US 20050246581A1

(19) **United States**

(12) **Patent Application Publication**

(10) **Pub. No.: US 2005/0246581 A1**

(43) **Pub. Date: Nov. 3, 2005**

Jardine et al.

(54) **ERROR HANDLING SYSTEM IN A REDUNDANT PROCESSOR**

(75) Inventors: **Robert L. Jardine**, Cupertino, CA (US); **James S. Klecka**, Georgetown, TX (US); **William F. Bruckert**, Los Gatos, CA (US); **James R. Smullen**, Carmel, CA (US); **David J. Garcia**, Los Gatos, CA (US)

Correspondence Address:
HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400 (US)

(73) Assignee: **Hewlett-Packard Development Company, L.P.**, Houston, TX

(21) Appl. No.: **11/045,401**

(22) Filed: **Jan. 27, 2005**

Related U.S. Application Data

(60) Provisional application No. 60/557,812, filed on Mar. 30, 2004.

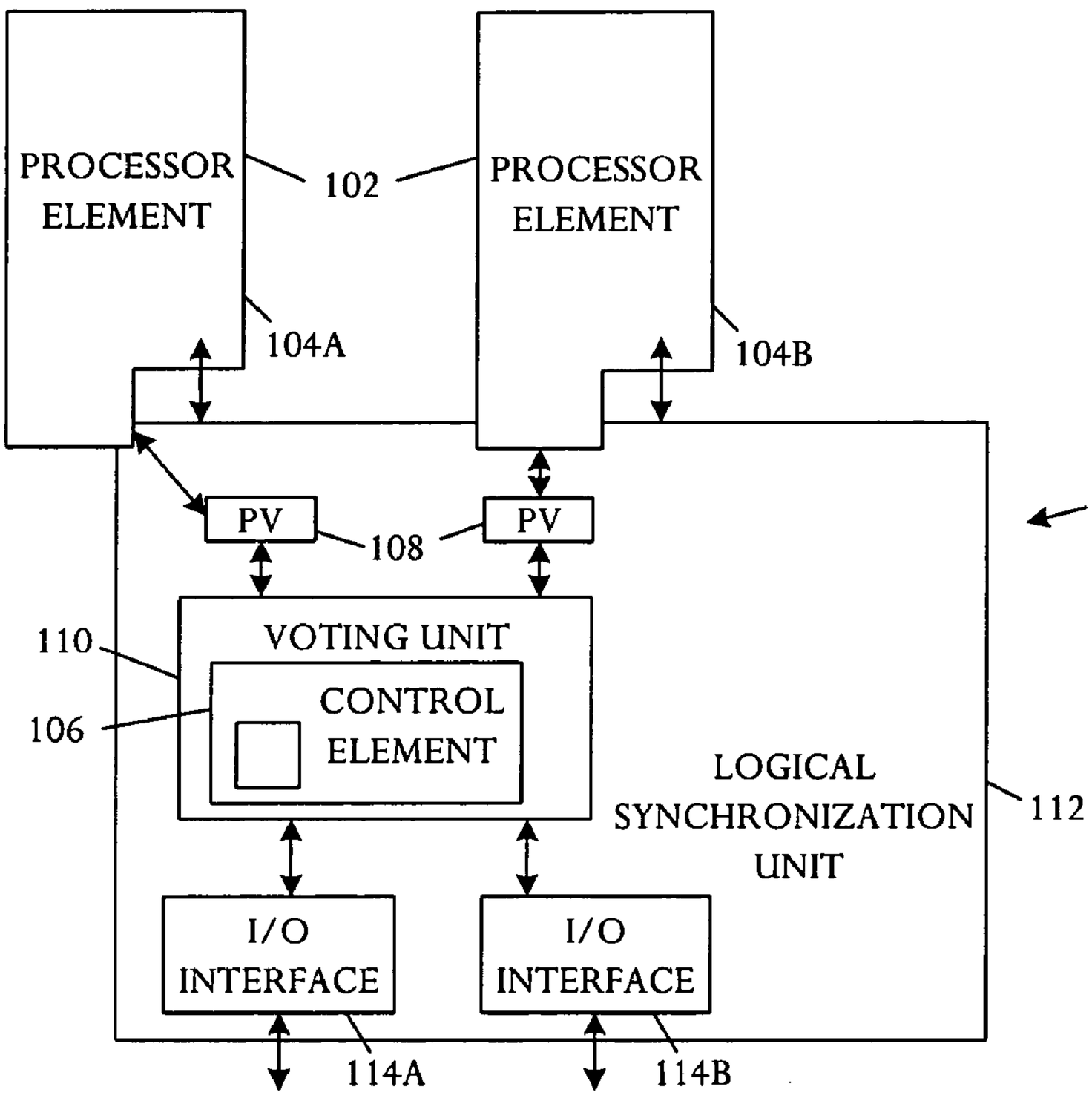
Publication Classification

(51) **Int. Cl.**[7] **G06F 11/00**

(52) **U.S. Cl.** **714/12**

(57) **ABSTRACT**

In a redundant-processor computing device, an error handling method comprises detecting equivalent disparity among processor elements of the computing device operating and responding to the detected equivalent disparity by evaluating secondary considerations of processor fidelity.



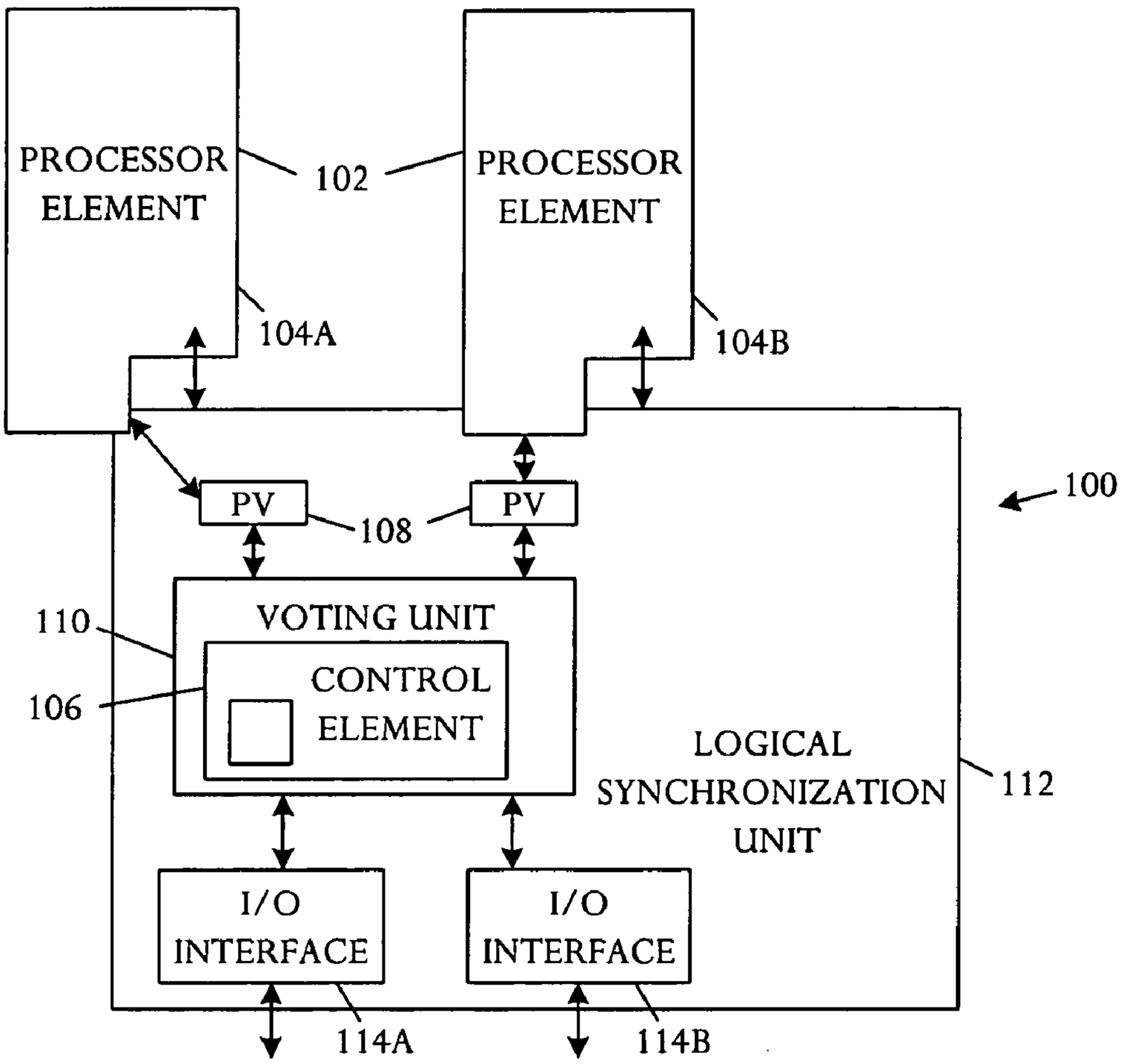

PROCESSOR ELEMENT
102
PROCESSOR ELEMENT
104A
104B
PV
108
PV
100
VOTING UNIT
110
CONTROL ELEMENT
106
LOGICAL SYNCHRONIZATION UNIT
112
I/O INTERFACE
I/O INTERFACE
114A
114B

FIG. 1

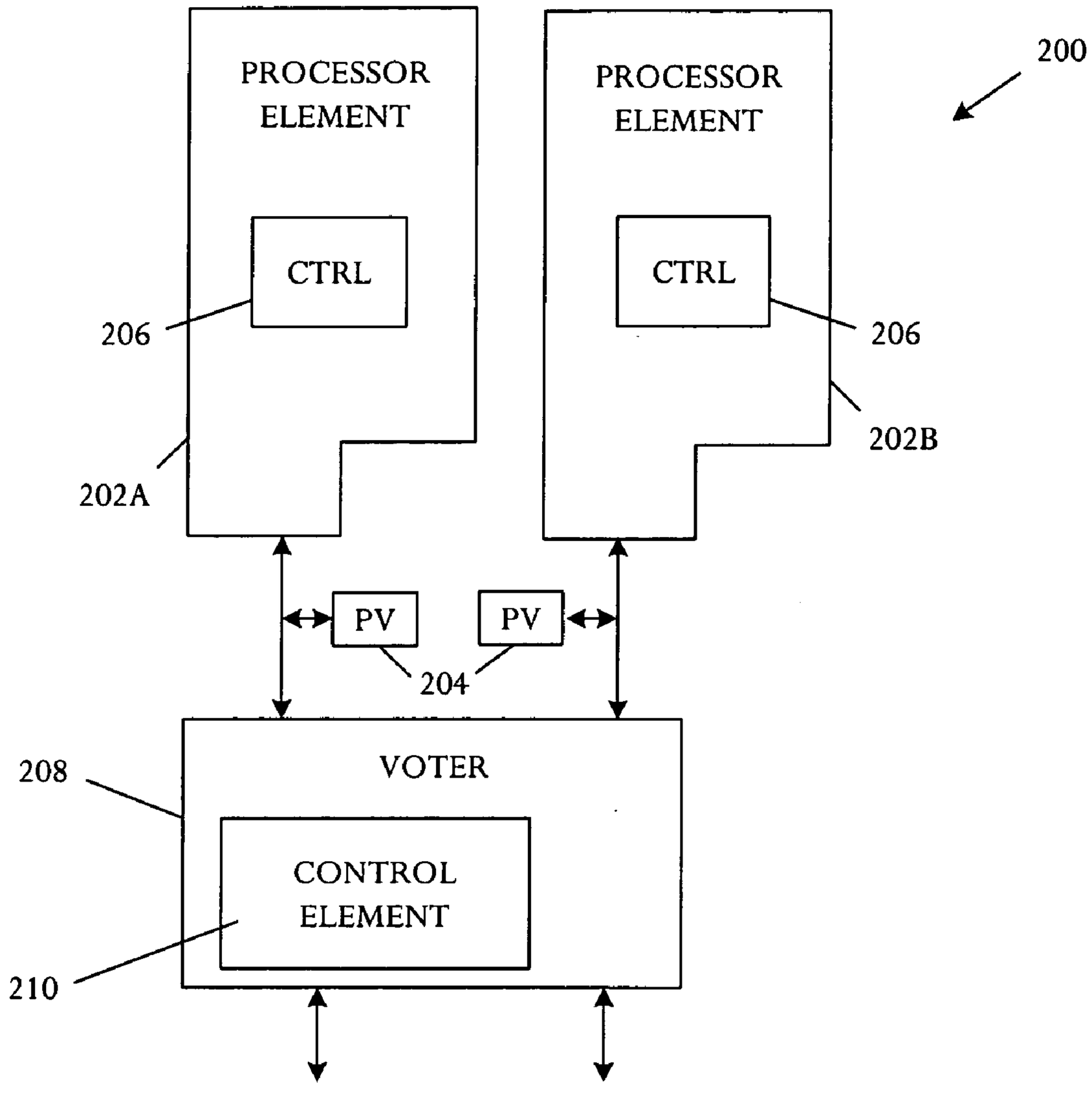

PROCESSOR ELEMENT
CTRL
206
202A
PROCESSOR ELEMENT
CTRL
206
202B
200
PV
PV
204
VOTER
208
CONTROL ELEMENT
210

FIG. 2

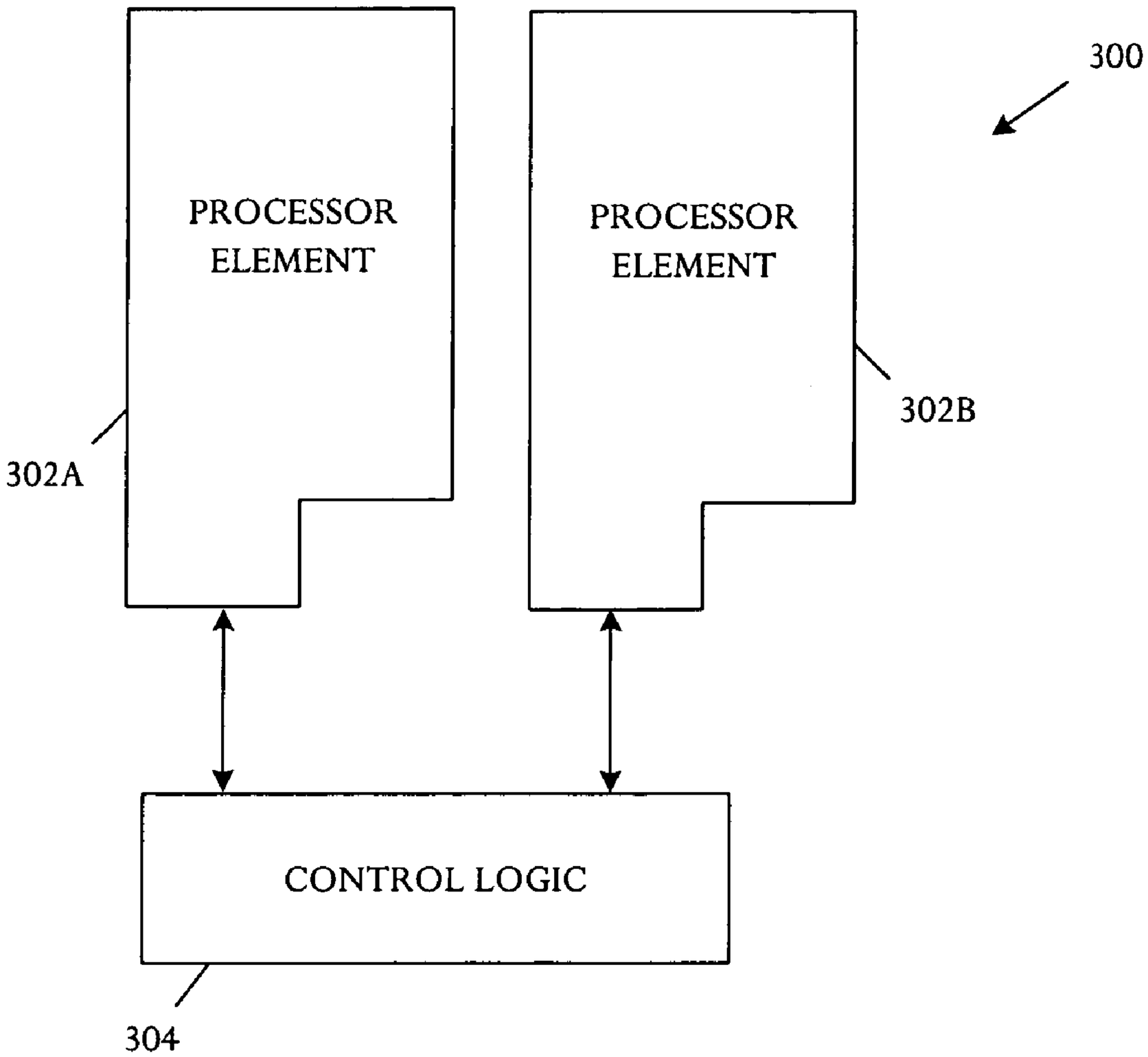

300
PROCESSOR ELEMENT
PROCESSOR ELEMENT
302A
302B
CONTROL LOGIC
304

FIG. 3

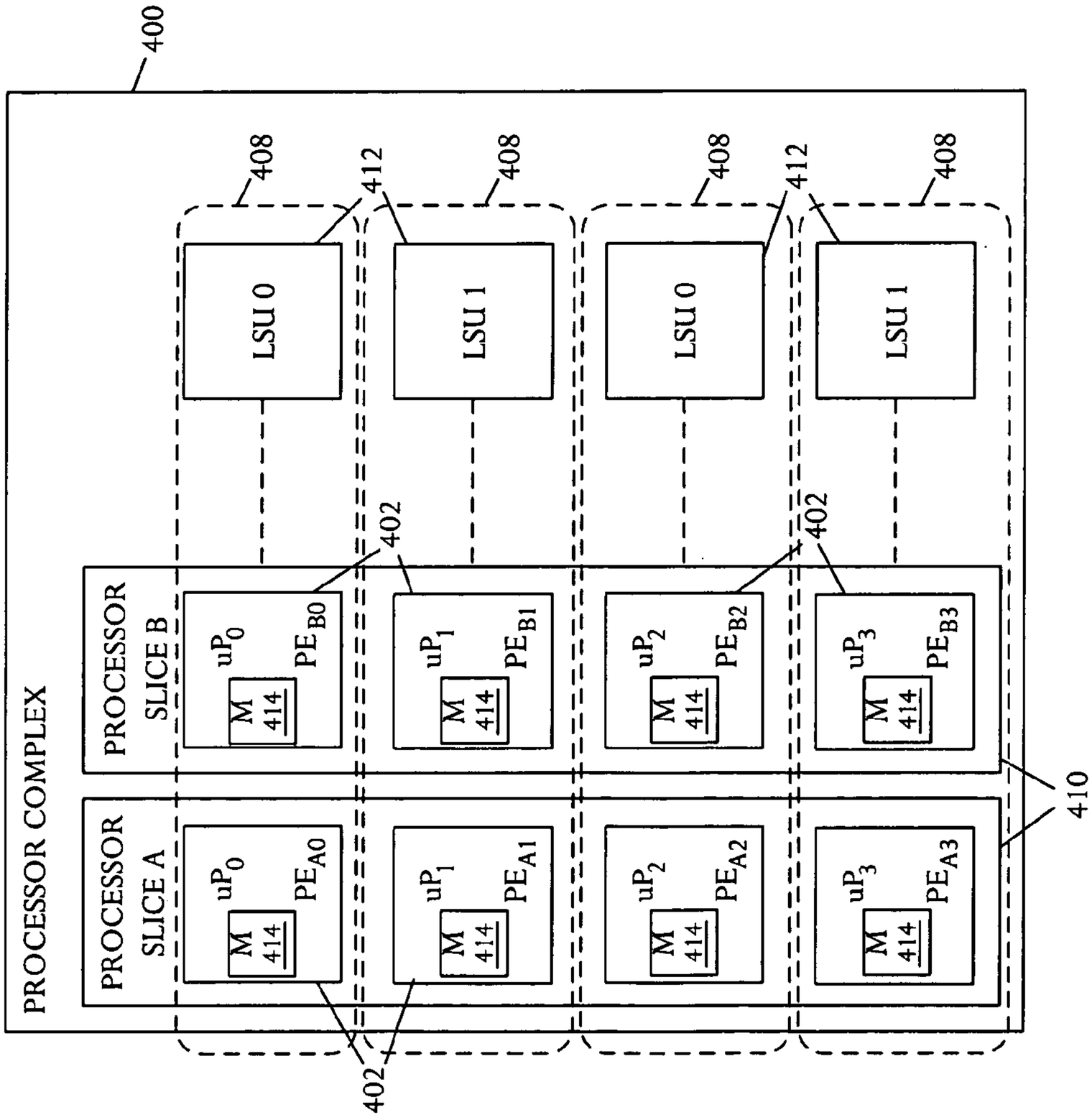

400
408
412
LSU 0
LSU 1
PROCESSOR COMPLEX
PROCESSOR SLICE B
PROCESSOR SLICE A
uP0
uP1
uP2
uP3
M 414
PEB0
PEB1
PEB2
PEB3
PEA0
PEA1
PEA2
PEA3
402
410

FIG. 4

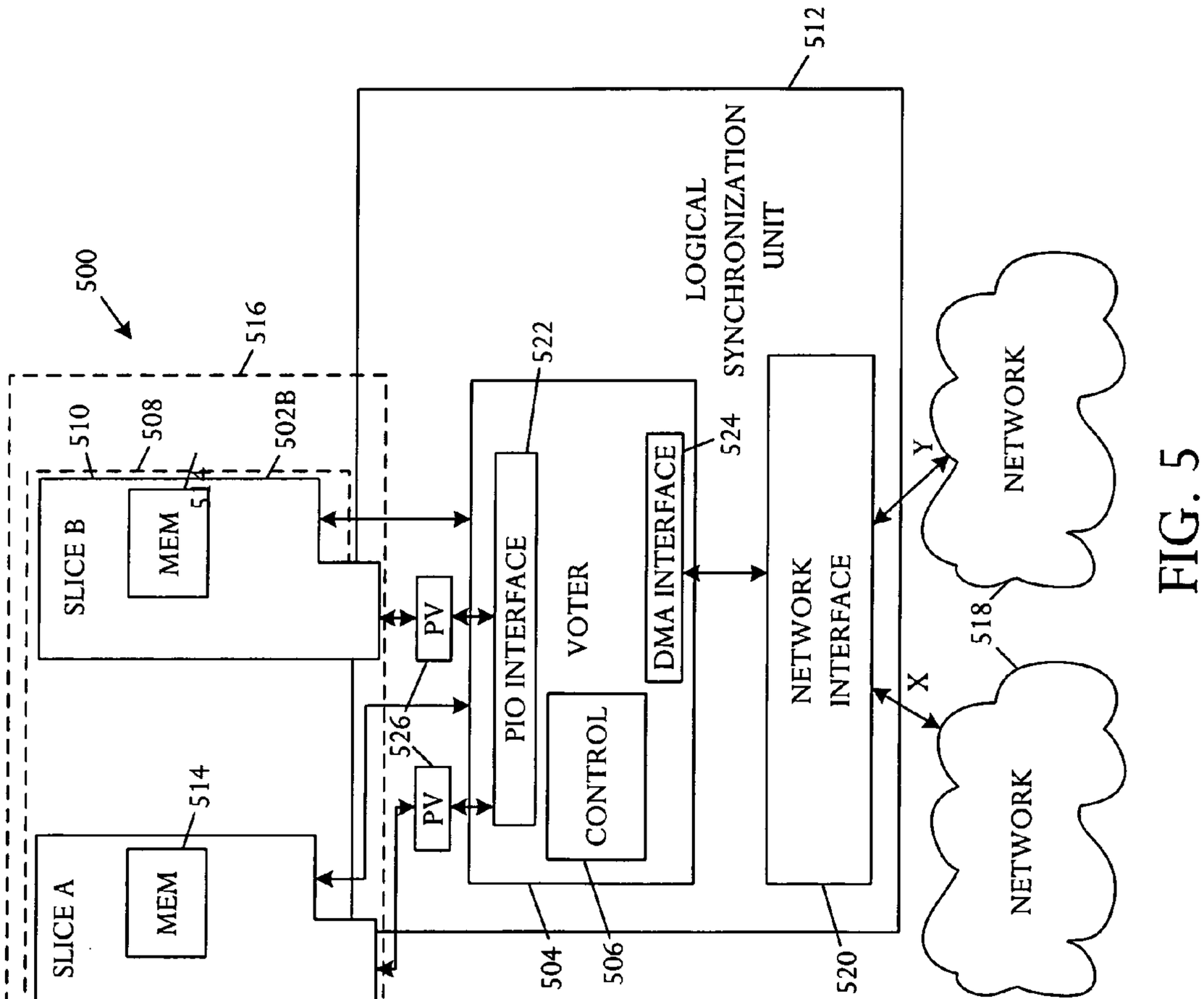

500
516
512
LOGICAL SYNCHRONIZATION UNIT
522
502B
510
508
SLICE B
MEM
514
PV
526
PIO INTERFACE
VOTER
CONTROL
DMA INTERFACE
524
NETWORK INTERFACE
Y
X
NETWORK
518
SLICE A
MEM
514
PV
510
502A
504
506
520

FIG. 5

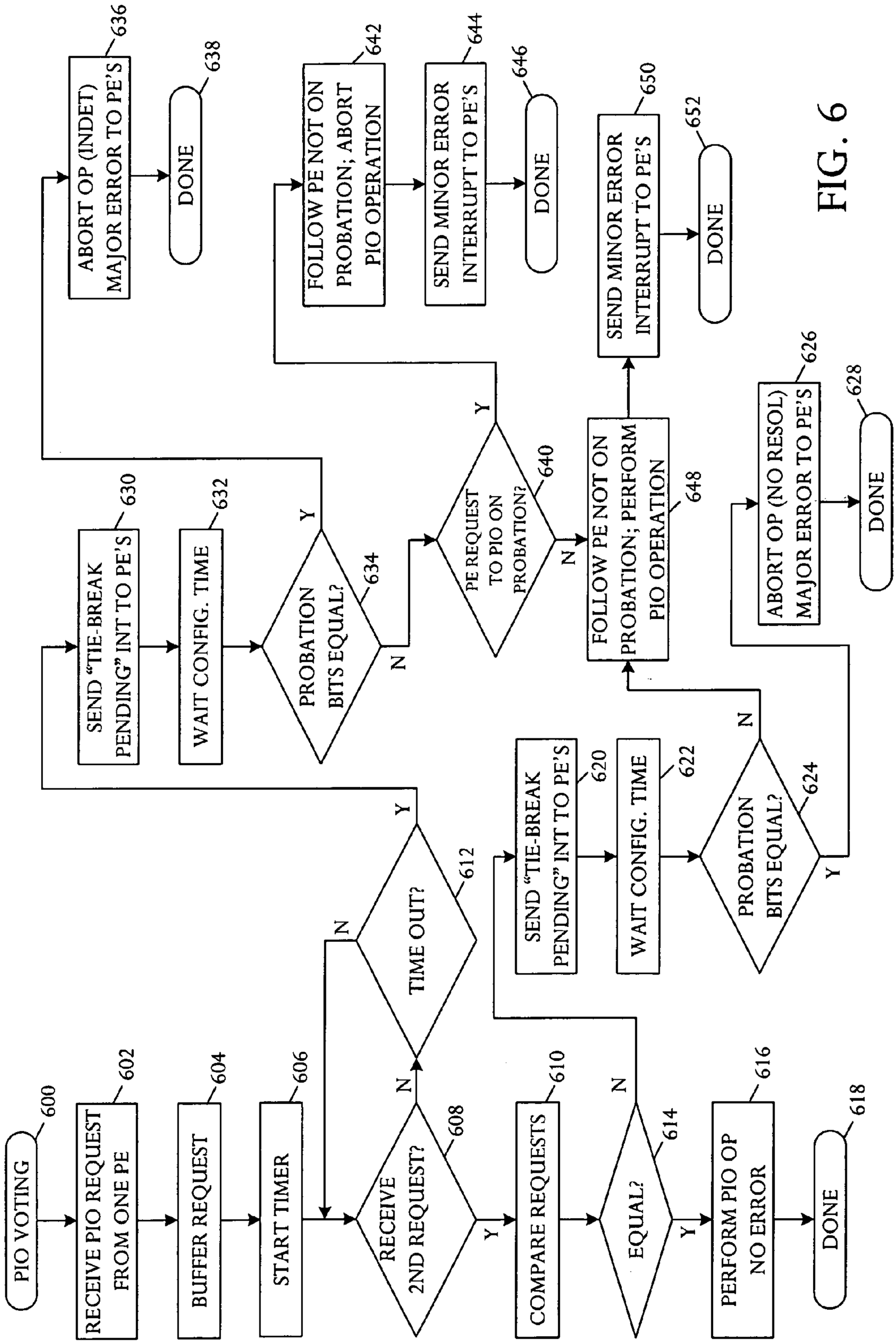

PIO VOTING
600
RECEIVE PIO REQUEST FROM ONE PE
602
BUFFER REQUEST
604
START TIMER
606
RECEIVE 2ND REQUEST?
608
COMPARE REQUESTS
610
TIME OUT?
612
EQUAL?
614
PERFORM PIO OP NO ERROR
616
DONE
618
SEND "TIE-BREAK PENDING" INT TO PE'S
620
WAIT CONFIG. TIME
622
PROBATION BITS EQUAL?
624
ABORT OP (NO RESOL) MAJOR ERROR TO PE'S
626
DONE
628
SEND "TIE-BREAK PENDING" INT TO PE'S
630
WAIT CONFIG. TIME
632
PROBATION BITS EQUAL?
634
PE REQUEST TO PIO ON PROBATION?
640
FOLLOW PE NOT ON PROBATION; PERFORM PIO OPERATION
648
SEND MINOR ERROR INTERRUPT TO PE'S
650
DONE
652
FOLLOW PE NOT ON PROBATION; ABORT PIO OPERATION
642
SEND MINOR ERROR INTERRUPT TO PE'S
644
DONE
646
ABORT OP (INDET) MAJOR ERROR TO PE'S
636
DONE
638
Y
N

FIG. 6

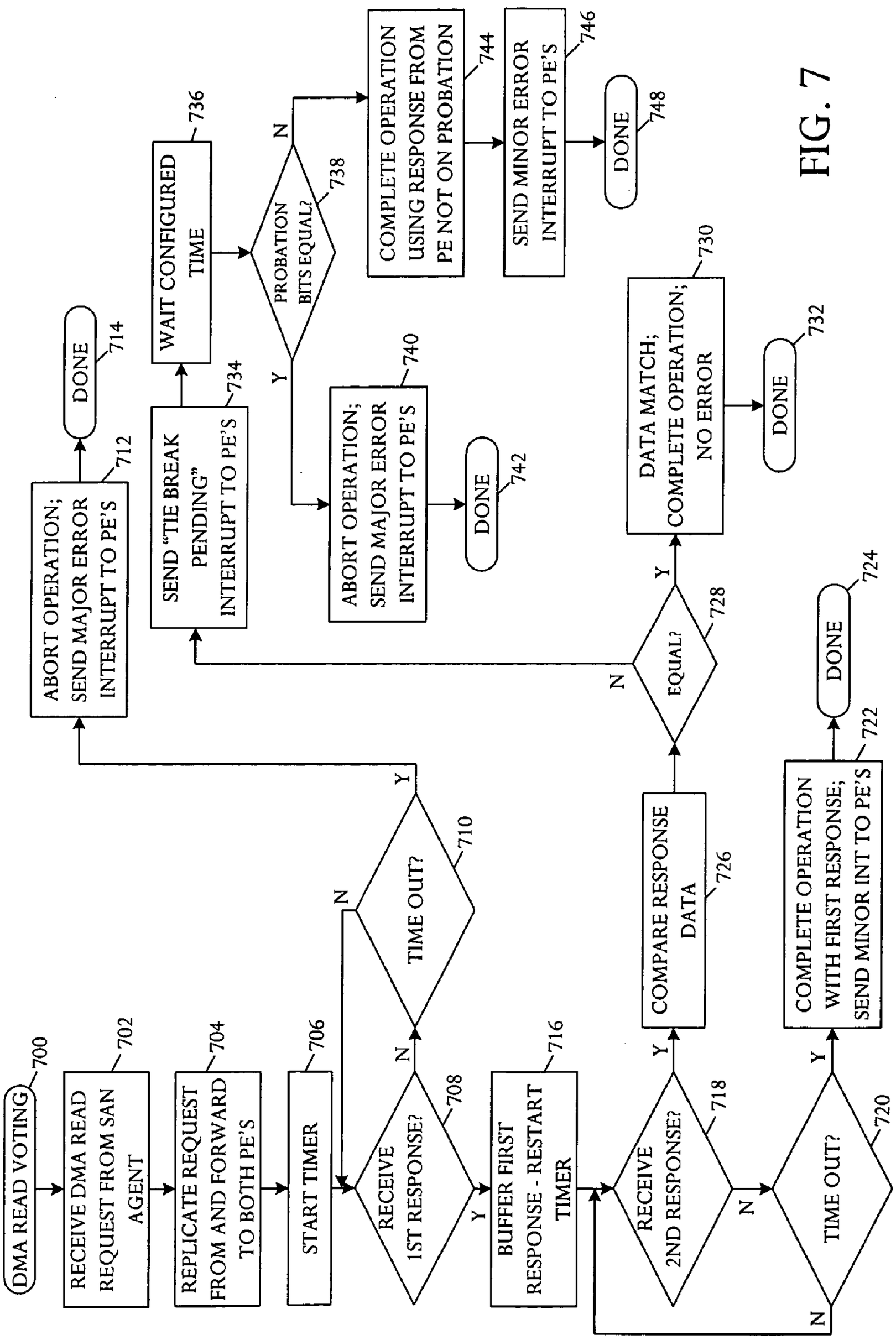

DMA READ VOTING
700
RECEIVE DMA READ REQUEST FROM SAN AGENT
702
REPLICATE REQUEST FROM AND FORWARD TO BOTH PE'S
704
START TIMER
706
RECEIVE 1ST RESPONSE?
708
N
Y
BUFFER FIRST RESPONSE - RESTART TIMER
716
RECEIVE 2ND RESPONSE?
718
N
Y
TIME OUT?
720
N
Y
COMPLETE OPERATION WITH FIRST RESPONSE; SEND MINOR INT TO PE'S
722
DONE
724
COMPARE RESPONSE DATA
726
EQUAL?
728
N
Y
DATA MATCH; COMPLETE OPERATION; NO ERROR
730
DONE
732
TIME OUT?
710
N
Y
ABORT OPERATION; SEND MAJOR ERROR INTERRUPT TO PE'S
712
DONE
714
WAIT CONFIGURED TIME
736
SEND "TIE BREAK PENDING" INTERRUPT TO PE'S
734
PROBATION BITS EQUAL?
738
N
Y
COMPLETE OPERATION USING RESPONSE FROM PE NOT ON PROBATION
744
SEND MINOR ERROR INTERRUPT TO PE'S
746
DONE
748
ABORT OPERATION; SEND MAJOR ERROR INTERRUPT TO PE'S
740
DONE
742

FIG. 7

ERROR HANDLING SYSTEM IN A REDUNDANT PROCESSOR

BACKGROUND

[0001] System availability, scalability, and data integrity are fundamental characteristics of enterprise systems. A continuous performance capability is imposed in financial, communication, and other fields that use enterprise systems for applications such as stock exchange transaction handling, credit and debit card systems, telephone networks, and the like. Highly reliable systems are often implemented in applications with high financial or human costs, in circumstances of massive scaling, and in conditions that outages and data corruption cannot be tolerated.

[0002] Some systems combine multiple redundant processors running the same operations so that an error in a single processor can be detected and/or corrected. Results attained for each of the processors can be mutually compared. If all results are the same, all processors are presumed, with high probability of correctness, to be functioning properly. However, if results differ analysis is performed to determine which processor is operating incorrectly. Results from the multiple processors can be "voted" with the "winning" result determined to be correct. For example, a system with three processor elements typically uses the result attained by two of the three processors.

[0003] A difficulty arises for duplex systems with two executing processors since the even number of processor elements can result in a "voting tie" situation that may lead to aborted operation and outage. Ties can be avoided by running an odd number of processors, although a single processor does not have the fault detection capability provided by voting. A three or more processor system adds product cost.

SUMMARY

[0004] In accordance with an embodiment of a redundant-processor computing device, an error handling method comprises detecting equivalent disparity among processor elements of the computing device operating and responding to the detected equivalent disparity by evaluating secondary considerations of processor fidelity.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] Embodiments of the invention relating to both structure and method of operation may best be understood by referring to the following description and accompanying drawings:

[0006] **FIG. 1** is a schematic block diagram that illustrates an embodiment of a control apparatus for usage in a redundant-processor computing device and having capability to resolve a mutual disparity or tie condition;

[0007] **FIG. 2** is a schematic block diagram depicting an embodiment of a computing system with capability to resolve disparity and break ties among a plurality of processor elements using a probation vector;

[0008] **FIG. 3** is a schematic block diagram illustrating an embodiment of a computing system configured in a redundant-processor arrangement that imposes a selected-duration short delay in the event of a disparity or tie condition;

[0009] **FIG. 4** is a schematic block diagram showing an embodiment of a processor complex within which the illustrative error handling system may be implemented;

[0010] **FIG. 5** is a schematic block diagram showing an embodiment of a computing system capable of detecting equivalent disparity among the processor elements and responding by evaluating secondary considerations of processor fidelity;

[0011] **FIG. 6** is a flow chart depicting an embodiment of an error handling method in a redundant-processor computing device that has tie-breaking capability during programmed input/output (PIO) voting in a duplex configuration; and

[0012] **FIG. 7** is a flow chart illustrating an embodiment of an error handling method in a redundant-processor computing device during direct memory access (DMA) read voting in a duplex configuration.

DETAILED DESCRIPTION

[0013] A processor may incorporate multiple, redundant, loosely-coupled processor elements for error detection. A duplex arrangement using two processor elements is susceptible to a"voting tie" situation. Ties may be avoided by using an odd number of processors at the expense of fault detection capability if a single processor element is used and by adding costs for incorporation of additional processor elements. The illustrative system and method may use other information to resolve conflicts and break ties. Accordingly, an effective processor may be configured using only two processor elements for voting or comparison.

[0014] Referring to **FIG. 1**, a schematic block diagram illustrates an embodiment of a control apparatus **100** for usage in a redundant-processor computing device **102**. The control apparatus **100** is operative in a configuration with a plurality of processor elements **104**A and **104**B and can resolve a mutual disparity or "tie" condition among processor elements. The control apparatus **100** can be used to break ties in the case of voting error, for example with an even number of processor elements, using other available information. The control apparatus **100** includes a control element **106** that detects equivalent disparity among the processor elements **104**A, **104**B and responds by evaluating secondary considerations of processor fidelity.

[0015] The control element **106** determines whether evaluation of the secondary considerations is insufficient to resolve the disparity among the processor elements **104**A, **104**B and, if so, terminates computing device operations.

[0016] The computing device **102** can be a computer processor that uses multiple, redundant, loosely-synchronized processor elements **104**A, **104**B to detect and manage errors. A configuration with an even number of processor elements **104**A, **104**B is susceptible to a voting "tie" condition in which actions or results from the processor elements differ. For example, a computing device **102** may have two processor elements **104**A, **104**B so that any disparity is equivalent and results in a tie condition. Typically, an odd number of processing element, for example three, can be used at added cost to avoid ties.

[0017] In some situations, other information which may be called secondary considerations of fidelity may be available

to resolve the disparity and break the tie. The other information is heuristic data which is sufficiently predictive to be trusted for disparity resolution. If the tie cannot be broken by use of the other information, then the error is considered sufficiently serious that the processor is halted due to an inability to guarantee correctness of either of the unequal voted data items.

[0018] Some embodiments may include a control element **106** that evaluates the secondary conditions of processor fidelity while the processor elements **104**A, **104**B are executing before equivalent disparity is detected and sets a probation vector **108** according to the evaluation. For example, the probation vector **108** may be implemented in a voting unit **110** and used by the voting unit **110** to resolve disparities and break ties in predetermined conditions. In a particular example, the probation vector **108** can have one bit of state per processor element **104**A, **104**B. Control logic, such as software, executing in each processor element **104**A, **104**B can set the bit in conditions that the logic has accumulated information for usage in breaking future ties, or very recent ties. The control logic can periodically reset the probation vector bits.

[0019] The voting unit **110**, upon detecting a disparity or tie condition, may delay acting upon the condition or declaring a fatal error situation. Instead, the voting unit **110** can hold the compared values for a short duration time period before acting. Accordingly, the control element **106** can interject a delay between equivalent disparity detection and termination of computer device operation. The delay enables control logic, for example software, to possibly detect other errors or gather information pertinent to resolving the disparity or breaking the tie. The delay can also break potential race conditions. For example, if a self-detectable error occurs simultaneously or nearly the same time as a misvote, the delay enables further collection of information or analysis before the voter declares the misvote condition, enabling recognition of the error and resolution of the vote. In a particular duplex embodiment, if the control logic sets one of the two bits in the probation vector **108** during the short delay imposed by the voting unit **110**, then the voting unit **110** resolves the disparity or breaks the tie in favor of the processor element, either **104**A or **104**B, that is not on probation. While the condition remains an error condition, the error is made recoverable. If the control logic does not set the bit in the probation vector prior to or during the delay, the error is considered to be fatal to the computing device **102**, and operation is halted.

[0020] In a particular embodiment, the computing device **102** can include a logical synchronization unit **112** that contains the voting unit **110** and input/output interfaces **114**A and **114**B. For example, the interfaces **114**A and **114**B may include a programmed input/output (PIO) interface and a direct memory access (DMA) interface. A possible equivalent disparity or tie condition may include a condition of a first processor element that performs a programmed input/output (PIO) action while a second processor element does not. A second example of a equivalent disparity or tie condition may be a miscompare on voted data whereby data supplied by the two processor elements **104**A, **104**B does not match on a programmed input/output (PIO) action or a direct memory access (DMA) action. Other equivalent disparity or tie conditions include a first processor element performing a PIO read while a second processor element performs a PIO write, or first and second processors reading or writing different addresses.

[0021] Referring to **FIG. 2**, a schematic block diagram depicts an embodiment of a computing system **200** with capability to resolve disparity among a plurality of processor elements **202**A, **202**B configured in a redundant-processor arrangement. The computing system **200** further includes a probation vector **204** coupled to the processor elements **202**A, **202**B and has a signal allocated to each of the processor elements **202**A, **202**B. A control element **206** is coupled to the processor elements **202**A, **202**B and evaluates processor fidelity, setting the probation vector **204** according to results of the evaluation.

[0022] The probation vector **204** is used to monitor secondary considerations of processor element fidelity before an error is detected, when more abundant information relating to processor element conditions and functionality are available. In contrast, a system that does not begin acquiring status information until an error is detected may have more limited functional capabilities, and a possible inability to perform actions diagnostic of processor element fidelity. Acquisition of status and operational information while the processor elements **202**A, **202**B are executing in due course simplifies operation because information is merely noted when available. Higher complexity algorithms that execute following error or disparity detection and require information to be evoked or stimulated, as well as monitored, can be avoided.

[0023] The computing system **200** also includes a voter **208** that is coupled to the plurality of processor elements **202**A, **202**B and compares actions taken by the processor elements **202**A, **202**B to determine disparity in processor actions. A control element **210** responds to disparity among the processor elements **202**A, **202**B based on the probation vector **204**. In some embodiment, the control element **210** can interject a delay between disparity detection and computer device operation termination to allow monitoring of additional information that may be useful in resolving disparity and breaking ties. The control element **210** can also determine whether evaluation of processor fidelity is insufficient to resolve the disparity among the processor elements **202**A, **202**B and, if so, terminate computing system operations.

[0024] Referring to **FIG. 3**, a schematic block diagram illustrates an embodiment of a computing system **300** including a plurality of processor elements **302**A, **302**B configured in a redundant-processor arrangement, and control logic **304** that imposes a selected-duration short delay in the event of a disparity or tie condition. The control logic **304** compares actions taken by the processor elements **302**A, **302**B and determines disparity in the actions, then waits the selected delay duration after equivalent disparity detection before initiating an action in response to the disparity condition. The control logic **304** may respond after the delay according to evaluation of secondary considerations of processor fidelity.

[0025] The selected delay has duration sufficient to enable near-simultaneous arrival of information for usage in resolving the disparity condition. The delay is imposed in case of processor disparity or tie, to enable simultaneous or near-simultaneous arrival of information that can be used in disparity resolution and/or tie-breaking. The delay has suit-

able duration to enable logic to receive a high priority interrupt and perform a few computations and is sufficiently long to enable information to be acquired at the same time or very close to occurrence of an error. Typical delay duration, using current processor operating speeds and technology, is on the order of tens or hundreds of microseconds, sufficient to handle the interrupt and execute hundreds or thousands of instructions. The delay may assist in avoiding or breaking race conditions.

[**0026**] The selected delay also has an upper limit. A lengthy examination of error state or diagnostic execution may not be acceptable. During the time between disparity and selection of the winning processor, the parallel input/output (PIO) operation or direct memory access (DMA) operation is suspended, possibly disrupting communications with other processors as well as communications on the interprocessor network due to backpressure. Such disruption is generally not desirable and is avoided by imposing an upper limit on the delay duration.

[**0027**] Referring to **FIG. 4**, a schematic block diagram shows an embodiment of a processor complex **400** within which the illustrative error handling system may be implemented. In a redundant-processor arrangement, the processor complex **400** includes a plurality of logical processors **408**, each a computing engine capable of executing processes and implemented using one or more processor elements **402**, each in a different processor slice **410**, combined with one or more logical synchronization units (LSUs) **412**. Each processor element **402** is a single microprocessor or microprocessor core capable of executing a single instruction stream. A processor slice **410** typically comprises one or more processor elements **402**, each with a dedicated memory **414** or sharing a partitioned memory. A processor complex **400** comprises one or more logical processors **408**. A processor complex **400** comprises one or more processor slices **410**. Within a complex **400**, each slice **410** has the same number of processor elements.

[**0028**] A processor complex with one slice is called a simplex complex. A two-slice processor complex is called a duplex, dual modular redundant, or DMR complex. A three-slice processor complex is called a triplex, tri-modular redundant, or TMR complex. A processor complex **400** includes both processor elements **402** and corresponding logical synchronization units **412**.

[**0029**] A computing system comprises one or more logical processors **408**. The computing system also comprises one or more processor complexes **400**. The processor complexes **400** are interconnected via a network, for example a System Area Network (SAN), a local area network (LAN), a wide area network (WAN), or the like, or simply a connection to a bus. The network is used for connection to both other processors and to input/output (I/O) devices. Voting or output data comparison is performed for all data transfers between the logical processor and the network or the network I/O adapter.

[**0030**] In a logical processor **408**, one, two, three, or possibly more processor elements cooperate to perform logical processor operations. Cooperative actions include coordinating or synchronizing mutually among the processor elements, exchanging data, replicating input data, and voting on operations and output data selection. In the illustrative embodiment, the various cooperative actions can be implemented within or supported by implementation of the logical synchronization units **412**.

[**0031**] Referring to **FIG. 5**, a schematic block diagram illustrates an embodiment of a computing system **500** comprising a plurality of processor elements **502**A, **502**B configured in a redundant-processor arrangement, and a voter **504** coupled to the plurality of processor elements **502**A, **502**B that compares actions taken by the processor elements and determines disparity in the actions. The computing system **500** further includes a control element **506** coupled to the processor elements **502**A, **502**B and the voter **504** that detects equivalent disparity among the processor elements **502**A, **502**B and responds by evaluating secondary considerations of processor fidelity. The illustrative computing system **500** may comprise a single logical processor of multiple logical processors in a complex or system.

[**0032**] The computing system **500** may also include a logical synchronization unit **512** comprising the voter **504** and a network interface **520**, such as a SAN interface that can be configured to connected to the network **518** by one or more ports. For example the network connection may be made as shown in **FIG. 5** via an X-fiber port and a Y-fiber port, although wire ports may otherwise be used.

[**0033**] The logical synchronization unit **512** maintains multiple bit sets that represent, at any point in time, the set of processor elements that are enabled to perform selected operations. The voter **504** includes a plurality of multiple-bit configuration registers to indicate which processor elements are expected and enabled to participate in selected operations, for example including programmed input/output (PIO) operations with the processor elements **502**A, **502**B, and network interface **520** DMA operations including direct memory access (DMA) output voting and DMA input replication.

[**0034**] The voter configuration bits represent which processor elements are meant to be "assigned" to a logical processor and are therefore eligible for performing various operations such as output voting operations. Configured processor elements are defined, for any particular operation type, as the set of processor elements enabled by the configuration bits in the logical synchronization unit **512** to perform that operation type. The logical synchronization unit **512** ignores operations of non-configured processor elements on an operation-type basis. Configuration bits are set whenever processor elements "join" the configuration of a logical processor **508**. Configuration bits are reset or cleared when processor elements leave the configuration, for example by being voted out.

[**0035**] Participating processor elements are the set of configured processor elements, for a particular instance of an operation, that actually perform the operation at a particular time, within a timeout period. The set of configured processor elements is related to a type of operation while the set of participating processor elements is related to an individual instance of an operation. The configured processor elements have outbound data transfers voted. Participating processor elements are the elements that actually issue a particular outbound data transfer.

[**0036**] For a particular voted operation, all configured processor elements are expected to participate. A processor element that does not participate times out on the operation.

[0037] Generally, voted operations can result in conditions including full agreement, timeout, simple miscompare, tie miscompare, and full miscompare. In full agreement, all configured processor elements participate within a reasonable time and supply identical data for an identical operation so that all voted data matches. In a timeout condition, one or more configured processor elements do not participate in time. For a simple miscompare, a strict majority of configured processor elements supplies identical data, and a strict minority supplies different data. A strict majority is greater than half, for example one in simplex, two in duplex, and two or three in triplex. A tie miscompare occurs with an even number of processor elements, for example two, configured in which the data does not compare. For a full miscompare, all sets of voted data, for example all three in triplex, miscompare pair-wise so that no strict majority results. The different types of conditions are mapped into one of three error conditions including no error, minor error, and major error.

[0038] For no error, the operation proceeds and no error is reported. For a minor error, the operation proceeds but an error is reported. Minor errors include such timeouts or disagreements among voted data in which available information is sufficient to enable resolution. Triplex configurations in which two processors are in agreement and duplex configurations in disagreement when other information is available to resolve the disagreement are examples of minor error conditions. For a major error, the operation does not proceed, but rather is aborted, and an error is reported. Major errors include such timeouts and disagreements among voted data that the condition cannot be resolved. Triplex configurations with three-way disagreement or timeouts in two of the three processors, and duplex configurations with disagreeing processors and no available information for resolution are examples of major error conditions.

[0039] In addition to timeouts and disagreement among redundant processors, other errors that may be handled using the illustrative techniques include breaks in cabling, for example between the processor elements and voter. Such errors can often be self-signaling. Self-signaling errors are defined as errors detected by an explicit detection element or mechanism as distinguished from implicit detection techniques such as voting. Various types of errors may be self-signaling. Examples of self-signaling errors include direct memory access (DMA) read timeouts, errors detected by parity or other error checking codes, loss-of-signal in optical signals, and loss of electrical continuity for electrical signals. With respect to the various illustrative systems, voting detects errors, but in a duplex configuration voting alone does not distinguish which voted data is correct and which incorrect. Self-signaling errors designate which of the two data suppliers is incorrect.

[0040] The processor elements **502**A, **502**B each have a memory **514** or share a partitioned memory with other processor elements in the same slice.

[0041] The PIO operations are processor-initiated reads (loads) or writes (stores) to any part of the processor's address space other than "main memory". The address space may contain registers, pseudo-registers, and memory other than main memory. PIO operations may be targeted to resources in either the voter **504** or to a network interface **520**. Operations targeted to the voter **504** may be either unvoted or voted (compared), depending on the address of the register being accessed. Operations to the network interface **520** may always be voted in some implementations.

[0042] When any configured processor element initiates a voted PIO read, the voter **504** captures the operation and read address and sets a timer. The voter **504** waits for all configured processor elements to perform the same operation. When all configured processor elements initiate the operation, the operation and address are compared, for example a bit-by-bit comparison of the entire operation and address. If one or more of the configured processor elements fails to initiate the operation within a configurable timeout period, a PIO timeout condition exists. In circumstances that the illustrative error handling and tie-breaking technique is not implemented or is disabled, operation can be described as follows. If all configured processor elements participate, then for the case of full agreement or simple miscompare, the operation proceeds, ignoring miscompared data, if any. A simple miscompare is handled as a minor error. Otherwise (not full agreement or simple miscompare), the operation is aborted—not performed, and a "bus error" is returned to all requesting processor elements, and a major error is reported. If the operation is not aborted, then if the PIO read is targeted to the network interface **520**, the voter **504** forwards the operation and address to the network interface **520** and waits for a response. If the operation is targeted to the voter **504**, then the voter **504** accesses data directly. The response data, when available, is replicated by the voter **504** and sent as a response to all participating processor elements at approximately the same instant.

[0043] When any configured processor element initiates a voted PIO write operation, the voter **504** captures the operation, write address, and write data, then sets a timer and waits for all configured processor elements to perform the same operation. When all configured processor elements initiate the operation, the operation, write address, and write data are bit-by-bit compared. If one or more of the configured processor elements fails to initiate the operation within a configurable timeout period, a PIO timeout condition exists. In circumstances that the illustrative error handling and tie-breaking technique is not implemented or is disabled, the operation is as follows. If all configured processor elements participate, in the case of full agreement or simple miscompare, the operation proceeds, ignoring miscompared data. A simple miscompare is handled as a minor error. Otherwise (not full agreement or simple miscompare), the operation is aborted—not performed. No direct response is made to the processor element because no response or acknowledgement is normally made to write operations. When the operation is aborted, a major error is reported. An operation that is not aborted is handled according to the target address of the write operation. For a PIO write to the network interface **520**, the voter **504** forwards the operation, address, and data to the network interface **520**. For a PIO write to the voter **504**, the voter **504** performs the write operation directly. No response is made to the processor element. Because of the possible side-effects with PIO write operations, and because software does not necessarily verify the effect, or success, of each write operation, when a PIO write operation is aborted due to a major voting error, the voter **504** suspends all future PIO write operations, which are also aborted, until the software detects the error and re-enables PIO voting. Typically the error is detected by

handling an error interrupt. Note, however, that in the triplex case for a simple miscompare such as one processor element write and two processor elements time out, no requirement is made to abort all future programmed I/O write operations.

[**0044**] PIO operations are initiated by the processor elements, as contrasted to DMA operations which are initiated by the network interface **520**. Therefore, PIO timeouts are possible in two different circumstances. In a first circumstance, one or two processor elements, operating correctly, initiate a PIO operation, and one processor element, operating incorrectly or stopped, fails to perform the PIO operation. The error is detected when the timer expires. Without further information, the processor element operating incorrectly may be indeterminable, for example when two processor elements are configured and one times out.

[**0045**] In a second circumstance, one processor element, operating incorrectly, initiates a PIO operation that should not occur, and the other processor element or processor elements, operating correctly, do not initiate a PIO operation. Again the error is detected when the timer expires, although without further information the processor element operating incorrectly may be indeterminable, for example for two active processor elements, one of which times out. Accordingly, the processor elements that do not participate are not necessarily incorrect.

[**0046**] In the triplex case, the strict majority is always trustworthy so that if one processor element times out and the remaining two processor elements perform a PIO operation, then the processor element that times out is in error and ignored, while the PIO operation proceeds and the data voted with a minor error indicated. Also in the triplex case, if two processor elements time out when a single processor element performs a PIO operation, then the processor element that performs the PIO operation is in error and the PIO is ignored. The lone processor element can be considered a "rogue processor element" and the attempted PIO operation is called a "rogue operation".

[**0047**] In the duplex case, whether the processor element performing the PIO or the processor element that is not participating is in error cannot be determined, without other evidence. Accordingly a tie or disparity condition exists.

[**0048**] Some embodiments of the computing system **500** further comprise a two-processor elements configuration **502**A, **502**B, the programmed input/output (PIO) interface **522**, and a direct memory access (DMA) interface **524** coupled to the voter **504**. An action disparity that is detectable by the control element **506** is a miscompare on voted data with non-matching data supplied by two processor elements **502**A, **502**B either on a PIO action or a DMA action.

[**0049**] Direct memory access (DMA) reads are outbound operations initiated by the network interface **520**. The voter **504** replicates and forwards the DMA read operation and address to all configured processor elements at approximately the same instant, subject to congestion delays in the different slices that may cause an operation to arrive at the processor elements at slightly different times. The voter **504** then starts a timer and waits for the responses. Response data flows from the processor elements **502**A, **502**B, through the voter **504**, to the network interface **520**. Responses arriving from the configured processor elements are buffered in the voter **504** rather than being sent immediately to the network interface **520**. The later responses, upon arrival, are compared with the earlier responses saved in the data buffers. When a strict majority of the responses agree, one copy of the data, from one of the agreeing responses, is communicated to the network interface **520**. If a strict majority of the responses do not agree, then data is not sent to the network interface **520** in a manner that can be interpreted as valid data.

[**0050**] Any processor element that does not respond within the timeout period is declared to have timed out. Unlike PIO timeouts, no rogue situations can occur with DMA timeouts because the DMA operation is initiated through the network interface **520**. Accordingly, if one processor element times out, that processor element is necessarily erroneous in both duplex and triplex cases. The condition is considered a minor error, and the DMA operation proceeds using data supplied by the processor element or processor elements that do not time out. If two processor elements time out, or the only processor element in a simplex case, then a major error is indicated and the DMA operation is aborted. The voter **504** generates an error notification interrupt to all configured processor elements in the case of any disagreement or timeout, whether data is successfully forwarded to the network interface **520** or otherwise. The interrupt indicates which processor elements did time out, if any, and all comparison results.

[**0051**] Direct memory access (DMA) writes are inbound operations with data flowing from the network interface **520** through the voter **504** to the processor elements **502**A, **502**B. DMA write operations are initiated by the network interface **520**. The voter **504** replicates and forwards the operation, address, and data to all configured slices at approximately the same instant. No response is made to the network interface **520**.

[**0052**] The voted PIO operations and DMA responses are protected by timeouts. PIO and DMA timeout values are both configurable by control logic, such as software, and may be different.

[**0053**] For PIO timeouts, the timer is started when the first PIO operation arrives. The timer is restarted when each operation arrives, giving a full timeout period for the later arrivals relative to the earlier ones, a behavior used in all implementations due to the possibility of a PIO operation being initiated by a "rogue processor element". PIO operations can never time out in the simplex case because the operation originates from the processor element.

[**0054**] For DMA read response timeouts, DMA operations can time out in simplex, duplex, and triplex configurations because the operation is originated from the network interface **520**. The timer is started when the DMA request is forwarded by the voter **504** to memories of all processor elements. The timer may optionally be restarted when each response arrives, giving a full timeout period for the later responses. Alternatively, a single timeout interval may be applied to all configured processor elements. Either option is possible since no rogue DMA operations can occur.

[**0055**] In an illustrative embodiment, special case handling can be used when a PIO timeout occurs or a miscompare occurs on voted data in a duplex configuration. In the illustrative example, duplex tie handling generally is inap-

plicable to DMA timeouts, and to simplex or triplex configurations. Two disparity or tie conditions include a PIO timeout in which one processor element performs a PIO operation and the other processor element does not, and a miscompare on voted data in which data supplied by the two processor elements does not match, either on a PIO operation or a DMA read operation.

[**0056**] In the absence of any other information, a tie or disparity condition in a duplex configuration is ambiguous whereby the trustworthiness of each processor element is not obvious, leading to a typical policy of halting the logical processor.

[**0057**] However, occasionally, other information, termed secondary considerations of processor fidelity, may exist. The other information may be considered sufficiently strong, even if circumstantial, to implicate one of the two processor elements. For example, a recent logged history of other detected recoverable errors may be indicative of a degradation of processor element reliability.

[**0058**] Another example of pertinent reliability information is a recent history of processor replacement. A newly-reintegrated processor element or a new slice may be a more likely source of error than an element that has long been installed without a history of error. Such early life problems are frequently discovered within a short time, on the order of minutes, following installation.

[**0059**] A further example of reliability information is inherent in the multiple-dimensional configuration of logical and physical processors, for example as shown in **FIG. 4**. Processor slices with multiple processor elements connected physically but not logically include hardware that is shared within a processor slice. Hardware may be shared among logical processors. When shared hardware ceases functioning correctly, errors such as intermittent errors can occur. If errors are detected in one logical processor but not another, information about the errors can be communicated between processor elements in a processor slice so that all processor elements within the slice have sufficient information to break any ties that occur.

[**0060**] Some embodiments of the computing system **500** further comprise a probation vector **526** coupled to the voter **504** and coupled to the processor elements **502**A, **502**B. The probation vector **526** holds a signal for each processor element **502**A, **502**B. A processor control policy executable on the processor elements **502**A, **502**B evaluates the secondary conditions of processor fidelity during processor element execution. The processor control policy sets bits in the probation vector according to the evaluation of secondary considerations before determining whether a major or minor error has occurred. In a particular embodiment, the probation vector **526** enables implementations to supply the other information to the voter **504** in such a way that duplex tie conditions can be simply resolved. The probation vector **526** comprises one bit for each processor element **502**A, **502**B. The logical synchronization unit **512** uses the probation vector **526** as a tie-breaker in some conditions.

[**0061**] In an illustrative embodiment, PIO writes to the probation vector **526** are not voted. The initial or reset value of the probation vector **526** is all zero. Each processor element **502**A, **502**B can set or reset the probation bit only for the processor slice associated with the processor element, and not for any other slice. The probation vector **526** is not used in simplex and triplex modes for which the bits may still be set by the control logic, but are ignored.

[**0062**] The probation vector **526** is ignored if both processor slices agree, indicating no error. The probation vector **526** is also ignored in the case of self-signaling errors, which indicate the error source so that tie-breaking is superfluous. Self-signaling errors are thus classified as minor errors. The errors occur when the voter **504** can be certain that one slice should be ignored, for example in an outbound DMA read response, when one slice does not respond and times out, or when the slice supplies data marked as "known bad". "Known bad" data relates to another example of self-signaling error and is data returned from a processor element's memory or cache that generates a detected error, such as a parity or other detected error, when accessed.

[**0063**] Control logic in each processor element sets the associated probation bit independently of the other processor elements; reaching an agreement among processor elements is not required. Accordingly, both probation bits may be asserted at any time. The control logic resets the probation bits after some amount of time. The time duration is an implementation-defined parameter or policy.

[**0064**] In some circumstances, the probation bits may be set for all processor elements on a processor slice due to an error that places behavior of the entire slice in doubt. Accordingly, the control logic can propagate the probation bits from one processor element, where an error has been detected, to other processor elements in the slice by an implementation-defined technique. Examples of an implementation-defined system include exchanging probation bits via a register in a slice application-specific integrated circuit (ASIC), and/or using inter-processor, intra-slice interrupts.

[**0065**] In some embodiments of the computing system **500**, the control element **506** interjects a delay between equivalent disparity detection and computing device operation termination. When a duplex, non-self-signaling error occurs, a delay is imposed prior to declaring the situation an error condition. During the delay period, operation is held in limbo. The operation is neither completed nor aborted. The delay enables the tie to be broken by control logic setting the probation bit after the error but before the timeout elapses. The delay does not occur, and therefore does not add any latency, in full agreement cases, and in simplex and triplex cases.

[**0066**] The delay period in a tie-breaker situation is maintained sufficiently small to avoid excessive network back-pressure, therefore preventing an increase in congestion in the network. For current technology, a sufficiently small delay may be of the order of the range of tens or hundreds of microseconds.

[**0067**] In an illustrative embodiment, when the delay period begins, the logical synchronization unit **512** sends an interrupt to all participating slices to inform the slices that a miscomparison has occurred, although an error has not been declared. The interrupt is in addition to the interrupt that is generated at the end of the delay, when the voting error is final, either major or minor.

[**0068**] In a tie-breaker or disparity condition, if after the delay interval has elapsed, both probation bits are enabled or both probation bits are disabled, then a major error exists

and the operation is aborted. If the bits are in opposite states, then the slice with the probation bit off or reset is obeyed, and the slice with the probation bit is on or set is ignored. If the probation bits are used to break the tie, then a minor error is declared.

[0069] The policy for setting probation bits and duration that the bit setting is maintained is implementation-specific.

[0070] In the illustrative embodiment, the logical synchronization unit **512** reports all errors, both major and minor, to control logic, such as software via an interrupt and status register. In one implementation, status register bits indicate that the tie-break mechanism has been invoked and designate which processor element or slice is obeyed.

[0071] Referring to **FIG. 6**, a flow chart depicts an embodiment of an error handling method **600** in a redundant-processor computing device during programmed input/output (PIO) voting in a duplex configuration. The method **600** comprises detecting equivalent disparity **612**, and **610**, **614** among processor elements of the computing device, and responding to the detected equivalent disparity by evaluating **624**, **634** secondary considerations of processor fidelity.

[0072] The method can further comprise determining whether evaluation **624**, **634** of the secondary considerations is insufficient to resolve the disparity among the processor elements. If so, computing device operations are terminated **626** and **628**, **636** and **638**. Delay can be inserted **620** and **622**, **630** and **632** between equivalent disparity detection and computer device operation termination **626**, **636**.

[0073] In the illustrative method **600**, control logic receives **602** a programmed input/output (PIO) request from one processor element (PE), buffers **604** the request, and starts **606** a timer. If a second request is received **608**, the two requests are compared **610**. Otherwise, if the timer has not elapsed **612**, whether the second request is received is determined **608**. If the timer has elapsed **612**, analysis of secondary considerations of processor fidelity begins.

[0074] In conditions that a second request is received **608**, following comparison **610** of the requests, if the requests match **614**, the control logic performs the PIO operation **616** and no error is indicated, and the method is complete **618**. Otherwise, an equivalent disparity condition exists in the form of a miscompare on voted data whereby command, address, or data supplied by two processor elements does not match on a programmed input/output (PIO) action. Operation, address, and, for a write operation, data are compared to determine a match condition. Secondary consideration analysis begins with the control logic sending **620** a "tie break pending" interrupt to the processor elements, and waiting **622** the configured time. Generally, the secondary conditions of processor fidelity can be evaluated during processor element execution, and a probation vector can be set according to the evaluation prior to determination of major or minor error. If probation bits are equal **624**, the operation is aborted **626** since the tie or disparity condition cannot be resolved and a major error interrupt is sent to the processor elements. The method is terminated **628**.

[0075] Otherwise, if the probation bits are not equal **624**, the control logic follows direction **648** of the processor element that is not on probation, and the PIO operation specified by the non-probation processor element is performed. A minor error interrupt is sent **650** to the processor elements, and the method completes **652** with a suitable minor error handling technique. In some embodiments, the minor error is addressed by marking the loser of the voting decision as no longer participating in the logical processor. Subsequent input/output operations or other accesses to the voter are ignored. In other embodiments, software processing in the loser is interrupted and software executing in remaining processor elements shuts down the offending processor element.

[0076] For analysis of secondary considerations of processor fidelity after the timer elapses **612**, an equivalent disparity condition occurs in which a first processor element performs a programmed input/output (PIO) action while a second processor element does not. Control logic sends **630** a "tie break pending" interrupt to the processor elements, and waits **632** the configured time. If probation bits are equal **634**, the operation is aborted **636** since correct operation cannot be determined. A major error interrupt is sent to the processor elements. The method is terminated **638**.

[0077] Otherwise, if the probation bits are not equal **634**, control logic determines whether the processor element requesting the PIO operation is on probation **640**. If the processor element is on probation **640**, the control logic follows direction **642** of the processor element that is not on probation and ignores or aborts the PIO operation, sends **644** a minor error interrupt to the processor elements, handles the minor error, and terminates **646** the method. If the processor element requesting the PIO is not on probation **640**, the control logic follows direction **648** of the processor element that is not on probation, and the PIO operation specified by the non-probation processor element is performed. A minor error interrupt is sent **650** to the processor elements, and the method is complete **652**.

[0078] Referring to **FIG. 7**, a flow chart depicts an embodiment of an error handling method **700** in a redundant-processor computing device during direct memory access (DMA) read voting in a duplex configuration. The method **700** comprises detecting equivalent disparity **726** and **728** among processor element memories of the computing device, and responding to the detected equivalent disparity by evaluating **738** secondary considerations of processor fidelity.

[0079] The method can further comprise determining whether evaluation **738** of the secondary considerations is insufficient to resolve the disparity among the processor elements. If so, computing device operations are terminated **740** and **742**. Delay can be inserted **736** between equivalent disparity detection and computer device operation termination **740**, **742**.

[0080] In the illustrative method **700**, control logic receives **702** a direct memory access (DMA) read request from a network agent such as a system area network (SAN) agent, replicates and forwards **704** the request to both processor elements, and starts **706** a timer. If a first response is received **708**, the first response is buffered **716**. In some embodiments, the timer may be restarted as the first response is buffered **716**. If the timer has not timed out **710**, whether the first response is received is again determined **708**. If the timer has timed out **710**, the operation is aborted **712** and a major error interrupt is sent to both processor elements. The major error interrupt is indicative of a double timeout. The method is then terminated **714**.

[0081] In conditions that the first response is received **708** and the first response is buffered **716**, whether a second response has been received is determined **718**. If the second response has been received **718**, the first and second response data are compared **726**. Otherwise, if the timer has not timed out **720**, whether the second response has been received is again determined **718**. If the timer has timed out **720**, the operation is completed **722** using data from the first response and a minor error interrupt is sent to both processor elements. The single timeout condition is indicative of a "self-signaling error". The method is then terminated **724**.

[0082] In conditions that the second response is received **718** and data from the first and second responses is compared **726**, if the first and second response data are equal **728** the data match so that the operation is completed **730** with no error. The method completes successfully **732**. Otherwise, the first and second response data are unequal **728** and a "tie break pending" interrupt **734** is sent to the processor elements. A delay is inserted **736** to wait for a configured time. Probation bits are read to determine whether the probation bits are equal **738**. If so, the operation is aborted **740** since the tie cannot be resolved using the probation bits and a major error interrupt is sent to the processor elements. The method terminates unsuccessfully **742**. Otherwise, the probation bits are not equal **738** and the operation is completed **744** using the response from the processor element that is not on probation. A minor error interrupt is sent **746** to the processor elements, the minor error handled by marking the loser for removal or shutting down the offending processor element, and the method is terminated **748**.

[0083] While the present disclosure describes various embodiments, these embodiments are to be understood as illustrative and do not limit the claim scope. Many variations, modifications, additions and improvements of the described embodiments are possible. For example, those having ordinary skill in the art will readily implement the steps necessary to provide the structures and methods disclosed herein, and will understand that the process parameters, materials, and dimensions are given by way of example only. The parameters, materials, and dimensions can be varied to achieve the desired structure as well as modifications, which are within the scope of the claims. For example, although the illustrative structures and methods are most applicable to multiple-processor systems in a duplex configuration, various aspects may be implemented in configurations with more or fewer processors. Furthermore, the illustrative embodiments depict particular arrangements of components. Any suitable arrangement of components may be used. The various operations performed may be implemented in any suitable matter, for example in hardware, software, firmware, or the like.

What is claimed is:

1. A control apparatus for usage in a redundant-processor computing device including a plurality of processor elements, the control apparatus comprising:

a control element that detects equivalent disparity among the processor elements and responds by evaluating secondary considerations of processor fidelity.

2. The apparatus according to claim 1 further comprising:

a control element that determines whether evaluation of the secondary considerations is insufficient to resolve the equivalent disparity among the processor elements and, if so, terminates operations of the computing device.

3. The apparatus according to claim 2 further comprising:

a control element that interjects a delay between equivalent disparity detection and the evaluation of secondary considerations of processor fidelity.

4. The apparatus according to claim 1 further comprising:

a control element that determines whether the evaluation of the secondary considerations is sufficient to resolve the equivalent disparity among the processor elements and, if so, completes an operation according to the resolution.

5. The apparatus according to claim 1 further comprising:

a control element that evaluates the secondary conditions of processor fidelity and sets a probation vector according to the evaluation.

6. The apparatus according to claim 1 further comprising:

a processor element that evaluates the secondary conditions of processor fidelity and sets a probation vector according to the evaluation.

7. The apparatus according to claim 1 wherein an equivalent disparity condition comprises one or more of conditions including:

a condition of a first processor element performing a programmed input/output (PIO) action while a second processor element does not;

a condition of a first processor element performing a PIO read while a second processor element performs a PIO write;

a condition of a first processor and a second processor reading different addresses;

a condition of a first processor and a second processor writing different addresses; and

a miscompare on voted data whereby data supplied by two processor elements does not match on a programmed input/output (PIO) action or a direct memory access (DMA) action.

8. An error handling method in a redundant-processor computing device comprising:

detecting equivalent disparity among processor elements of the computing device; and

responding to the detected equivalent disparity by evaluating secondary considerations of processor fidelity.

9. The method according to claim 8 further comprising:

determining whether the evaluation of the secondary considerations is insufficient to resolve the equivalent disparity among the processor elements; and

if so, terminating operations of the computing device.

10. The method according to claim 9 further comprising:

inserting a delay between equivalent disparity detection and termination of computer device operation.

11. The method according to claim 8 further comprising:

determining whether evaluation of the secondary considerations is sufficient to resolve the equivalent disparity among the processor elements; and

if so, completing an operation according to the resolution.

12. The method according to claim 8 further comprising:

evaluating the secondary conditions of processor fidelity; and

setting a probation vector according to the evaluation.

13. The method according to claim 8 wherein an equivalent disparity condition one or more of conditions including:

a condition of a first processor element performing a programmed input/output (PIO) action while a second processor element does not;

a condition of a first processor element performing a PIO read while a second processor element performs a PIO write;

a condition of a first processor and a second processor reading different addresses;

a condition of a first processor and a second processor writing different addresses; and

a miscompare on voted data whereby data supplied by two processor elements does not match on a programmed input/output (PIO) action or a direct memory access (DMA) action.

14. A computing system comprising:

a plurality of processor elements configured in a redundant-processor arrangement;

a voter coupled to the plurality of processor elements that compares actions taken by the processor elements and determines disparity in the actions; and

a control element coupled to the processor elements and the voter that detects equivalent disparity among the processor elements and responds by evaluating secondary considerations of processor fidelity.

15. The system according to claim 14 further comprising:

a two-processor element configuration; and

a programmed input/output (PIO) interface coupled to the voter whereby an action disparity that is detectable by the control element is a PIO timeout with one processor element performing a PIO action and one processor element not performing the PIO action.

16. The system according to claim 14 further comprising:

a two-processor element configuration; and

a programmed input/output (PIO) interface and a direct memory access (DMA) interface coupled to the voter whereby an action disparity that is detectable by the control element is a miscompare on voted data with non-matching data supplied by two processor elements either on a PIO action or a DMA action.

17. The system according to claim 14 further comprising:

a probation vector coupled to the voter and coupled to the processor elements and having a signal allocated to each of the processor elements; and

a control element that evaluates the secondary conditions of processor fidelity and sets the probation vector according to the secondary considerations of processor fidelity.

18. The system according to claim 14 further comprising:

a control element that determines whether evaluation of the secondary considerations is insufficient to resolve the disparity among the processor elements and, if so, terminates computing device operations.

19. The system according to claim 18 further comprising:

a control element that interjects a delay between equivalent disparity detection and evaluation of secondary considerations of processor fidelity.

20. A computing system comprising:

a plurality of processor elements configured in a redundant-processor arrangement;

a probation vector coupled to the processor elements and having a signal allocated to each of the processor elements; and

a logic that evaluates processor fidelity and sets the probation vector according to the evaluation.

21. The system according to claim 20 further comprising:

a control element that evaluates processor fidelity and sets a probation vector according to the evaluation.

22. The system according to claim 20 further comprising:

a processor element that evaluates processor fidelity and sets a probation vector according to the evaluation.

23. The system according to claim 20 further comprising:

a voter coupled to the plurality of processor elements that compares actions taken by the processor elements and determines disparity in the actions; and

a control-element that responds-to disparity among the processor elements based on the probation vector.

24. The system according to claim 23 further comprising:

a control element that interjects a delay between disparity detection and computer device operation termination.

25. The system according to claim 20 further comprising:

a control element that determines whether evaluation of processor fidelity is insufficient to resolve the disparity among the processor elements and, if so, terminates computing system operations.

26. The system according to claim 20 wherein for a two-processor system a disparity condition comprises one or more conditions selected from a group consisting of:

a condition of a first processor element performing a programmed input/output (PIO) action while a second processor element does not;

a condition of a first processor element performing a PIO read while a second processor element performs a PIO write;

a condition of a first processor and a second processor reading different addresses;

a condition of a first processor and a second processor writing different addresses; and

a miscompare on voted data whereby data supplied by two processor elements does not match on a programmed input/output (PIO) action or a direct memory access (DMA) action.

27. A computing system comprising:

a plurality of processor elements configured in a redundant-processor arrangement; and

control logic coupled to the processor element plurality that mutually compares actions taken by ones of the processor elements and determines equivalent disparity in the actions, and waits a selected delay after equivalent disparity detection before initiating an action responsive to the disparity condition.

28. The computing system according to claim 27 further comprising:

control logic that responds after the delay according to evaluation of secondary considerations of processor fidelity.

29. The computing system according to claim 27 wherein:

the selected delay has a duration sufficient to enable near-simultaneous arrival of information for usage in resolving the disparity condition.

* * * * *