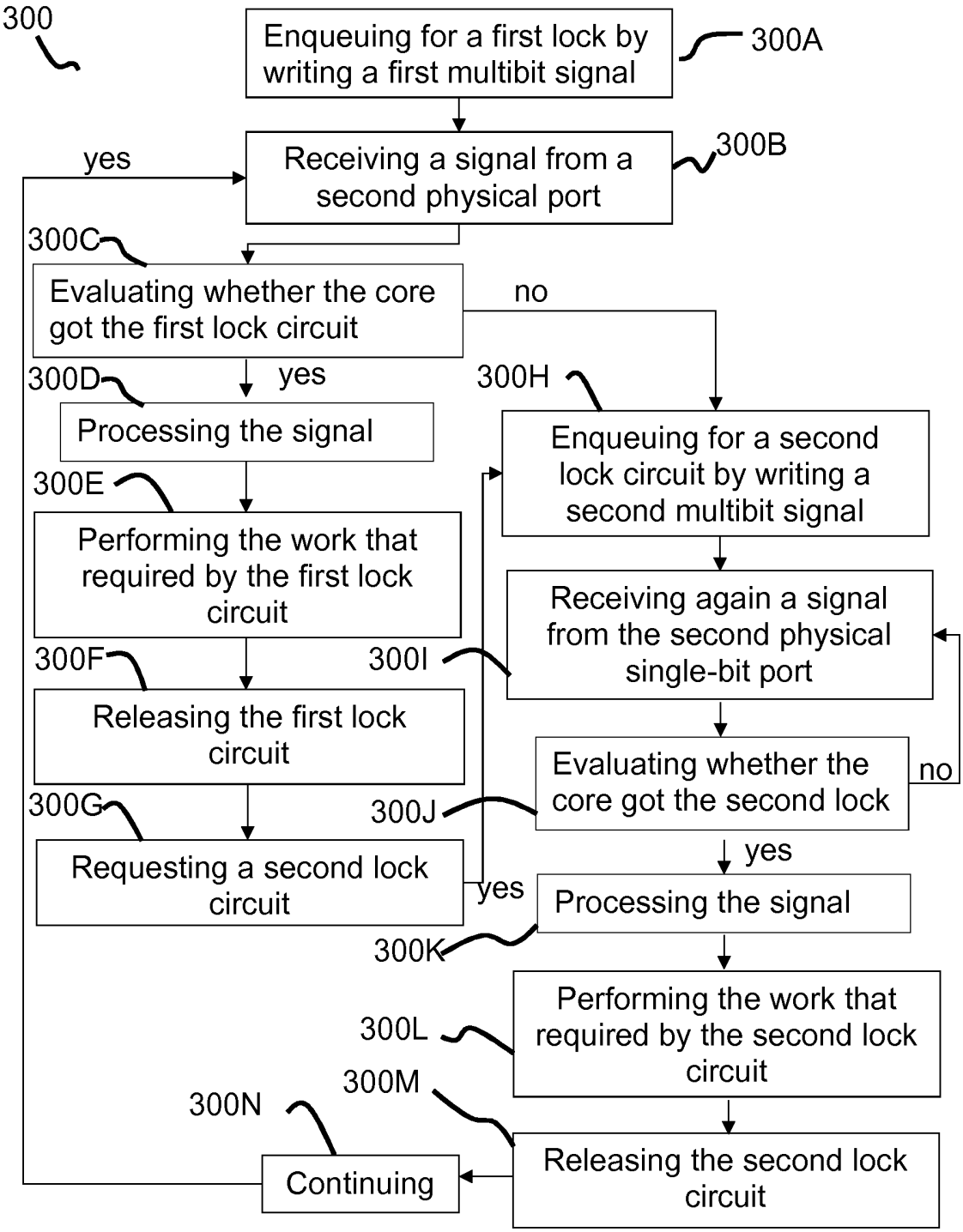
300
Enqueuing for a first lock by writing a first multibit signal
300A
Receiving a signal from a second physical port
300B
yes
300C
Evaluating whether the core got the first lock circuit
no
300D
yes
300H
Processing the signal
Enqueuing for a second lock circuit by writing a second multibit signal
300E
Performing the work that required by the first lock circuit
Receiving again a signal from the second physical single-bit port
300F
300I
Releasing the first lock circuit
Evaluating whether the core got the second lock
no
300G
300J
Requesting a second lock circuit
yes
yes
Processing the signal
300K
Performing the work that required by the second lock circuit
300L
300M
300N
Releasing the second lock circuit
Continuing

Fig. 12

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2015/054491

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F9/48 G06F15/173 G06F9/52
ADD.

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

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

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, IBM-TDB, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 7 571 270 B1 (NEMIROVSKY MARIO [US] ET AL) 4 August 2009 (2009-08-04) column 3, line 8 - line 20 column 3, line 34 - line 64 column 4, line 5 - line 50 column 7, line 32 - line 44 column 8, line 15 - line 2 column 9, line 52 - line 56	1-15
X	US 2006/143416 A1 (KAMIGATA TERUHIKO [JP] ET AL) 29 June 2006 (2006-06-29) paragraphs [0039] - [0056]	1-15
A	EP 0 086 601 A2 (TOKYO SHIBAURA ELECTRIC CO [JP]) 24 August 1983 (1983-08-24) 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

"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
15 June 2015	22/06/2015

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Alecu, Mihail

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2015/054491

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 7571270 B1	04-08-2009	NONE	
US 2006143416 A1	29-06-2006	JP 2006185348 A	13-07-2006
		US 2006143416 A1	29-06-2006
EP 0086601 A2	24-08-1983	DE 3376699 D1	23-06-1988
		EP 0086601 A2	24-08-1983
		JP S58140862 A	20-08-1983
		US 4621318 A	04-11-1986