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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
12 March 2015 (12.03.2015)

WIPO | PCT

(10) International Publication Number
WO 2015/035341 A1

(51) International Patent Classification:
G06F 9/318 (2006.01)

(21) International Application Number:
PCT/US2014/054621

(22) International Filing Date:
8 September 2014 (08.09.2014)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/874,874 6 September 2013 (06.09.2013) US

(71) Applicant *(for all designated States except US)*: **HUAWEI TECHNOLOGIES CO., LTD.** [CN/CN]; Huawei Administration Building, Bantian, Longgang District, Guangdong, Shenzhen, Guangdong 518129 (CN).

(71) Applicant *(for US only)*: **FUTUREWEI TECHNOLOGIES, INC.** [US/US]; 5340 Legacy Drive, Suite 175, Plano, Texas 75024 (US).

(72) Inventors: **GE, Yiqun**; 944 Fletcher Circle, Kanata, Ontario K2T 0B7 (CA). **SHI, Wuxian**; 944 Fletcher Circle, Kanata, Ontario K2T 0B7 (CA). **ZHANG, Qifan**; 389 Emile-Pomiville Ave, Lachine, Québec H8R 0A3 (CA). **HUANG, Tao**; 296 Goldridge Drive, Kanata, Ontario K2T 1L1 (CA). **TONG, Wen**; 12 Whitestone Drive, Ottawa, Ontario K2C 4A7 (CA).

(74) Agents: **CARLSON, Brian A.** et al.; SLATER & MATSIL, L.L.P., 17950 Preston Rd., Suite 1000, Dallas, Texas 75252 (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).

Published:

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

(54) Title: SYSTEM AND METHOD FOR AN ASYNCHRONOUS PROCESSOR WITH ASSISTED TOKEN

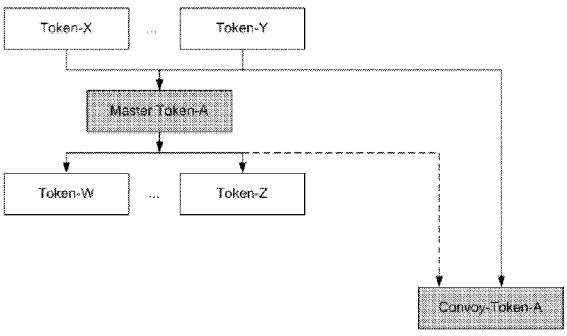

Figure 6

(57) Abstract: Embodiments are provided for an asynchronous processor using master and assisted tokens. In an embodiment, an apparatus for an asynchronous processor comprises a memory to cache a plurality of instructions, a feedback engine to decode the instructions from the memory, and a plurality of XUs coupled to the feedback engine and arranged in a token ring architecture. Each one of the XUs is configured to receive an instruction of the instructions form the feedback engine, and receive a master token associated with a resource and further receive an assisted token for the master token. Upon determining that the assisted token and the master token are received in an abnormal order, the XU is configured to detect an operation status for the instruction in association with the assisted token, and upon determining a needed action in accordance with the operation status and the assisted token, perform the needed action.

WO 2015/035341 A1

System and Method for an Asynchronous Processor with Assisted Token

[0001] This application claims the benefit of U.S. Provisional Application No. 61/874,874 filed on September 6, 2013 by Yiqun Ge et al. and entitled "Method and Apparatus for Asynchronous Processor with Assisted Token," which is hereby incorporated herein by reference as if reproduced in its entirety.

TECHNICAL FIELD

[0002] The present invention relates to asynchronous processing, and, in particular embodiments, to system and method for an asynchronous processor with assisted token.

BACKGROUND

[0003] Micropipeline is a basic component for asynchronous processor design. Important building blocks of the micropipeline include the RENDEZVOUS circuit such as, for example, a chain of Muller-C elements. A Muller-C element can allow data to be passed when the current computing logic stage is finished and the next computing logic stage is ready to start. Instead of using non-standard Muller-C elements to realize the handshaking protocol between two clockless (without using clock timing) computing circuit logics, the asynchronous processors replicate the whole processing block (including all computing logic stages) and use a series of tokens and token rings to simulate the pipeline. Each processing block contains a token processing logic to control the usage of tokens without time or clock synchronization between the computing logic stages. Thus, the processor design is referred to as an asynchronous or clockless processor design. The token ring regulates the access to system resources. The token processing logic accepts, holds, and passes tokens between each other in a sequential manner. When a token is held by a token processing logic, the block can be granted the exclusive access to a resource corresponding to that token, until the token is passed to a next token processing logic in the ring. There is a need for an improved asynchronous processor architecture which can handle instructions and processing resources with more efficiency.

SUMMARY OF THE INVENTION

[0004] In accordance with an embodiment, a method performed by an asynchronous processor includes receiving, at an execution unit (XU), a master token associated with a resource and further receiving an assisted token for the master token. Upon determining that the assisted token and the master token are received in an abnormal order, the XU detects an operation status at the XU in association with the assisted token, and upon determining a needed action in accordance with the operation status and the assisted token, performs the needed action.

[0005] In accordance with another embodiment, a method performed by an asynchronous processor includes performing a branch speculation at an XU, and receiving, at the XU, a program-counter (PC) jump token and a cancel branch speculation assisted token for the PC jump token. Upon determining that the cancel branch speculation assisted token is received in an abnormal order with the PC jump token, the XU detects whether the cancel branch speculation assisted token corresponds to the branch speculation, and upon determining the cancel branch speculation assisted token corresponds to the branch speculation, revokes the cancel branch speculation and fetches a new instruction at the XU.

[0006] In accordance with another embodiment, an apparatus for an asynchronous processor comprises a memory configured to cache a plurality of instructions, and a feedback engine coupled to the memory and configured to decode the instructions from the memory. The apparatus further comprises a plurality of XUs coupled to the feedback engine and arranged in a token ring architecture. Each one of the XUs is configured to receive an instruction of the instructions form the feedback engine, and receive a master token associated with a resource and further receive an assisted token for the master token. Upon determining that the assisted token and the master token are received in an abnormal order, the XU is configured to detect an operation status for the instruction in association with the assisted token, and upon determining a needed action in accordance with the operation status and the assisted token, perform the needed action.

[0007] In accordance with yet another embodiment, an apparatus for an asynchronous processor comprises a memory configured to cache a plurality of instructions, and a feedback engine coupled to the memory and configured to decode the instructions from the memory. The apparatus also comprises a plurality of XUs coupled to the feedback engine and arranged in a token ring architecture. Each one of the XUs is configured to receive an instruction of the instructions form the feedback engine, perform a branch speculation related to the instruction, and receive a PC jump token and a cancel branch speculation assisted token for the PC jump token. Upon determining that the cancel branch speculation assisted token is received in an abnormal order with the PC jump token, the XU is configured to detect whether the cancel branch speculation assisted token corresponds to the branch speculation, and upon determining the cancel branch speculation assisted token corresponds to the branch speculation, revoke the branch speculation and fetch a new instruction.

[0008] The foregoing has outlined rather broadly the features of an embodiment of the present invention in order that the detailed description of the invention that follows may be better understood. Additional features and advantages of embodiments of the invention will be described hereinafter, which form the subject of the claims of the invention. It should be appreciated by those skilled in the art that the conception and specific embodiments disclosed may be readily utilized as a basis for modifying or

designing other structures or processes for carrying out the same purposes of the present invention. It should also be realized by those skilled in the art that such equivalent constructions do not depart from the spirit and scope of the invention as set forth in the appended claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] For a more complete understanding of the present invention, and the advantages thereof, reference is now made to the following descriptions taken in conjunction with the accompanying drawing, in which:

[0010] Figure 1 illustrates a Sutherland asynchronous micropipeline architecture;

[0011] Figure 2 illustrates a token ring architecture;

[0012] Figure 3 illustrates a token ring based control logic;

[0013] Figure 4 illustrates a token gate system;

[0014] Figure 5 illustrates a token processing logic;

[0015] Figure 6 illustrates an embodiment of assisted token gating;

[0016] Figure 7 illustrates an embodiment of an assisted token processing logic;

[0017] Figure 8 illustrates an embodiment of using master and assisted token processing logics;

[0018] Figure 9 illustrates an example of an order recovery scenario in accordance with an embodiment;

[0019] Figure 10 illustrates an example of a flush penalty scenario;

[0020] Figure 11 illustrates another example of a flush penalty scenario;

[0021] Figure 12 illustrates of a taken and non-taken assisted tokens attached to a program-counter (PC) jump token in accordance with an embodiment;

[0022] Figure 13 illustrates an example of revoking incorrect speculation in accordance with an embodiment;

[0023] Figure 14 illustrates an embodiment of a two level branch speculation over speculation;

[0024] Figure 15 illustrates an example of revoking incorrect speculation over speculation in accordance with an embodiment; and

[0025] Figure 16 illustrates an embodiment of a method applying master and assisted tokens in a token based asynchronous processor system.

[0026] Corresponding numerals and symbols in the different figures generally refer to corresponding parts unless otherwise indicated. The figures are drawn to clearly illustrate the relevant aspects of the embodiments and are not necessarily drawn to scale.

DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

[0027] The making and using of the presently preferred embodiments are discussed in detail below. It should be appreciated, however, that the present invention provides many applicable inventive concepts that can be embodied in a wide variety of specific contexts. The specific embodiments discussed are merely illustrative of specific ways to make and use the invention, and do not limit the scope of the invention.

[0028] Figure 1 illustrates a Sutherland asynchronous micropipeline architecture. The Sutherland asynchronous micropipeline architecture is one form of asynchronous micropipeline architecture that uses a handshaking protocol to operate the micropipeline building blocks. The Sutherland asynchronous micropipeline architecture includes a plurality of computing logics linked in sequence via flip-flops or latches. The computing logics are arranged in series and separated by the latches between each two adjacent computing logics. The handshaking protocol is realized by Muller-C elements (labeled C) to control the latches and thus determine whether and when to pass information between the computing logics. This allows for an asynchronous or clockless control of the pipeline without the need for timing signal. A Muller-C element has an output coupled to a respective latch and two inputs coupled to two other adjacent Muller-C elements, as shown. Each signal has one of two states (e.g., 1 and 0, or true and false). The input signals to the Muller-C elements are indicated by A(i), A(i+1), A(i+2), A(i+3) for the backward direction and R(i), R(i+1), R(i+2), R(i+3) for the forward direction, where i, i+1, i+2, i+3 indicate the respective stages in the series. The inputs in the forward direction to Muller-C elements are delayed signals, via delay logic stages The Muller-C element also has a memory that stores the state of its previous output signal to the respective latch. A Muller-C element sends the next output signal according to the input signals and the previous output signal. Specifically, if the two input signals, R and A, to the Muller-C element have different state, then the Muller-C element outputs A to the respective latch. Otherwise, the previous output state is held. The latch passes the signals between the two adjacent computing logics according to the output signal of the respective Muller-C element. The latch has a

memory of the last output signal state. If there is state change in the current output signal to the latch, then the latch allows the information (e.g., one or more processed bits) to pass from the preceding computing logic to the next logic. If there is no change in the state, then the latch blocks the information from passing. This Muller-C element is a non-standard chip component that is not typically supported in function libraries provided by manufacturers for supporting various chip components and logics. Therefore, implementing on a chip the function of the architecture above based on the non-standard Muller-C elements is challenging and not desirable.

[0029] Figure 2 illustrates an example of a token ring architecture which is a suitable alternative to the architecture above in terms of chip implementation. The components of this architecture are supported by standard function libraries for chip implementation. As described above, the Sutherland asynchronous micropipeline architecture requires the handshaking protocol, which is realized by the non-standard Muller-C elements. In order to avoid using Muller-C elements (as in Figure 1), a series of token processing logics are used to control the processing of different computing logics (not shown), such as processing units on a chip (e.g., ALUs) or other functional calculation units, or the access of the computing logics to system resources, such as registers or memory. To cover the long latency of some computing logics, the token processing logic is replicated to several copies and arranged in a series of token processing logics, as shown. Each token processing logic in the series controls the passing of one or more token signals (associated with one or more resources). A token signal passing through the token processing logics in series forms a token ring. The token ring regulates the access of the computing logics (not shown) to the system resource (e.g., memory, register) associated with that token signal. The token processing logics accept, hold, and pass the token signal between each other in a sequential manner. When a token signal is held by a token processing logic, the computing logic associated with that token processing logic is granted the exclusive access to the resource corresponding to that token signal, until the token signal is passed to a next token processing logic in the ring. Holding and passing the token signal concludes the logic's access or use of the corresponding resource, and is referred to herein as consuming the token. Once the token is consumed, it is released by this logic to a subsequent logic in the ring.

[0030] Figure 3 illustrates an example of a token ring based control logic, which is part of the token ring architecture above. According to this logic, the token ring consists of a cascade (series) of token processing logics and an inverter. The token processing logics process a token signal as discussed above. The inverter simply inverts the state (e.g., from 0 to 1, or 1 to 0) of the token signal from the last token processing logic before sending the inverted token signal back to the start of the ring. The token signal is referred to as an edge signal that passes through one token processing logic to the next. Additionally, an

external enable signal controls when to process an incoming token signal at each token processing logic. Further, a pulse or active signal is generated from each token processing logic according to the processing of its incoming token signal, where the token signal can suffer from certain processing latency when going through a token processing logic. The pulse signal is sent from a token processing logic to a corresponding computing logic (not shown) to control, start, or allow the computing logic access to a resource associated with the token signal.

[0031] Figure 4 illustrates a token gate system where releasing each token in the token system gates the processing of a corresponding token in the system at the same ALU (intra-ALU token gate system). Specifically, designated tokens are used to gate other designated tokens in a given order for gating and passing the tokens through the ALUs. The gating process means that when a designated token passes through an ALU, a second designated token is then allowed to be processed and passed by the same ALU in the token ring architecture. In other words, releasing one token by the ALU becomes a condition to consume (process) another token in that ALU in that given order. In this example, the token gating system includes two token rings that pass across the logics (or ALUs): a token ring for passing token-A, and another token ring for passing token-B. Releasing token-B gates (is a condition to allow consuming) token-A for each token processing logic (e.g., ALU). Figure 4 also shows a cascade of tokens in the token gating system according to a predefined gating and passing order of the tokens through the ALUs. For example, token-X and token-Y gate token-A, and token-A gates token-W and token-Z.

[0032] Figure 5 illustrates a token processing logic, as part of the token ring architecture. The token processing logic can be implemented in an arithmetic and logic unit (ALU) of an asynchronous processor. A token processing logic within an ALU can be abstracted into 3 logics: token sense & latch logic, token delay logic, and pulse/active signal generation logic.

[0033] Figure 6 illustrates an embodiment of assisted token gating. In a normal order of token passing, a master token-A is received at an ALU before assisted token-A. In an abnormal order of token passing, the assisted token-A is received at an ALU before the master token-A. The assisted token is gated by the same conditions as its master token. If the assisted token is disabled, it is also gated by its master token. Hence, the assisted token is received at an ALU after its master token (the normal order is kept). If the assisted token is enabled (abnormal order), it is not gated by its master token until a normal order is recovered.

[0034] The token system conveys some information by gating and passing among tokens. Using gating, one operation is finished clearly and a next operation can be started. Using passing, one common resource is accessed clearly and a next request can be sent for it. Tokens are used for avoiding the mesa-stability of the clockless (asynchronous) circuit, avoiding the structural hazard (resource conflict), and

preserving the program order. Conveying more information using the token system and using the tokens for other functionality is useful and desired.

[0035] Figure 7 illustrates an embodiment of an assisted token processing logic. According to this logic, once an assisted token is allowed, it is released immediately. This is implemented to minimize the assisted token latency if it is used. The assisted token does not generate any pulse or active signals. The assisted token also does not indicate any exclusive access to a resource (e.g., a register, buffer, file, predictor), and does not generate any clock-on-demand.

[0036] Figure 8 illustrates an embodiment of using master and assisted token processing logics. An order-detect logic detects an abnormal order, that is, the assisted token, also referred to herein as convoy token, comes earlier than the master token. If an abnormal order is detected, the convoy token is released earlier than the master token (keeping this abnormal order). Further, ALU is informed of this abnormal situation. If a normal order is detected, the convoy token is held until the master token is released (keeping this normal order). Alternatively, if the ALU decides to use this convoy token, the convoy token is passed earlier than the master token (starting an abnormal order).

[0037] Figure 9 illustrates an example of an order recovery scenario in accordance with an embodiment. In a normal order, the master token comes from logic-(1) earlier than the assisted or convoy token. Next, logic-(2) triggers an abnormal order, and sends the convoy order first. The abnormal order (convoy token first) passes to the next logics in the pipeline. The convoy is then passed through the token logics to chase up to the master, by looping back the token ring (a roundabout on the token ring), to return back to normal order.

[0038] Among the benefits of this assisted token processing logic, a token logic-(N) can pass certain information to the next token logics-(N+x), where N and x are integers. Meanwhile, this issuing token logic-(N) does not install itself (it undoes its calculation). In an abstract application scenario, the token logic-N meets a situation and needs to inform some other token-logics of the situation. Hence, the token logic-(N) sends an assisted token (abnormal order), as a protocol to indicate the situation to next logic. Any other token-logics that know this protocol can react properly to this abnormal order.

[0039] Figure 10 illustrates an example of a flush penalty scenario. The flush penalty is implemented to remove a branch speculation. A branch speculation is made before waiting for calculation results of a condition (e.g., if-then-else statement) that determines one of two (or more) possible branches of actions to take. Speculation is performed to save processing time, where the ALU speculates which branch to take prior to waiting for the results and thus knowing the correct branch. the speculation can be made based on historical data, for instance, or other criteria. When the results are obtained, the execution

of the branch speculation is kept if the branch speculation is found to be the correct branch according to the results. Otherwise, the branch speculation is flushed (its calculations and results erased), the correct branch is recalculated. Figure 11 illustrates another example of a flush penalty scenario. In this scenario, another flush may take place after one flush is clearly finished. This is the situation when multiple conditions are nested (e.g., nested if-then-else statements) and the corresponding nested branch speculations need to be flushed when the results that determine the correct nested branches are obtained. The penalty for such nested branch speculations is even greater than for a single branch speculation, e.g., in term of computation time.

[0040] Figure 12 illustrates of a taken and non-taken cancel tokens associated with a program-counter (PC) jump token in accordance with an embodiment. The taken-cancel token and the non-taken cancel token are assisted tokens to the PC jump token which is the master token. If a branch is taken (e.g., path A is taken), then the non-taken cancel token is sent, as a protocol, to next logics to cancel the non-taken branch (e.g., cancel path B). Alternatively, if a branch is not taken (path B is taken), then the taken-cancel token is sent to cancel the taken branch (cancel path A).

[0041] Figure 13 illustrates an example of revoking incorrect speculation in accordance with an embodiment. At ALU-1, a branch instruction is executed. A decision is made, where the non-taken branch is found to be the correct branch. Hence, the ALU sends a taken-cancel assisted token to indicate to next ALUs to cancel the taken branch. The ALU also fetches a new instruction. At the next ALU (ALU-2), the taken branch instruction is executed speculatively (before waiting for calculation results from ALU-1 that determines which correct branch to take), and the abnormal order is detected (the taken-cancel token is received or detected in order before the PC jump token). Since the assisted token is the taken-cancel token and performed the branch speculation is for the taken branch, this causes the ALU to undo its operation and fetch a new instruction. At ALU-3, the taken branch instruction is also executed speculatively, and the abnormal order is detected. The same situation of ALU-2 is repeated at ALU-3, which also causes ALU-3 to undo itself and fetch a new instruction. At ALU-(n-1) and ALU-0, non-taken branch instructions are executed speculatively. These ALUs detects an abnormal order and a surely-correct speculation is made (the correct non-taken branch is executed). Hence, ALU-(n-1) and ALU-0 continue execution and fetch a new instruction. In this scheme, the flush penalty is hidden or reduced in the ring of ALUs by speculation.

[0042] Figure 14 illustrates an embodiment of a two level branch speculation over speculation. In each branch level, an operation similar to the operation shown in Figure 12 takes place. At each of the first branch level and the second branch level in Figure 14, if a branch is taken, then the non-taken cancel assisted token is sent. Alternatively, if a branch is not taken, then the taken-cancel assisted token is sent.

[0043] Figure 15 illustrates an example of revoking incorrect speculation over speculation in accordance with an embodiment. The example shows a two level branch speculation over speculation, but the same principle can be applied to any multiple level branch speculation. At ALU-1, a branch instruction is executed. A decision to cancel at all levels the (nested) taken-branch instructions is made, and the ALU sends a taken-cancel assisted token, and fetches a new instruction. At ALU-2, the taken branch instruction's branch instruction (second level branch) is executed speculatively, and the abnormal order when receiving the taken-cancel token is detected. This causes the ALU to undo itself and fetch a new instruction. At ALU-3, the non-taken branch instruction's branch instruction is executed speculatively. The ALU also detects an abnormal order when receiving the taken-cancel token, but detects that the right speculation (non-taken branch) is made. Thus, ALU-3 continues execution and a decision to cancel all the second-level non-taken branch instructions is made. ALU-3 sends out a non-taken cancel assisted token and fetches a new instruction. At ALU-4, a first-level taken-branch instruction is executed speculatively, a taken-cancel assisted token is received first, and an abnormal order is detected causing the ALU to undo itself and fetch a new instruction. At ALU-5, a first-level taken-branch instruction is executed speculatively, a taken-cancel assisted token is received first, and an abnormal order is detected, causing the ALU to undo itself and fetch a new instruction. At ALU-(n-1), a first-level non-taken branch and second-level taken-branch instruction are executed speculatively. ALU-(n-1) receives first a taken-cancel assisted token, receives next a not-taken-cancel assisted token, and detects an abnormal order, but a right speculation is made (the correct two level branches are executed). ALU-(n-1) continues execution and fetches a new instruction. At ALU-0, a first-level non-taken branch and second-level non-taken branch instruction is executed speculatively. ALU-0 receives a taken-cancel assisted token first, detects an abnormal order, but also detects that the first-level speculation is right. ALU-0 then receives a non-taken cancel assisted token and detects an abnormal order, which causes the ALU to undo itself (undo the second level branch) and fetch a new instruction.

[0044] In the two level branch example above, two assisted tokens, one for indicating each correct branch, are issued and pass through the ALUs in addition to the master token (PC jump). In both revoking example of Figures 14 and 15, the assisted token is sent to indicate to the next logics beforehand the right branch executed at a previous logic and hence avoid, limit or reduce the number of flushes that would be needed in the case of incorrect branch speculations. This is one useful implementation of the assisted token architecture. However, the assisted tokens can be sent for other useful purposes or functions to signal ahead of time to next logics or ALUs that an action is required.

[0045] The system and method embodiments herein are described in the context of an ALU set in the asynchronous processor. The ALUs serve as instruction processing units that perform calculations and

provide results for the corresponding issued instructions. However in other embodiments, the processor may comprise other instruction processing units instead of the ALUs. The instruction units may be referred to sometimes as execution units (XUs) or execution logics, and may have similar, different or additional functions for handling instructions than the ALUs described above. In general, the system and method embodiments described herein can apply to any instruction execution units configured in accordance with the master and assisted token processing logics and suitable to implement the embodiments herein.

[0046] Figure 16 illustrates an embodiment of a method embodiment of a method applying master and assisted tokens in a token based asynchronous processor system. The method can be implemented using the embodiments above of the master and assisted token processing logics, e.g., using logic circuits. At step 1610, an assisted token is sent in an abnormal order (prior to sending its master token) from a token logic (e.g., ALU or XU) to a next token processing logic in the token ring architecture to indicate a needed action from the next logic. For instance, a taken-cancel token or a non-taken cancel token is sent by an ALU before its master token (PC jump token) to revoke branch speculation in the next logic(s). At step 1620, the logic detects if a token assisted token and its master are received in an abnormal order. For instance, the ALU also checks if it received itself master and assisted tokens in abnormal order. The steps 1610 and 1620 can be implemented in any order. At step 1630, if an abnormal order is detected, then the logic determines whether an action is needed. For instance, if a taken-cancel token or a non-taken cancel token is received before a PC jump token, then the ALU checks if the branch speculation matches the cancel token. At step 1640, the logic takes the action if needed and continues forwarding the assisted token and master token in the order received (normal or abnormal order). For, instance, the ALU cancels a taken (path A) branch speculation if the taken cancel token is received in abnormal order, or cancels the non-taken (path B) branch speculation if the non-taken cancel token is received in abnormal order. Otherwise, the ALU continues forwarding the tokens without additional action. In other embodiments, multiple assisted tokens may be used as described above for multiple master tokens of the token system

[0047] While several embodiments have been provided in the present disclosure, it should be understood that the disclosed systems and methods might be embodied in many other specific forms without departing from the spirit or scope of the present disclosure. The present examples are to be considered as illustrative and not restrictive, and the intention is not to be limited to the details given herein. For example, the various elements or components may be combined or integrated in another system or certain features may be omitted, or not implemented.

[0048] In addition, techniques, systems, subsystems, and methods described and illustrated in the various embodiments as discrete or separate may be combined or integrated with other systems, modules,

techniques, or methods without departing from the scope of the present disclosure. Other items shown or discussed as coupled or directly coupled or communicating with each other may be indirectly coupled or communicating through some interface, device, or intermediate component whether electrically, mechanically, or otherwise. Other examples of changes, substitutions, and alterations are ascertainable by one skilled in the art and could be made without departing from the spirit and scope disclosed herein.

WHAT IS CLAIMED IS:

1. A method performed by an asynchronous processor, the method comprising:

receiving, at an execution unit (XU), a master token associated with a resource and further receiving an assisted token for the master token; and

upon determining that the assisted token and the master token are received in an abnormal order, detecting an operation status at the XU in association with the assisted token.

2. The method of claim 1 further comprising upon determining a needed action in accordance with the operation status and the assisted token, performing the needed action.

3. The method of claim 1 further comprising forwarding the assisted token and the master token in the abnormal order to a next XU.

4. The method of claim 1 further comprising upon determining that the assisted token and the master token are received in a normal order, forwarding, to a next XU, the master token and the assisted token in the normal order without additional action.

5. The method of claim 1, wherein the abnormal order corresponds to receiving, at the XU, the assisted token before the master token.

6. The method of claim 1 further comprising:

determining, in accordance with a calculation at the XU, that a second action is needed by a next XU; and

sending the assisted token and the master token in the abnormal order to indicate to the next XU that the second action is needed.

7. The method of claim 1 further comprising:

receiving, at the XU with the master token, a second assisted token for the master token;

upon determining that the second assisted token and the master token are received in an abnormal order, determining a second operation status at the XU in accordance with the assisted token; and

upon determining a second needed action in accordance with the second operation status and the second assisted token, performing the needed action.

8. The method of claim 1, wherein the XU is an arithmetic and logic unit (ALU).

9. The method of claim 1, wherein the XU is one of a plurality of XUs arranged in a token based ring architecture.

10. A method performed by an asynchronous processor, the method comprising:

performing a branch speculation at an execution unit (XU);

receiving, at the XU, a program-counter (PC) jump token and a cancel branch speculation assisted token for the PC jump token; and

upon determining that the cancel branch speculation assisted token is received in an abnormal order with the PC jump token, detecting whether the cancel branch speculation assisted token corresponds to the branch speculation.

11. The method of claim 10 further comprising upon determining the cancel branch speculation assisted token corresponds to the branch speculation, revoking the cancel branch speculation and fetching a new instruction at the XU.

12. The method of claim 10 further comprising forwarding the cancel branch speculation assisted token and the PC jump token to a next XU in same received order.

13. The method of claim 10 further comprising upon determining that the cancel branch speculation assisted token and the PC jump token are received in a normal order, forwarding the cancel branch speculation assisted token and the PC jump token to a next XU in same received order without additional action.

14. The method of claim 10, wherein the abnormal order corresponds to receiving, at the XU, the cancel branch speculation assisted token before the PC jump token.

15. The method of claim 10 further comprising:

determining, in accordance with a calculation at the XU, which branch action to be performed by a next XU; and

sending, to the next XU, the cancel branch speculation assisted token corresponding to another branch speculation that is not needed at the next XU.16. The method of claim 10 further comprising:

performing, at the XU, a second level branch speculation corresponding to the branch speculation;

receiving, at the XU, a cancel second level branch speculation assisted token for the PC jump token;

upon determining that the cancel second level branch speculation assisted token is received in an abnormal order with the PC jump token, detecting whether the cancel second level branch speculation assisted token corresponds to the second level branch speculation; and

upon determining the cancel second level branch speculation assisted token corresponds to the second level branch speculation, revoking the second level branch speculation and fetching a new instruction at the XU.

17. The method of claim 10, wherein the branch speculation is one of a plurality of possible branches to be performed at the XU, and wherein the cancel branch speculation assisted token is one of a plurality of cancel branch speculation assisted token corresponding to the plurality of possible branches.

18. An apparatus for an asynchronous processor comprising:

a memory configured to cache a plurality of instructions;

a feedback engine coupled to the memory and configured to decode the instructions from the memory; and

a plurality of execution units (XUs) coupled to the feedback engine and arranged in a token ring architecture, wherein each one of the XUs is configured to:

receive an instruction of the instructions form the feedback engine;

receive a master token associated with a resource and further receive an assisted token for the master token; and

upon determining that the assisted token and the master token are received in an abnormal order, detect an operation status for the instruction in association with the assisted token.

19. The apparatus of claim 18, wherein each one of the XUs is further configured to, upon determining a needed action in accordance with the operation status and the assisted token, perform the needed action.

20. The apparatus of claim 18, wherein each one of the XUs is further configured to:

determine, in accordance with a calculation of the instruction, that a second action is needed by a next XU of the XUs; and

send the assisted token and the master token in the abnormal order to indicate to the next XU that the second action is needed.

21. The apparatus of claim 18, wherein the XUs are arithmetic and logic units (ALUs).

22. An apparatus for an asynchronous processor comprising:

a memory configured to cache a plurality of instructions;

a feedback engine coupled to the memory and configured to decode the instructions from the memory; and

a plurality of execution units (XUs) coupled to the feedback engine and arranged in a token ring architecture, wherein each one of the XUs is configured to:

receive an instruction of the instructions form the feedback engine;

perform a branch speculation related to the instruction;

receive a program-counter (PC) jump token and a cancel branch speculation assisted token for the PC jump token; and

upon determining that the cancel branch speculation assisted token is received in an abnormal order with the PC jump token, detect whether the cancel branch speculation assisted token corresponds to the branch speculation.

23. The apparatus of claim 22, wherein each one of the XUs is further configured to, upon determining the cancel branch speculation assisted token corresponds to the branch speculation, revoke the branch speculation and fetch a new instruction.

24. The apparatus of claim 22, wherein each one of the XUs is further configured to:

determine, in accordance with a calculation the each one of the XUs, which branch action to be performed by a next XU of the XUs; and

send, to the next XU, the cancel branch speculation assisted token corresponding to another branch speculation that is not needed at the next XU.

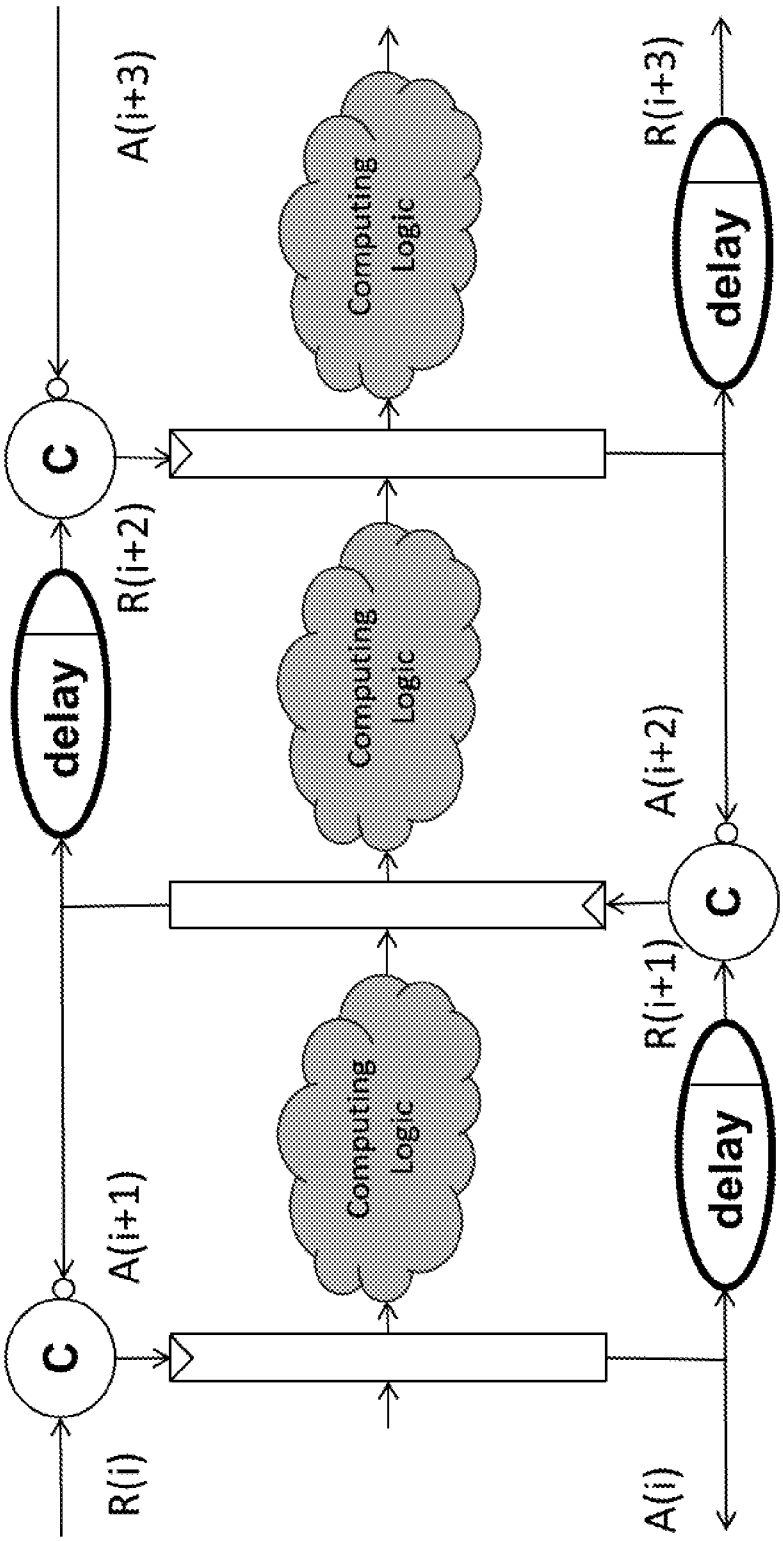
A(i+3)
R(i+3)
C
delay
Computing Logic
R(i+2)
A(i+2)
C
delay
Computing Logic
R(i+1)
A(i+1)
C
delay
Computing Logic
R(i)
A(i)

Figure 1

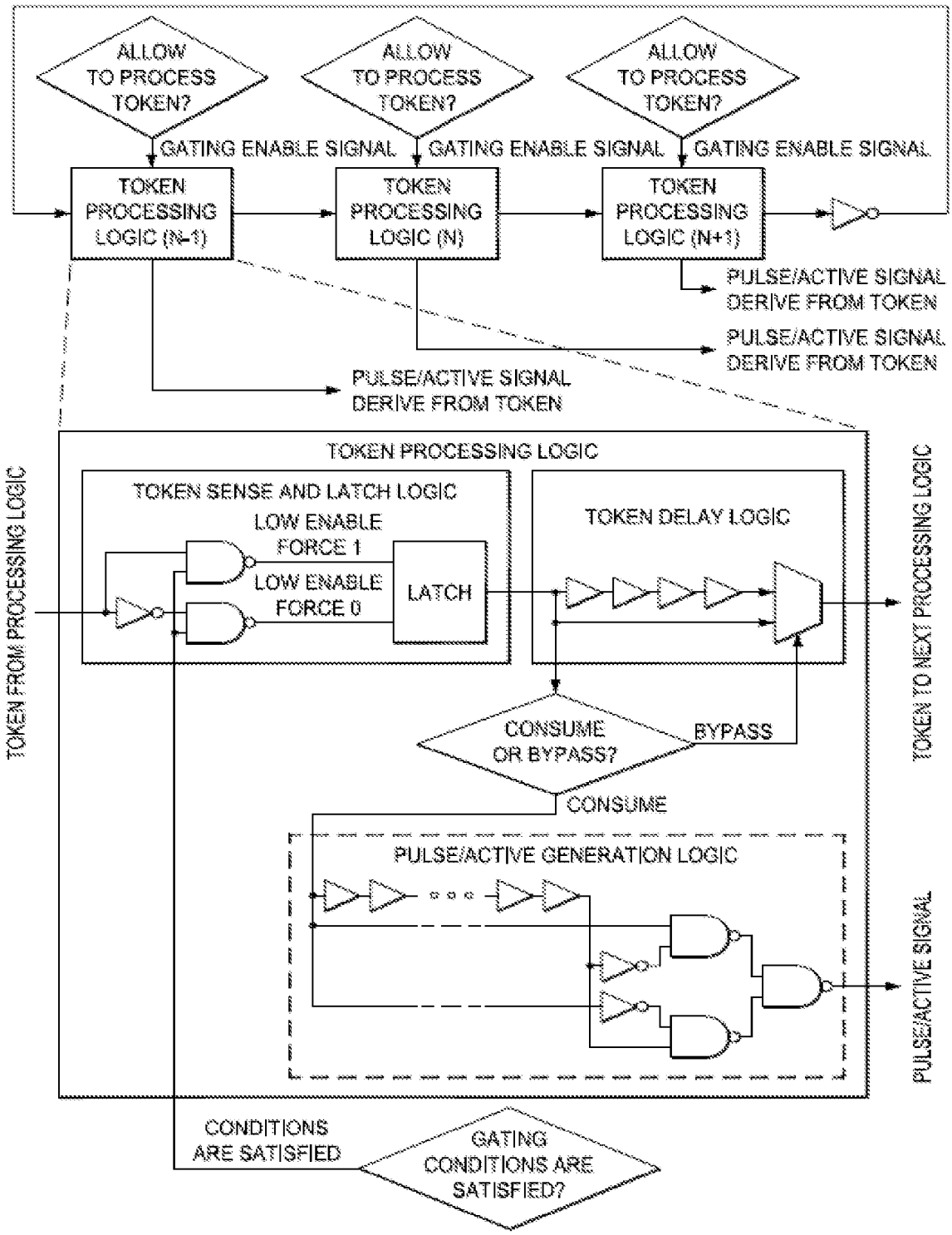
ALLOW TO PROCESS TOKEN?
ALLOW TO PROCESS TOKEN?
ALLOW TO PROCESS TOKEN?
GATING ENABLE SIGNAL
GATING ENABLE SIGNAL
GATING ENABLE SIGNAL
TOKEN PROCESSING LOGIC (N-1)
TOKEN PROCESSING LOGIC (N)
TOKEN PROCESSING LOGIC (N+1)
PULSE/ACTIVE SIGNAL DERIVE FROM TOKEN
PULSE/ACTIVE SIGNAL DERIVE FROM TOKEN
PULSE/ACTIVE SIGNAL DERIVE FROM TOKEN
TOKEN PROCESSING LOGIC
TOKEN SENSE AND LATCH LOGIC
LOW ENABLE FORCE 1
LOW ENABLE FORCE 0
LATCH
TOKEN DELAY LOGIC
TOKEN FROM PROCESSING LOGIC
TOKEN TO NEXT PROCESSING LOGIC
CONSUME OR BYPASS?
BYPASS
CONSUME
PULSE/ACTIVE GENERATION LOGIC
PULSE/ACTIVE SIGNAL
CONDITIONS ARE SATISFIED
GATING CONDITIONS ARE SATISFIED?

Figure 2

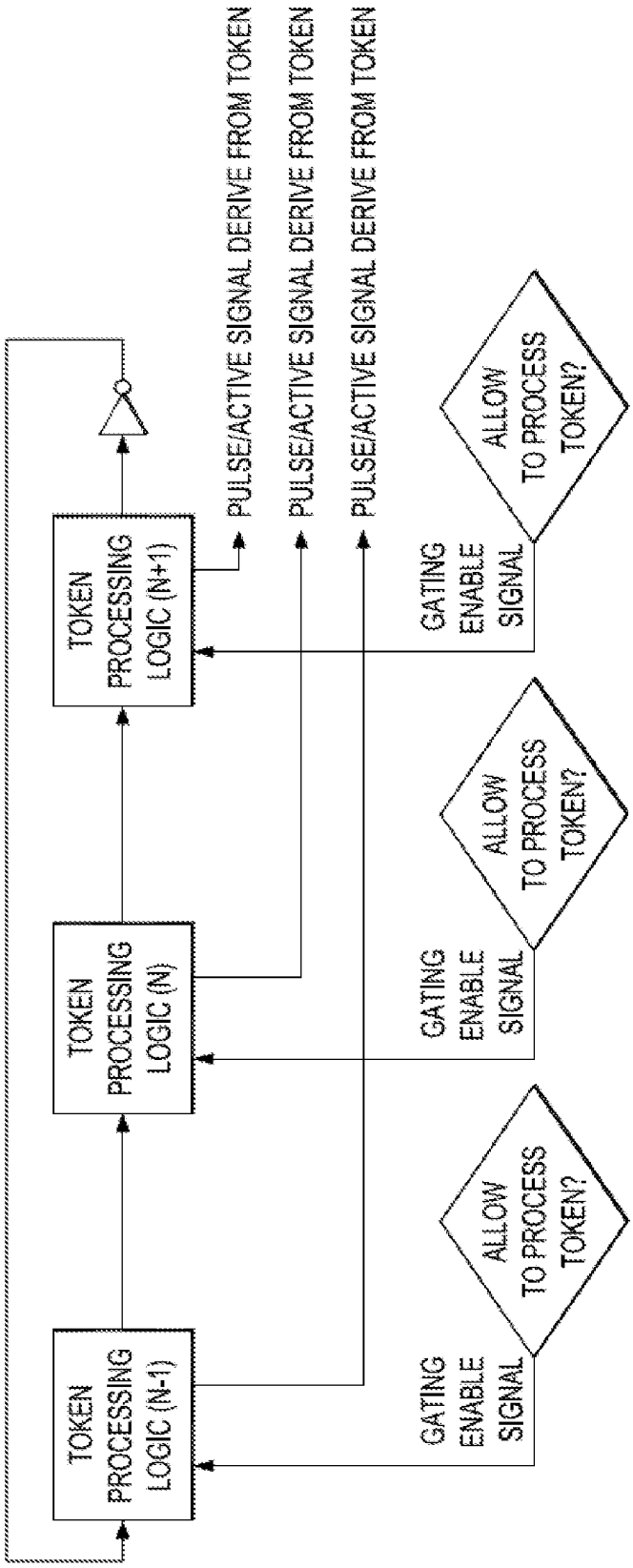
TOKEN PROCESSING LOGIC (N-1)
TOKEN PROCESSING LOGIC (N)
TOKEN PROCESSING LOGIC (N+1)
ALLOW TO PROCESS TOKEN?
ALLOW TO PROCESS TOKEN?
ALLOW TO PROCESS TOKEN?
GATING ENABLE SIGNAL
GATING ENABLE SIGNAL
GATING ENABLE SIGNAL
PULSE/ACTIVE SIGNAL DERIVE FROM TOKEN
PULSE/ACTIVE SIGNAL DERIVE FROM TOKEN
PULSE/ACTIVE SIGNAL DERIVE FROM TOKEN

Figure 3

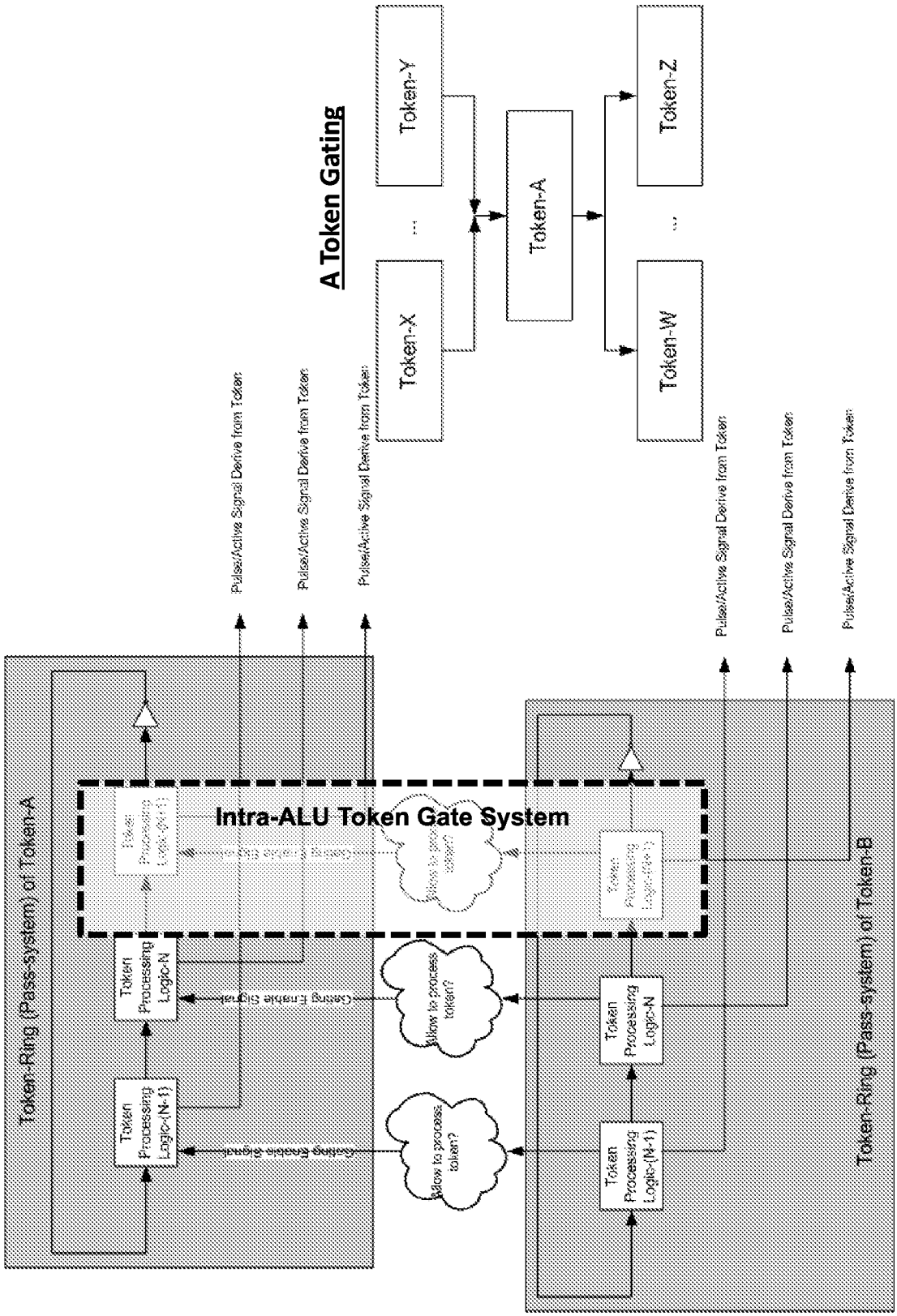
A Token Gating
Token-X
Token-Y
Token-A
Token-W
Token-Z
Pulse/Active Signal Derive from Token
Pulse/Active Signal Derive from Token
Pulse/Active Signal Derive from Token
Pulse/Active Signal Derive from Token
Pulse/Active Signal Derive from Token
Pulse/Active Signal Derive from Token
Intra-ALU Token Gate System
Token Ring (Pass-system) of Token-A
Token Ring (Pass-system) of Token-B
Token Processing Logic-N
Token Processing Logic-(N-1)
Allow to process token?
Allow to process token?
Allow to process token?
Allow to process token?

Figure 4

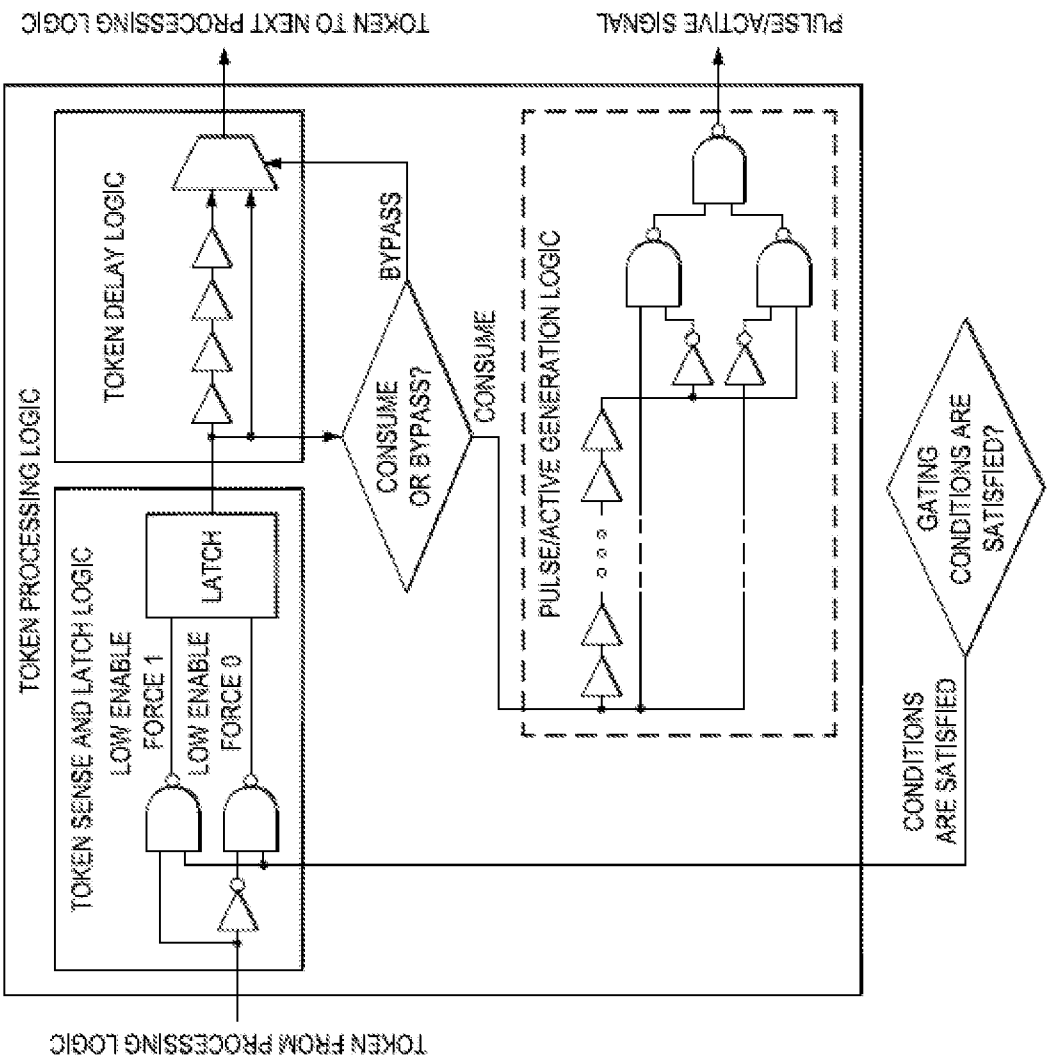
TOKEN TO NEXT PROCESSING LOGIC
PULSE/ACTIVE SIGNAL
TOKEN PROCESSING LOGIC
TOKEN DELAY LOGIC
BYPASS
CONSUME OR BYPASS?
CONSUME
PULSE/ACTIVE GENERATION LOGIC
GATING CONDITIONS ARE SATISFIED?
TOKEN SENSE AND LATCH LOGIC
LOW ENABLE FORCE 1
LOW ENABLE FORCE 0
LATCH
CONDITIONS ARE SATISFIED
TOKEN FROM PROCESSING LOGIC

Figure 5

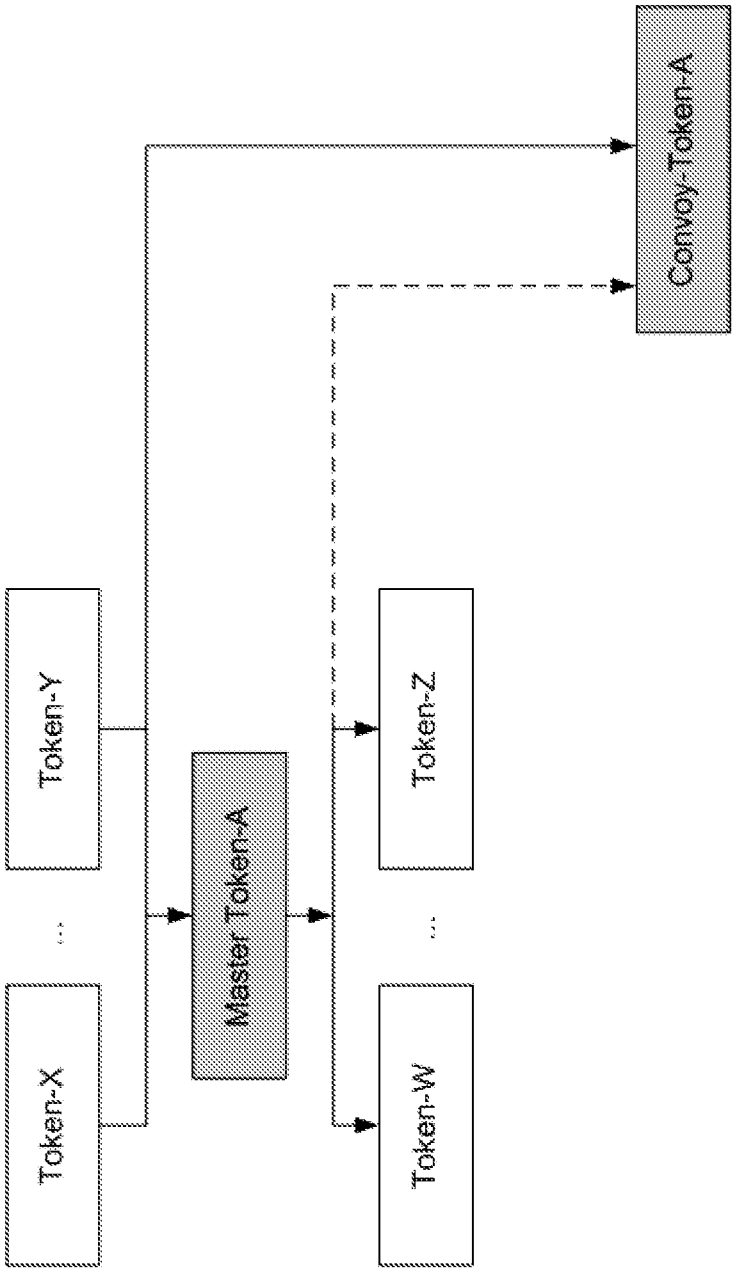

Figure 6

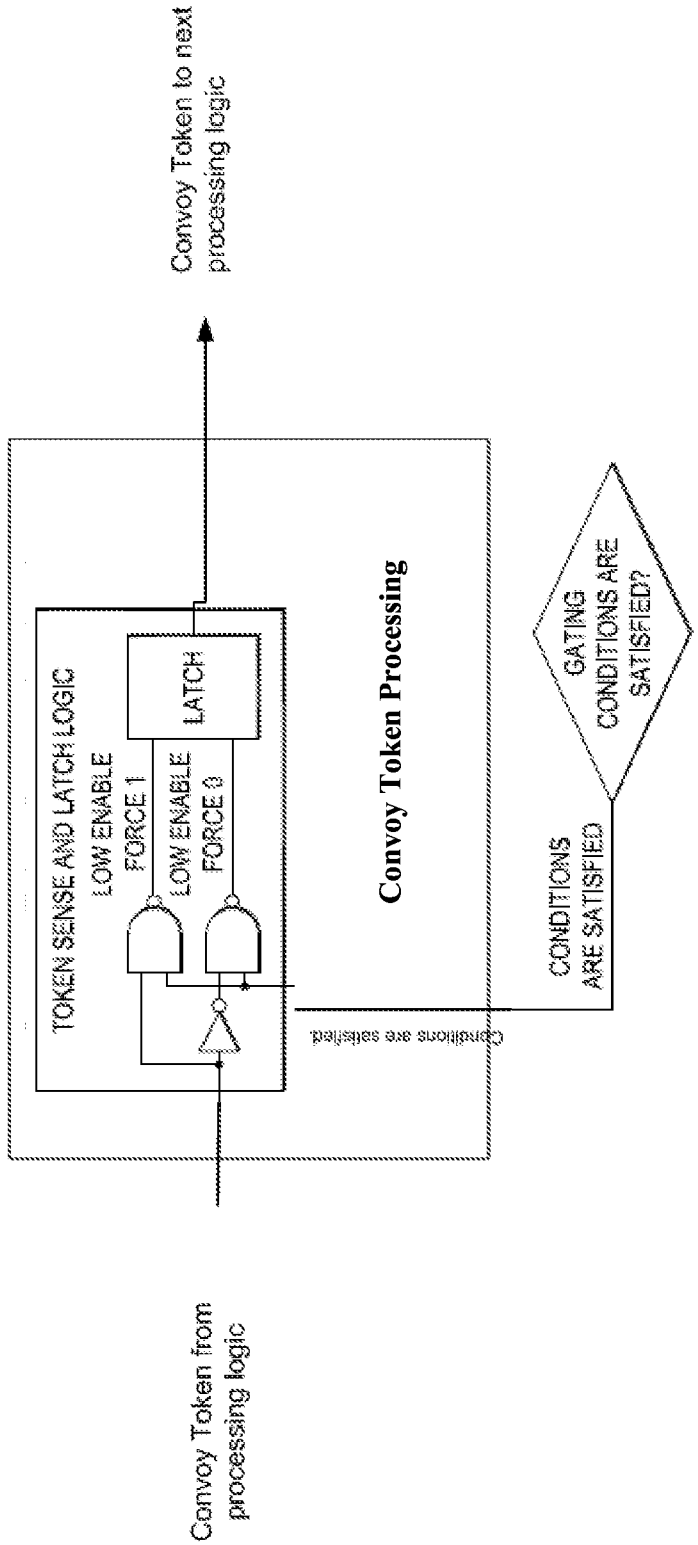
Convoy Token to next processing logic
TOKEN SENSE AND LATCH LOGIC
LOW ENABLE FORCE 1
LOW ENABLE FORCE 0
LATCH
Convoy Token Processing
GATING CONDITIONS ARE SATISFIED?
CONDITIONS ARE SATISFIED
Conditions are satisfied
Convoy Token from processing logic

Figure 7

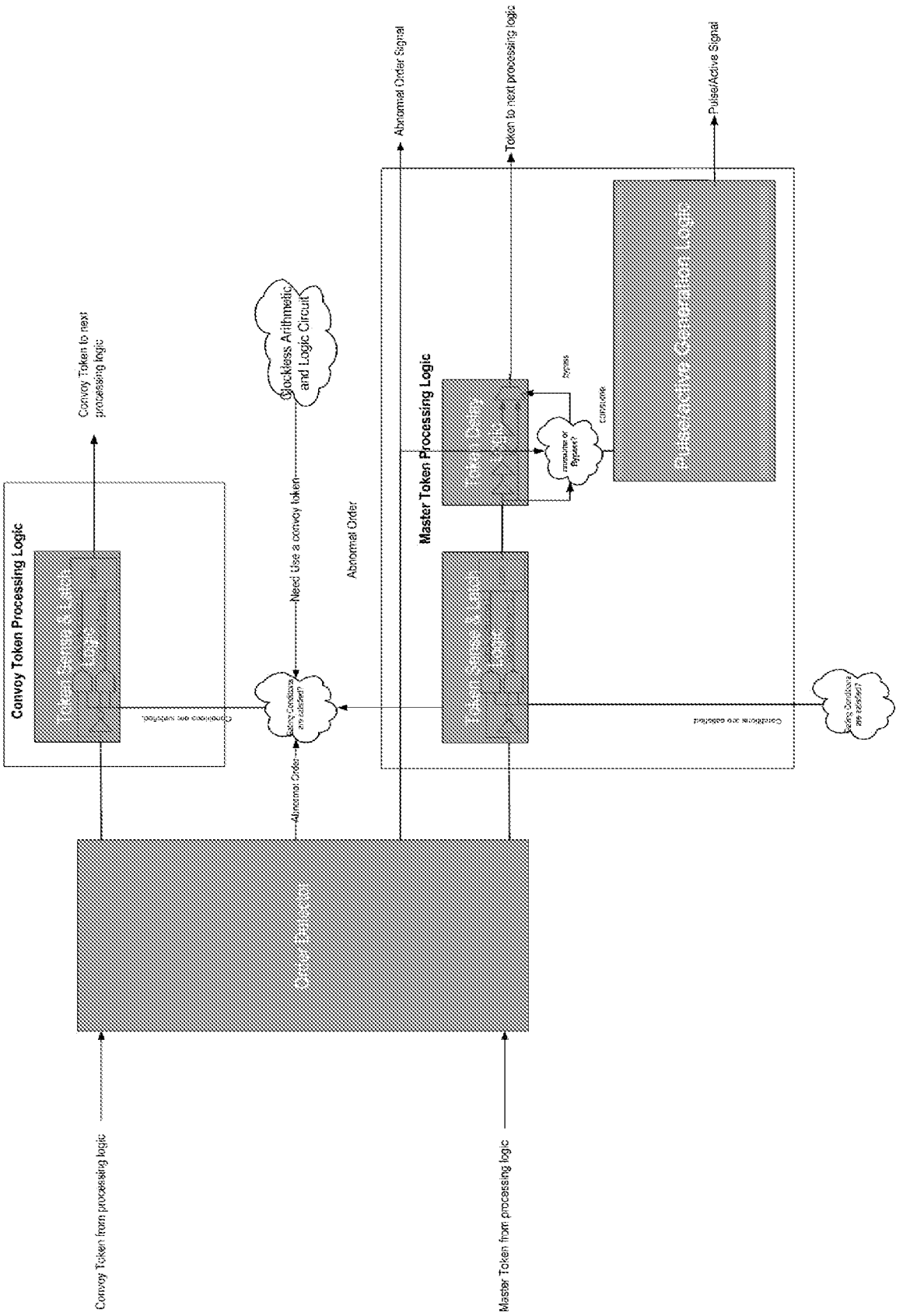
Abnormal Order Signal
Token to next processing logic
Pulse/Active Signal
Clockless Arithmetic and Logic Circuit
Convoy Token to next processing logic
Need use a convoy token
Abnormal Order
Master Token Processing Logic
bypass
consume
consume or Bypass?
Convoy Token Processing Logic
Abnormal Order
Convoy Token from processing logic
Master Token from processing logic

Figure 8

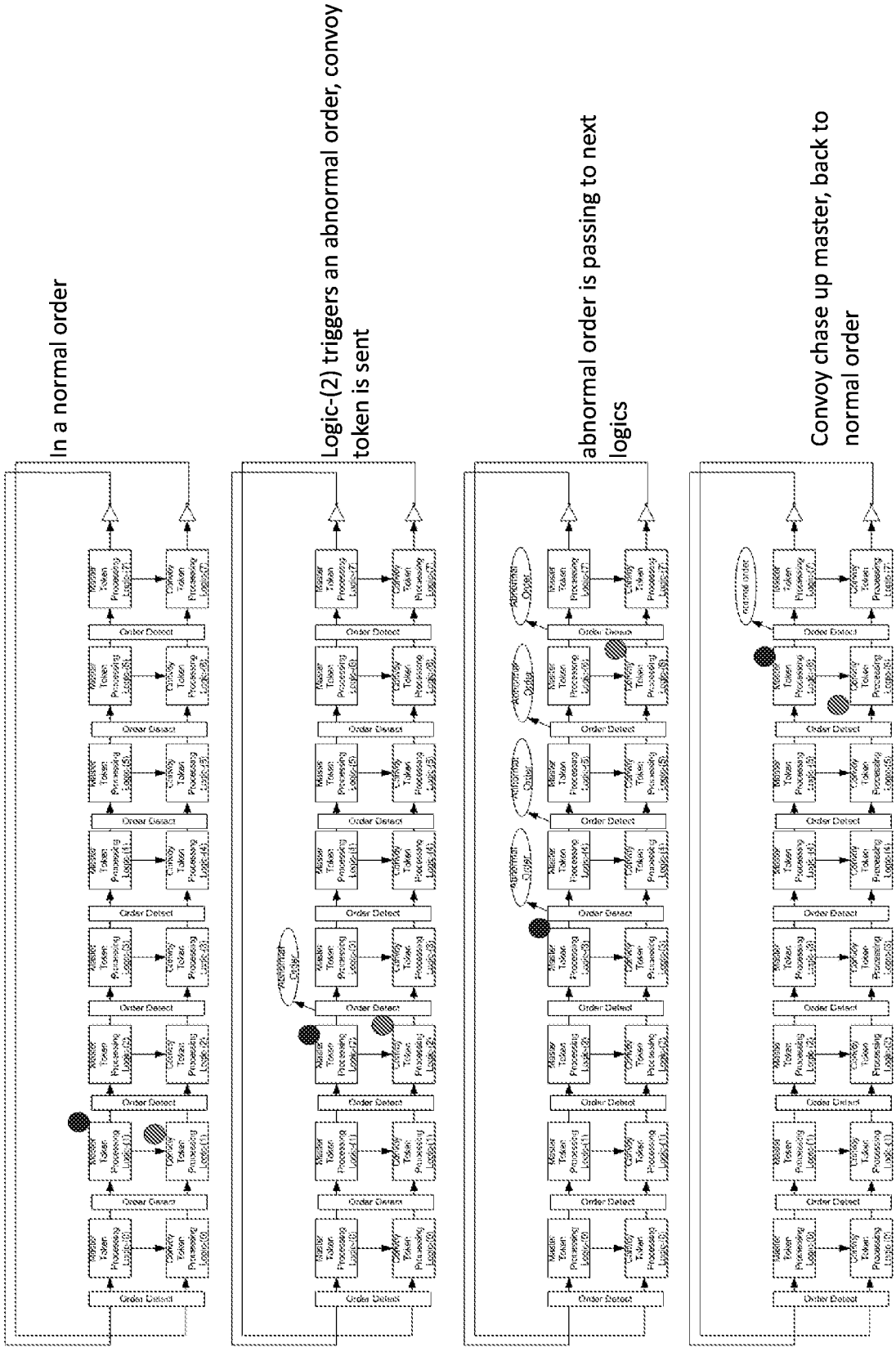
In a normal order
Logic-(2) triggers an abnormal order, convoy token is sent
abnormal order is passing to next logics
Convoy chase up master, back to normal order
Master Token
Convoy Token
Order Detect

Figure 9

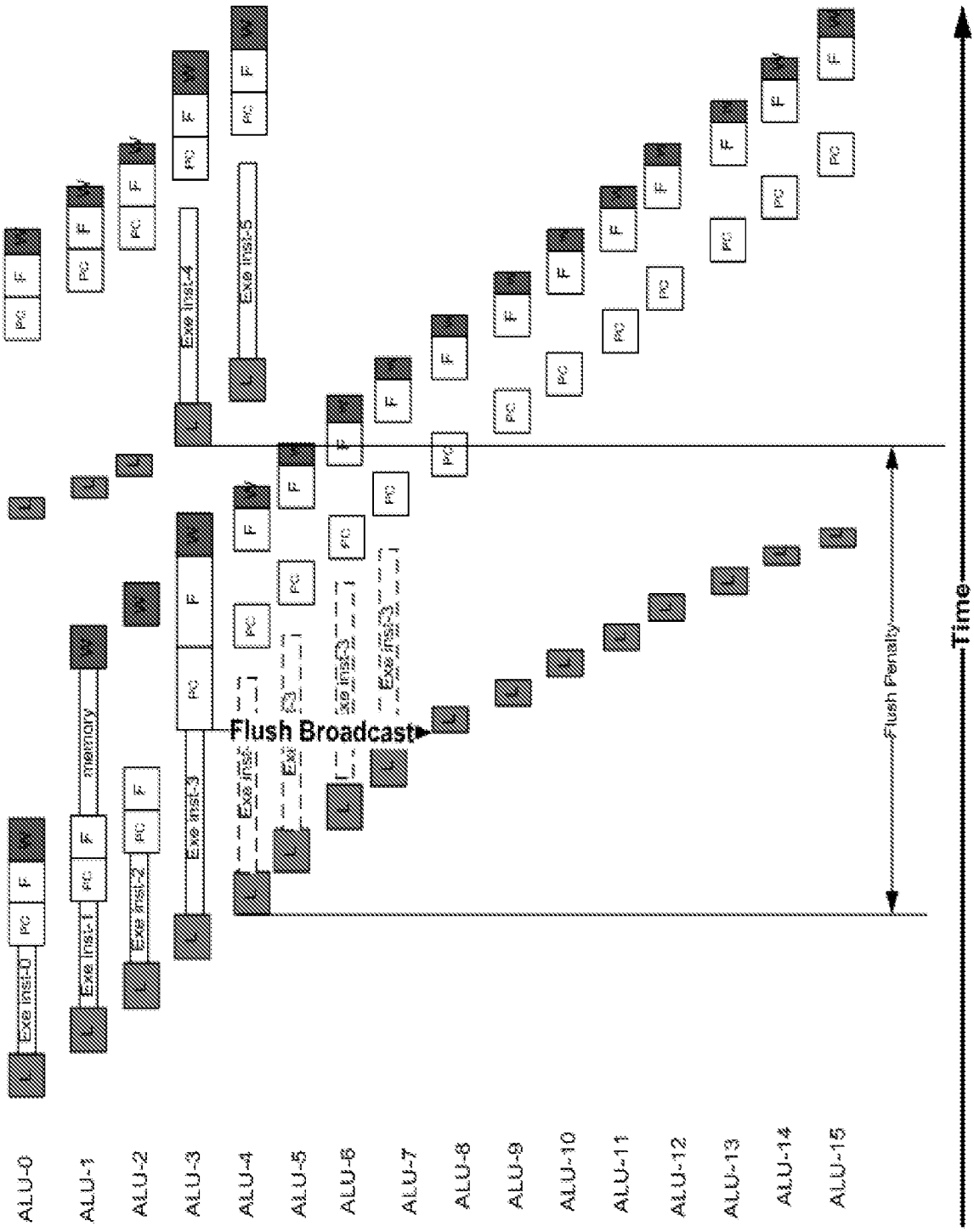
ALU-0
ALU-1
ALU-2
ALU-3
ALU-4
ALU-5
ALU-6
ALU-7
ALU-8
ALU-9
ALU-10
ALU-11
ALU-12
ALU-13
ALU-14
ALU-15
Flush Broadcast
Flush Penalty
Time

Figure 10

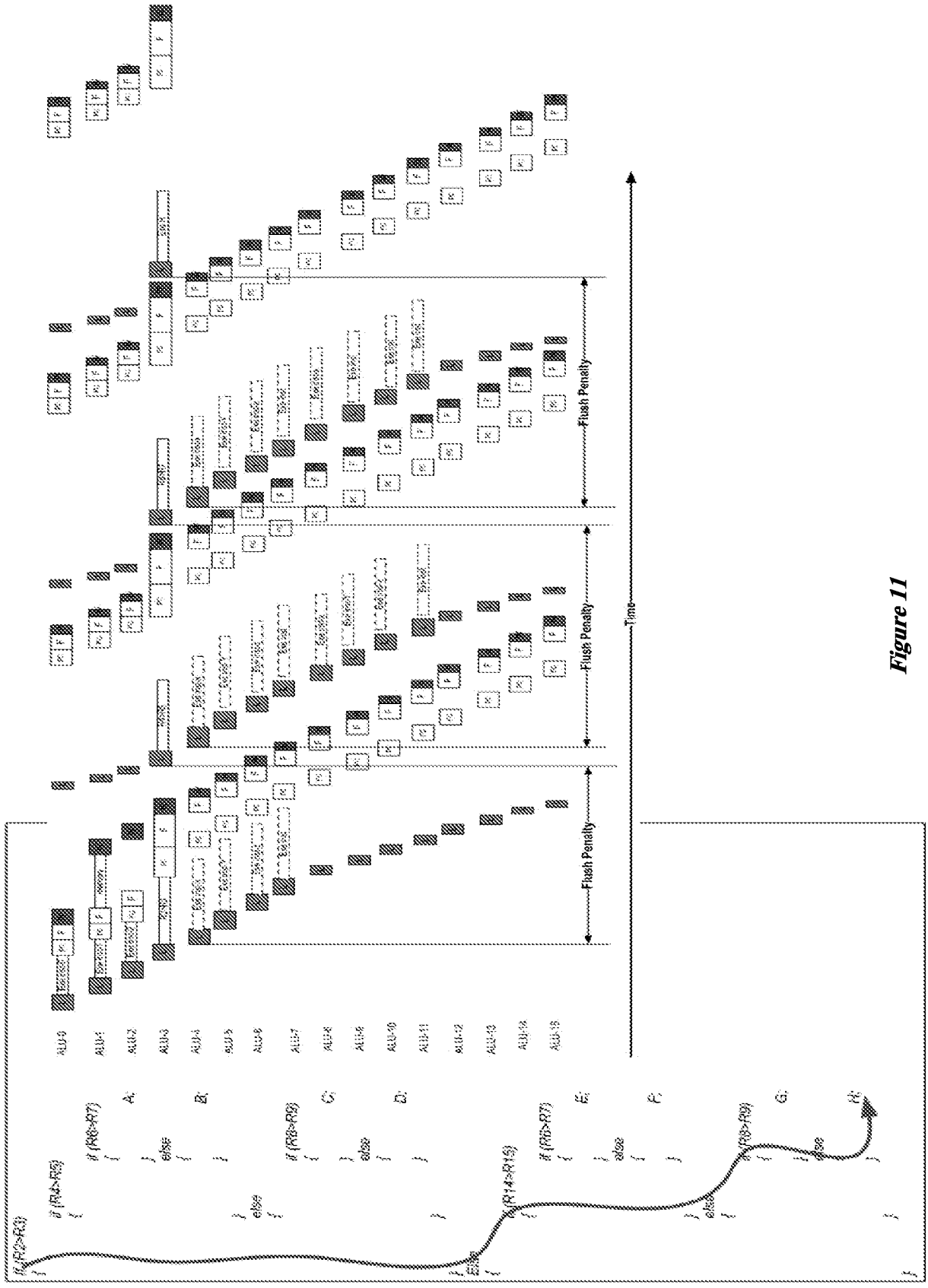

Figure 11

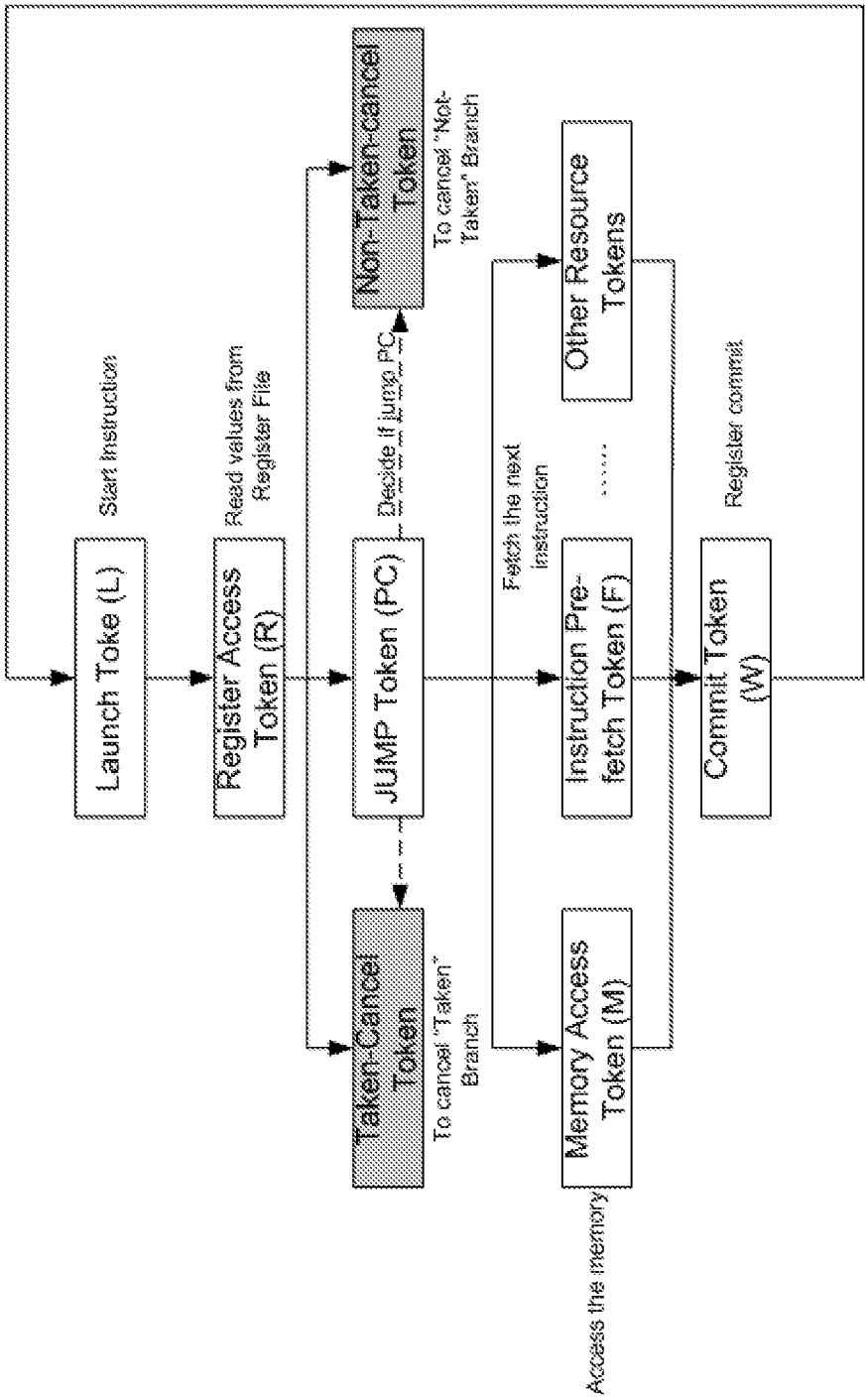
Launch Toke (L)
Start instruction
Register Access Token (R)
Read values from Register File
JUMP Token (PC)
Decide if jump PC
Non-Taken-cancel Token
To cancel "Not-Taken" Branch
Taken-Cancel Token
To cancel "Taken" Branch
Fetch the next instruction
Instruction Pre-fetch Token (F)
Other Resource Tokens
.....
Memory Access Token (M)
Access the memory
Commit Token (W)
Register commit

Figure 12

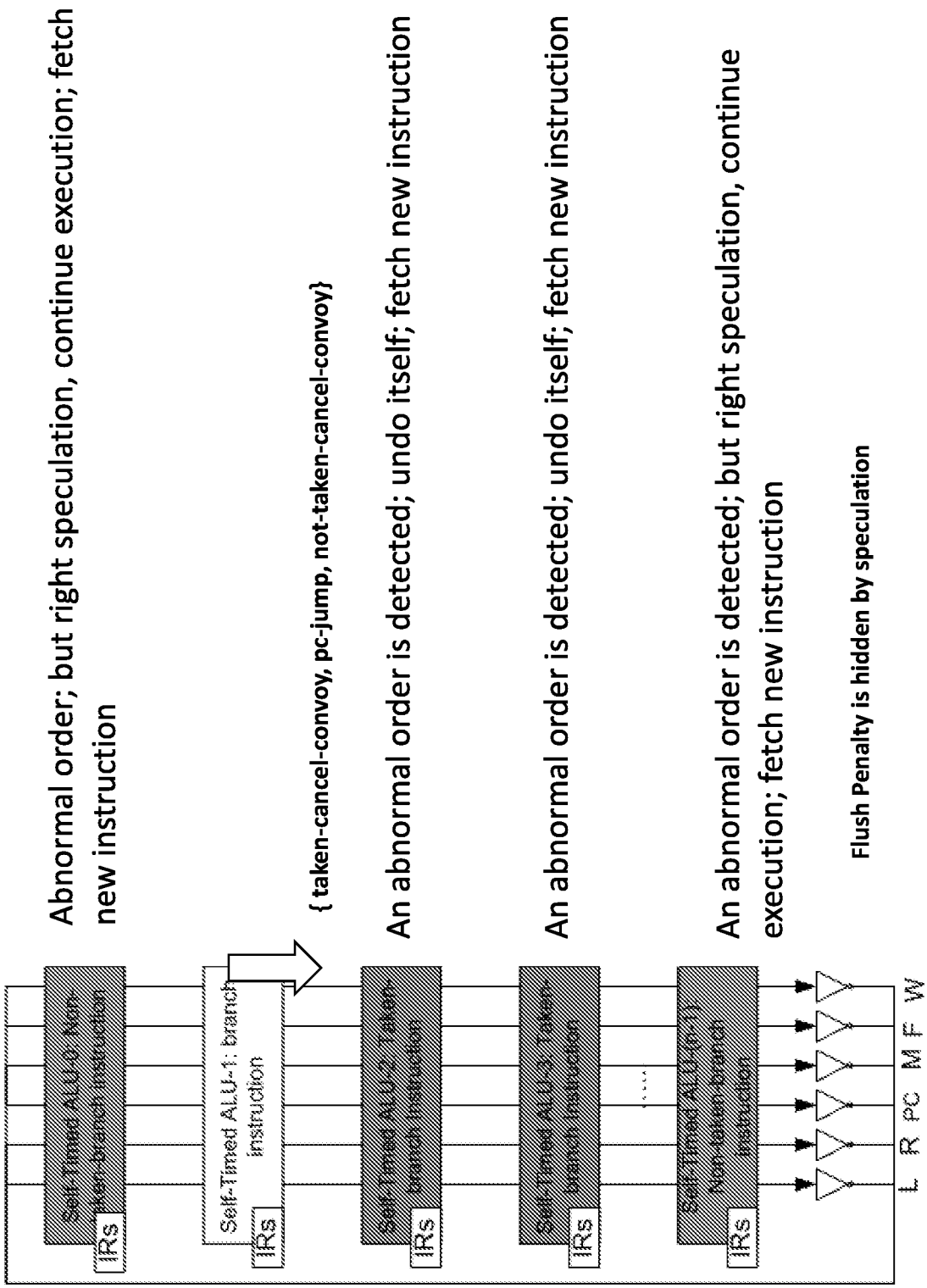
Abnormal order; but right speculation, continue execution; fetch new instruction
{ taken-cancel-convoy, pc-jump, not-taken-cancel-convoy}
An abnormal order is detected; undo itself; fetch new instruction
An abnormal order is detected; undo itself; fetch new instruction
An abnormal order is detected; but right speculation, continue execution; fetch new instruction
Flush Penalty is hidden by speculation
Self-Timed ALU-0: Non-taken-branch instruction
IRs
Self-Timed ALU-1: branch instruction
IRs
Self-Timed ALU-2: Taken-branch instruction
IRs
Self-Timed ALU-3: Taken-branch instruction
IRs
Self-Timed ALU(n-1): Non-taken-branch instruction
IRs
L R PC M F W

Figure 13

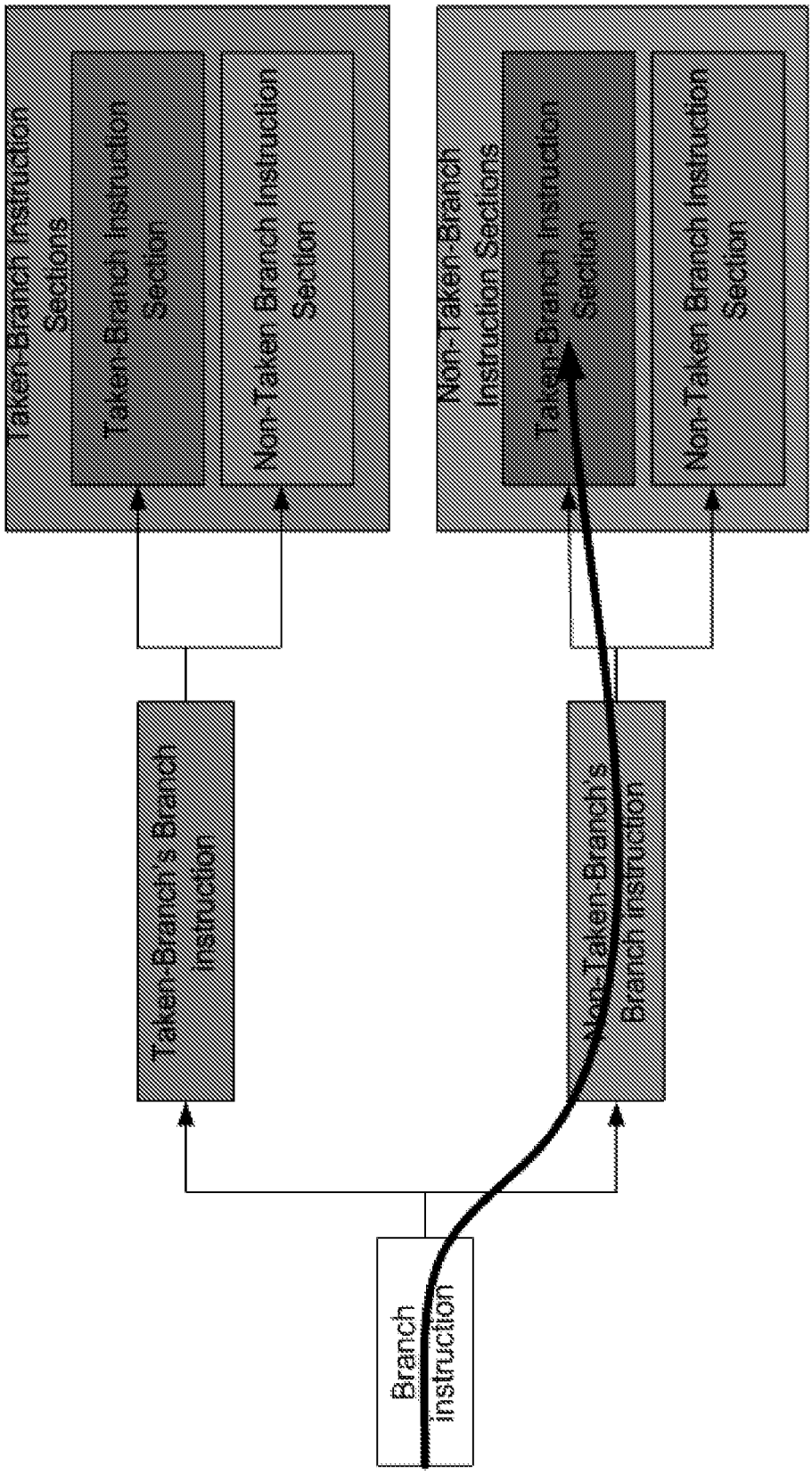
Taken-Branch Instruction Sections
Taken-Branch Instruction Section
Non-Taken Branch Instruction Section
Non-Taken-Branch Instruction Sections
Taken-Branch Instruction Section
Non-Taken Branch Instruction Section
Taken-Branch's Branch instruction
Non-Taken-Branch's Branch instruction
Branch instruction

Figure 14

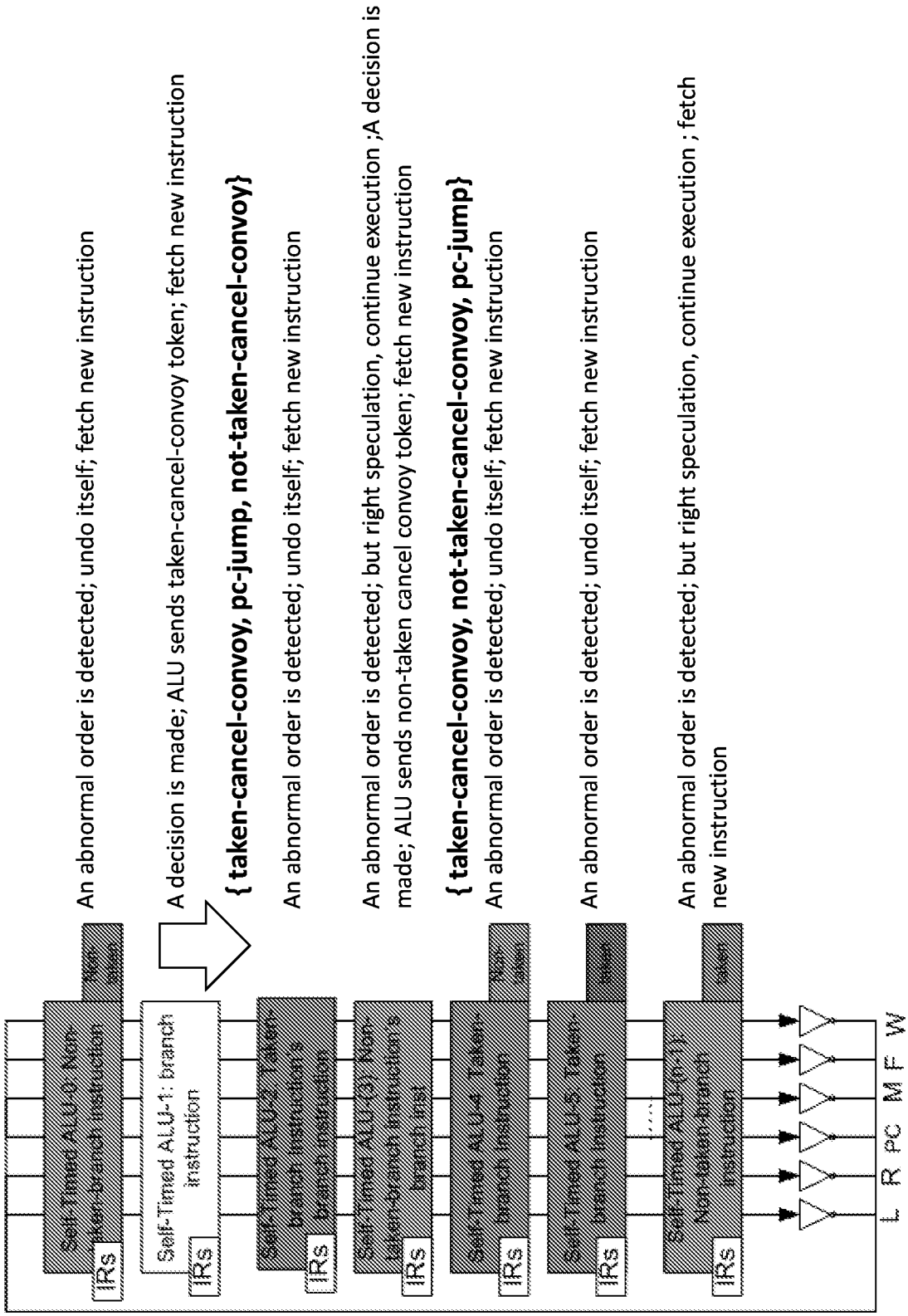
Self-Timed ALU-0: Non-taken-branch instruction
IRs
Non-abs
An abnormal order is detected; undo itself; fetch new instruction
Self-Timed ALU-1: branch instruction
IRs
A decision is made; ALU sends taken-cancel-convoy token; fetch new instruction
{ taken-cancel-convoy, pc-jump, not-taken-cancel-convoy}
Self-Timed ALU-2: Taken-branch instruction's branch instruction
IRs
An abnormal order is detected; undo itself; fetch new instruction
Self-Timed ALU-3: Non-taken-branch instruction's branch inst
IRs
An abnormal order is detected; but right speculation, continue execution ;A decision is made; ALU sends non-taken cancel convoy token; fetch new instruction
{ taken-cancel-convoy, not-taken-cancel-convoy, pc-jump}
Self-Timed ALU-4: Taken-branch instruction
IRs
Non-abs
An abnormal order is detected; undo itself; fetch new instruction
Self-Timed ALU-5: Taken-branch instruction
IRs
Abs
An abnormal order is detected; undo itself; fetch new instruction
.....
Self-Timed ALU-(n-1): Non-taken-branch instruction
IRs
Abs
An abnormal order is detected; but right speculation, continue execution ; fetch new instruction
L R PC M F W

Figure 15

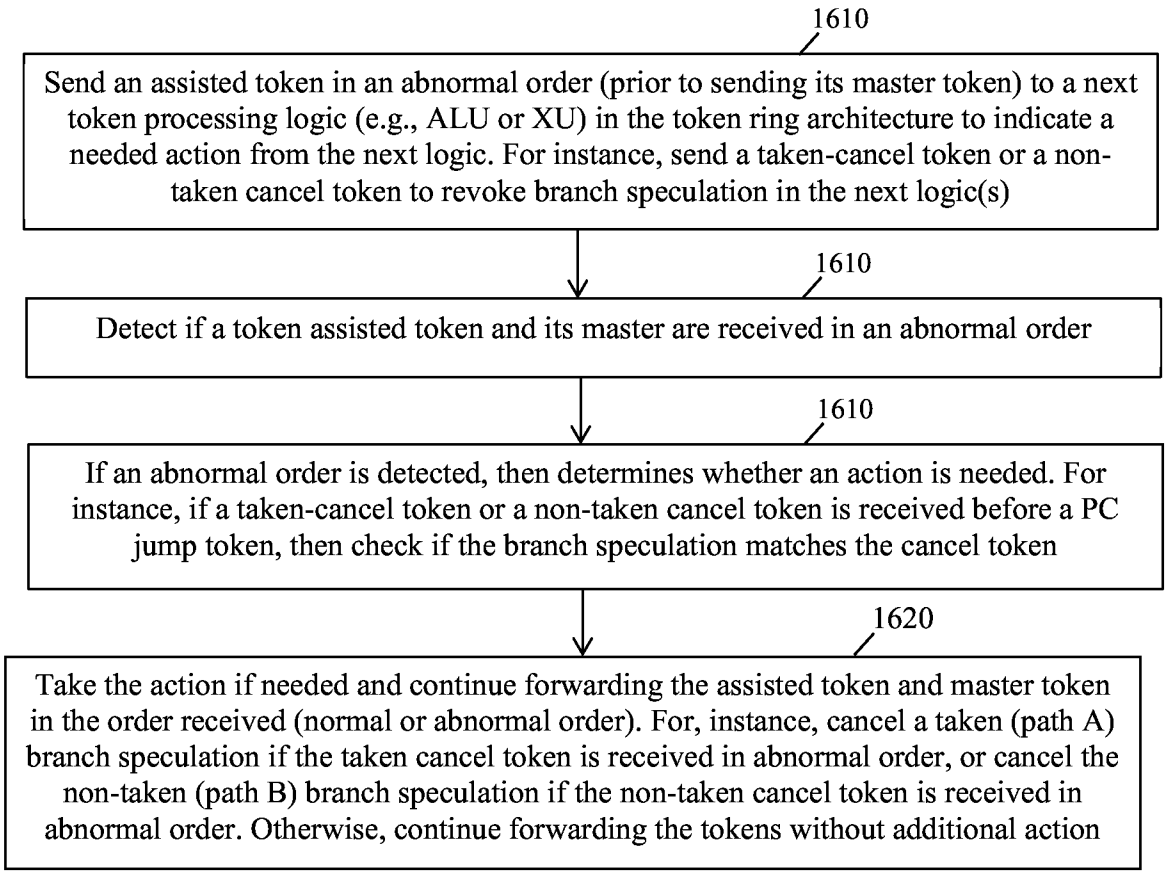
1610
Send an assisted token in an abnormal order (prior to sending its master token) to a next token processing logic (e.g., ALU or XU) in the token ring architecture to indicate a needed action from the next logic. For instance, send a taken-cancel token or a non-taken cancel token to revoke branch speculation in the next logic(s)
1610
Detect if a token assisted token and its master are received in an abnormal order
1610
If an abnormal order is detected, then determines whether an action is needed. For instance, if a taken-cancel token or a non-taken cancel token is received before a PC jump token, then check if the branch speculation matches the cancel token
1620
Take the action if needed and continue forwarding the assisted token and master token in the order received (normal or abnormal order). For, instance, cancel a taken (path A) branch speculation if the taken cancel token is received in abnormal order, or cancel the non-taken (path B) branch speculation if the non-taken cancel token is received in abnormal order. Otherwise, continue forwarding the tokens without additional action

Figure 16

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2014/054621

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - G06F 9/318 (2014.01)
CPC - G06F 9/30185 (2014.09)
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC(8) - G06F 9/318, G06F 7/74, G06F 9/445, G06F 9/455, G06F 9/48, G06F 9/38, G06F 9/30, G06F 9/44 (2014.01)
USPC - 703/27, 712/E09.054, 712/E09.037, 712/209, 703/25, 712/E09.059, 712/E09.046, 712/E09.055, 712/E09.049, 712/E09.029

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
CPC - G06F 9/30185, G06F 9/3824, G06F 9/3853, G06F 9/30174, G06F 9/382, G06F 9/3816, G06F 9/3812, G06F 9/30152, G06F 9/3836, G06F 9/3802, G06F 9/445, G06F 9/30167, G06F 9/4812, G06F 9/45533, G06F 7/74 (2014.09) (keyword delimited)

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

Orbit, Google Patents, Google Scholar, Google.
asynchronous processor, master token, execution unit, assisted token, abnormal order, operation status, program counter jump token

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2002/0156995 A1 (MARTIN et al) 24 October 2002 (24.10.2002) entire document	1-24
A	US 2011/0078418 A1 (ROZAS et al) 31 March 2011 (31.03.2011) entire document	1-24
A	EP 0 677 807 B1 (SONG) 03 March 1999 (03.03.1999) entire document	1-24

☐ Further documents are listed in the continuation of Box C. ☐

* 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
02 November 2014	19 NOV 2014

Name and mailing address of the ISA/US	Authorized officer:
Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-3201	Blaine R. Copenheaver PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

Form PCT/ISA/210 (second sheet) (July 2009)