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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
20 October 2011 (20.10.2011)

PCT

(10) International Publication Number
WO 2011/130376 A2

(51) International Patent Classification:
G06F 17/50 (2006.01) *G06F 19/00* (2011.01)

(21) International Application Number:
PCT/US2011/032268

(22) International Filing Date:
13 April 2011 (13.04.2011)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

12/759,625	13 April 2010 (13.04.2010)	US
12/960,745	6 December 2010 (06.12.2010)	US
13/025,075	10 February 2011 (10.02.2011)	US

(63) Related by continuation (CON) or continuation-in-part (CIP) to earlier applications:

US	13/025,075 (CON)
Filed on	10 February 2011 (10.02.2011)
US	12/960,745 (CON)
Filed on	6 December 2010 (06.12.2010)
US	12/759,625 (CON)
Filed on	13 April 2010 (13.04.2010)

(71) Applicant *(for all designated States except US)*: **SYNOPSYS, INC.** [GB/US]; 700 East Middlefield Road, Mountain View, CA 94043 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **SRIPADA, Subramanyam** [IN/US]; C/o Synopsys, Inc., 700 East Middlefield Road, Mountain View, CA 94043 (US). **SINGHAL, Sonia** [IN/US]; C/o Synopsys, Inc., 700 East Middlefield Road, Mountain View, CA 94043 (US). **MIZE, Loa** [US/US]; C/o Synopsys, Inc., 700 East Middlefield Road, Mountain View, CA 94043 (US). **MOON, Cho** [US/US]; C/o Synopsys, Inc., 700 East Middlefield Road, Mountain View, CA 94043 (US).

(74) Agents: **PANWAR, Rajendra** et al.; FENWICK & WESST LLP, Silicon Valley Center, 801 California Street, Mountain View, CA 94041 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM,

[Continued on next page]

(54) Title: COMPARISON OF TIMING CONSTRAINTS FOR ELECTRONIC CIRCUITS

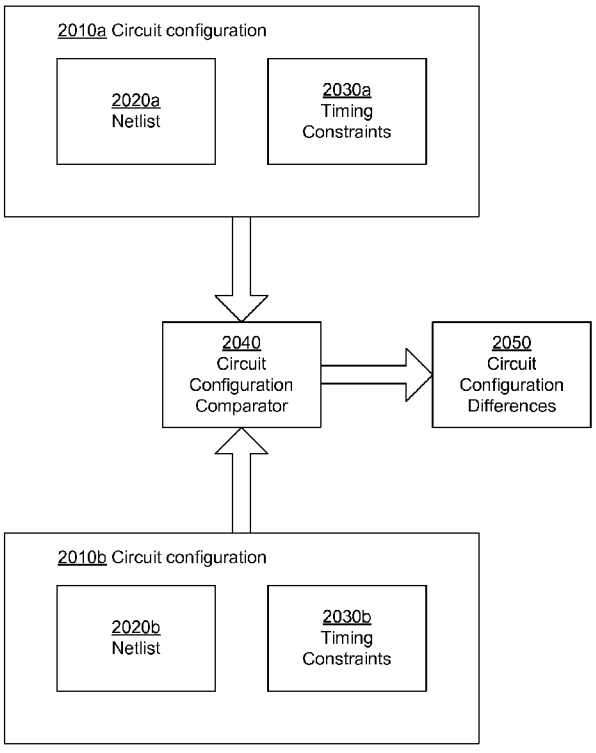


FIG. 20

(57) Abstract: Individual mode timing constraints associated with a set of netlists are combined into merged mode timing constraints. An initial merged mode constraint is generated by combining timing constraints from individual modes. Equivalence between the merged mode and the individual modes is verified by comparing timing relationships in the merged mode with timing relationships in the individual modes. In another aspect, timing behaviors associated with constraints of circuits are compared to identify mismatches between circuit configurations. Aggregate sets of timing relationships associated with timing nodes are determined for timing paths between start points and end points. Aggregate sets of timing relationships for corresponding timing nodes are matched to determine if mismatches exist between circuits.

WO 2011/130376 A2

AO, AT, AU, AZ, BA, BB, BG, BH, 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, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, 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, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, 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, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

COMPARISON OF TIMING CONSTRAINTS FOR ELECTRONIC CIRCUITS

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application is a continuation-in-part of U.S. Application Nos. 12/759,625 filed April 13, 2010, 12/960,745 filed Dec 6, 2010, and 13/025,075 filed Feb 10, 2011, all of which are incorporated by reference in their entirety.

FIELD OF THE INVENTION

[0002] This invention relates generally to electronic design automation (EDA) of circuits and in particular to comparison of timing constraints of circuits.

BACKGROUND

[0003] Due to the large number of components in state of the art electronic circuits, most of their design and production is computer implemented. An important operation performed during design of electronic circuits is timing analysis that validates timing performance of an electronic design.

[0004] One way to perform timing analysis is to use dynamic simulation which determines the full behavior of a circuit for a given set of input values. Timing analysis using dynamic simulation is highly computation intensive. A more efficient method for perform timing analysis validates timing performance of a circuit design by checking possible paths for timing violations. This method is called static timing analysis (STA) and is much faster than dynamic simulation since it does not simulate the logical operation of the electronic circuit. However, performing static timing analysis for large electronic circuits can take significant time.

[0005] Furthermore, timing analysis typically is repeated multiple times for the same electronic design for various combinations of modes and corners. Semiconductor device parameters can

vary with conditions such as fabrication process, operating temperature, and power supply voltage. A circuit fabricated using these processes may run slower or faster than specified due to variations in operating conditions or may even fail to function. Therefore timing analysis is performed for various operating conditions to make sure that the circuit performs as specified under these conditions. Such operating conditions for a circuit are modeled using corners that comprise a set of libraries characterized for process, voltage, and temperature variations.

[0006] The timing analysis of a circuit is also often repeated for different operating modes, for example, normal operating mode, test mode, scan mode, reset mode and so on. For example, a circuit used in a computer operates in a stand-by mode when the computer is in a stand-by mode. Similarly, during testing phase, a circuit may be operated in a test mode. A mode can be modeled using a set of clocks, input voltages, and timing constraints in similar operating conditions.

[0007] For performing timing analysis during implementation and sign-off of a circuit, designers usually verify a large number of modes and corners. Each circuit design may have tens of modes and tens of corners. Since each mode might be verified for each corner condition, the total number of scenarios in which the design is verified is the product of the number of modes and number of corners. This results in the timing analysis being performed a large number of times resulting in exorbitant costs.

[0008] One way to handle the large number of scenarios resulting from multiple modes and corners is to merge the modes into a smaller set, for example, a single mode. Since timing verification is performed for the combination of modes and corners, reduction in the number of modes reduces the total combinations of modes and corners by a much larger number. For example, if there are 10 modes and 10 corners, the total number of combination of modes and corners is $10 \times 10 = 100$. However if the 10 modes were combined to a single mode, the total number of combinations is reduced to $1 \times 10 = 10$ which is a 90% reduction in the number of combinations to be verified.

[0009] Conventionally modes are merged manually by designers. Furthermore, the manually merged modes are manually verified against the original set of modes or not verified. Due to the complexity of constraints associated with circuit designs, the generation of merged modes is difficult to handle manually. Since the number of timing constraints for a given netlist can be large, manual merging of modes can be error prone and have prohibitive costs. For example, a

circuit could have millions of lines of constraints and manually verifying correctness of merged constraints may not be practically feasible. Due to lack of confidence in the correctness of the merged modes, manually merged modes may be used during the implementation phase of the design but final sign-off of the design is performed using individual modes. Thus, there is a need for better approaches to comparing the timing constraints in a set of individual modes and those in an allegedly equivalent merged mode.

[0010] Similarly, there is also a need to compare timing constraints for different circuits that allegedly are equivalent (including, for example, different versions of the same circuit). For example, during the circuit design process, a later description of a circuit may be obtained by performing transformations on an earlier description of the circuit. It may be desirable to compare the timing constraints of the two circuits, for example to ensure that the timing constraints are equivalent and no changes were introduced as a result of the transformation. Timing constraints may also be compared for allegedly equivalent circuits produced by different sources.

SUMMARY

[0011] The above and other issues are addressed by a computer-implemented method, computer system, and computer program product for generating merged mode constraint from individual mode constraints associated with a set of netlists. A merged mode constraint is generated by combining timing constraints from individual modes. A set of timing relationships is determined for the merged mode as well as for each individual mode constraints. The timing relationship comprises timing information associated with a start point and an end point, for example, clock information, exception states, etc. The timing relationships of the merged mode are compared with the timing relationships of the individual modes to identify extraneous timing relationships present in the merged mode. Timing constraints are added to the merged mode to eliminate the extraneous timing relationships.

[0012] In another aspect, computer-implemented method, computer system, and computer program product verify equivalence of merged modes with respect to individual modes associated with a set of netlists. A source set of timing nodes and a sink set of timing nodes is identified for a given set of netlists. An aggregate set of timing relationships is obtained by

traversing from the source set of timing nodes to the sink set of timing nodes for the merged mode and each individual mode. The timing relationships present in the aggregate set for the merged mode are compared with the timing relationships in the aggregate sets for individual modes. If a timing relationship is identified that is present in a merged mode but is not present in any individual mode, an error is reported. If a timing relationship is identified in any individual mode that is not present in the merged mode, an error is reported.

[0013] In another aspect, computer-implemented method, computer system, and computer program product compare circuit configurations comprising timing nodes, edges between timing nodes and timing constraints. A first source set of timing nodes and a first sink set of timing nodes is identified in the first circuit configuration. A second source set of timing nodes and a second sink set of timing nodes is identified in the second circuit configuration such that the timing nodes in the second source set correspond to the first source set and the timing nodes in the second sink set correspond to the first sink set. A first aggregate set of timing constraints is determined for timing paths form the first source set to the first sink set. The aggregation of the timing constraints accounts for interactions between timing constraints of the timing paths. Similarly a second aggregate set of timing constraints is determined for timing paths from the second source set to the second sink set. A determination is made whether the first aggregate set of timing constraints is equivalent to the second aggregate set of timing constraints.

[0014] The features and advantages described in this summary and the following detailed description are not all-inclusive. Many additional features and advantages will be apparent to one of ordinary skill in the art in view of the drawings, specification, and claims hereof.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] FIG. 1 illustrates a merged mode obtained from multiple individual modes for a given circuit configuration, in accordance with an embodiment.

[0016] FIG. 2 illustrates timing relationships for a circuit associated with multiple modes, in accordance with an embodiment of the invention.

[0017] FIG. 3 illustrates an embodiment of the system architecture for automatic verification and generation of merged modes based on individual modes.

[0018] FIG. 4 shows a process illustrating how a merged mode is compared with individual modes with respect to different sets of start points and end points, in accordance with an embodiment.

[0019] FIG. 5 illustrates the steps for verifying whether each relationship present in every individual mode is present in the merged mode, in accordance with an embodiment.

[0020] FIG. 6 illustrates the steps for verifying whether each timing relationship present in the merged mode is present in at least one of the individual modes, in accordance with an embodiment.

[0021] FIG. 7 shows an example circuit configuration illustrating various terms related to a circuit.

[0022] FIGS. 8(a)-(d) illustrate the steps of computation of aggregate constraints at an end point of the circuit with respect to a set of start points, in accordance with an embodiment.

[0023] FIGS. 9(a)-(d) illustrate the steps of computation of aggregate constraints at a start point of the circuit with respect to an end point of the circuit, in accordance with an embodiment.

[0024] FIG. 10 illustrates the comparison of a reconvergent point in two circuit configurations, in accordance with an embodiment.

[0025] FIG. 11 illustrates the overall process for generating merged mode from individual modes for a netlist, in accordance with an embodiment.

[0026] FIG. 12 illustrates how constraints of a merged mode are determined from constraints of individual modes based on their impact on timing relationships, in accordance with an embodiment.

[0027] FIG. 13(a) illustrates the process for eliminating timing relationships between clocks in the merged mode that do not co-exist in any individual mode, in accordance with an embodiment.

[0028] FIG. 13(b) shows a circuit diagram to illustrate the process of elimination of timing relationships based on the flowchart shown in FIG. 13(a), in accordance with an embodiment.

[0029] FIG. 14 shows the flowchart for illustrating the steps for detecting and eliminating extraneous timing relationships in a merged mode, in accordance with an embodiment.

[0030] FIG. 15(a) illustrates the details of the first pass for eliminating extraneous timing relationships from the merged mode by disabling clock paths, in accordance with an embodiment.

[0031] FIG. 15(b) shows a circuit diagram to illustrate the process of elimination of extraneous timing relationships from the merged mode based on the flowchart shown in FIG. 15(a).

[0032] FIG. 16(a) illustrates the details of the second pass for eliminating extraneous timing relationships by disabling data paths in merged modes, in accordance with an embodiment.

[0033] FIG. 16(b) shows a circuit diagram to illustrate the process of elimination of extraneous timing relationships from the merged mode based on the flowchart shown in FIG. 16(a).

[0034] FIG. 17(a) illustrates the details of the third pass for eliminating extraneous timing relationships by disabling data paths feeding inputs to reconvergent timing nodes, in accordance with an embodiment.

[0035] FIG. 17(b) shows a circuit diagram to illustrate the process of elimination of extraneous timing relationships from the merged mode based on the flowchart shown in FIG. 17(a).

[0036] FIG. 18 illustrates an embodiment of the overall process for verifying equivalence between a merged mode and individual modes corresponding to a netlist.

[0037] FIG. 19 illustrates how the timing verification between a merged mode and individual modes is performed for different sets of start points and end points, in accordance with an embodiment.

[0038] FIG. 20 is a block diagram illustrating comparison of timing constraints of two circuit configurations.

[0039] FIG. 21 illustrates two circuit configurations for a circuit with a mismatch in timing constraints that may not be identified by a process based on iterative pairwise comparison of timing constraints.

[0040] FIG. 22 shows an example of a false mismatch shown by iterative pairwise comparison of timing constraints due to a redundant timing constraint in one of the circuit configurations.

[0041] FIG. 23 is a flow diagram illustrating an overall process for detecting mismatched timing constraints in circuit configurations.

[0042] FIG. 24 is a flow chart illustrating the steps for determining the aggregate constraints for a set of end points with respect to a set of start points.

[0043] FIG. 25 is a flow chart illustrating the steps for comparing timing constraints of two circuit configuration based on the first pass in FIG. 23.

[0044] FIG. 26 shows a flow chart illustrating the steps of the second pass in FIG. 23 for matching (start point, end point) pairs.

[0045] FIG. 27 is a flow chart illustrating the steps of the third pass in FIG. 23 for comparing reconvergent points of two circuit configurations.

[0046] FIG. 28 is a high-level block diagram illustrating an example of a computer for use in generation or verification of merged modes with individual modes or for comparison of timing constraints of circuit configurations, in accordance with an embodiment.

[0047] The Figures (FIGS.) and the following description describe certain embodiments by way of illustration only. One skilled in the art will readily recognize from the following description that alternative embodiments of the structures and methods illustrated herein may be employed without departing from the principles described herein. Reference will now be made in detail to several embodiments, examples of which are illustrated in the accompanying figures

DETAILED DESCRIPTION

[0048] Timing constraints can be specified using various commands, for example, set_multicycle_path, set_max_delay, set_min_delay, set_false_path, set_disable_timing, set_clock_groups, set_case_analysis and the like. There is a subset of commands called exceptions that specify exceptions to the default assumptions about timing delays. Examples of exceptions include set_multicycle_path, set_max_delay, set_min_delay and set_false_path commands. The set_multicycle_path command specifies a situation where timing constraints on a circuit path is based on multiple clock cycles rather than a single clock cycle. The set_max_delay command specifies a maximum delay for timing paths in a circuit design. The set_min_delay command specifies a minimum delay for timing paths in a circuit design. The set_false_path command identifies circuit paths in a design that can be ignored (or marked as "false") so that they are not considered during timing analysis. The set_disable_timing command disables timing through specified circuit components, for example, cells, pins, ports etc. The command set_clock_groups specifies clock groups that are mutually exclusive or asynchronous with each other in a design so that paths between these clocks are not considered during timing analysis. The command set_case_analysis specifies that a port or pin is at a constant logic value of 1 or 0.

[0049] If multiple conflicting timing constraints apply to a given path, the conflict can be resolved by a set of predetermined precedence rules. For example, one precedence rule might be

that a set_false_path constraint takes precedence over the set_max_delay path constraint. Then, if there are two constraints satisfied for a path, "set_max_delay –to A 5" and "set_false_path –to A," based on this precedence rule, the set_false_path constraint is dominant over the set_max_delay command. The precedence rules may be based on various criteria. For example, a specific type of constraint may always take precedence over another type of constraint, a constraint that provides more specific details may take precedence over a less specific constraint, or specific precedence rules may be specified by users.

[0050] A timing constraint may be applicable to multiple timing paths. For example, constraint "set_max_delay –from A" applies to all paths that start from point A, whereas constraint "set_false_path –to B" applies to all paths that end at B. However if there is a path from A to B, then the two constraints "set_max_delay –from A" and "set_false_path –to B" are both satisfied but will conflict. Based on the above precedence rule, the set_false_path constraint is dominant for paths that begin with A and end at B so that the set_max_delay constraints will not be used for these paths, but the set_max_delay constraint may still be used for other paths that begin with A but do not end in B.

[0051] Circuit designers perform static timing analysis of hundreds of combinations of modes and corners for implementation and sign-off of circuit designs. An electronic circuit can be represented as a circuit configuration comprising a description of the components of the circuit and the connections between the circuits specified as netlists and a representation of the timing constraints for the circuit. Each mode of a given circuit can be modeled as a set of timing constraints for the given set of netlists. Multiple modes can be merged into smaller set of modes, for example a set of one mode, to reduce the processing costs associated with timing analysis of the circuit design.

[0052] Embodiments generate merged modes and verify correctness of merged modes automatically. Automatic merging and verification of the merged modes gives high confidence in the correctness of the merged modes. As a result, the merged modes can be used for the implementation phase of the design as well as the final sign-off phase. Furthermore, automatic merging or verification of merged modes is less expensive and efficient to perform compared to manual processing.

[0053] FIG. 1 illustrates a merged mode obtained from multiple individual modes for a given circuit configuration. The individual modes 130 and the merged mode 120 correspond to the set

of netlists 110. (A letter after a reference numeral, such as "140a," indicates that the text refers specifically to the element having that particular reference numeral, while a reference numeral in the text without a following letter, such as "140," refers to any or all of the elements in the figures bearing that reference numeral.) Each mode 130, 120 represented in FIG. 1 corresponds to a set of timing constraints corresponding to the netlists 110. The association between the modes and the netlists is represented by the dashed lines 150. The merged mode 120 corresponds to a set of timing constraints that are equivalent to the individual modes 130, as represented by the arrows 140. There can be multiple merged modes 120 corresponding to the set of individual modes 130. The merged mode 120 can be defined manually. Embodiments automatically generate merged mode 120 based on the individual modes 130. Furthermore, embodiments perform automatic verification to determine if the merged mode 120 is equivalent to the individual modes 130.

[0054] FIG. 2 illustrates timing relationships for a circuit associated with multiple modes. The circuit comprises various circuit elements including flip-flops 270a, 270b, a multiplexer 260, two clock inputs clka 250a and clkb 250b, and a multiplexer select input signal 250c. There are two modes 245a and 245b associated with the circuit. The mode 1 comprises a command for creating a clock clka 250a and a command for specifying the value of port 250c to be the constant 0. The mode 2 comprises a command for creating a clock clkb 250b and a command for specifying the value of port 250c to be the constant 1. The table 205 shows a structure for storing timing relations associated with the circuit. The timing relationships 240a and 240b comprise information including a timing start point 210, a timing end point 215, a launch clock 220, a capture clock 225, and information 230, 235 indicating whether the timing constraint is present in mode 1 and/or mode 2 respectively. The information stored in a timing relationship is not limited to the information illustrated in table 205 and may comprise other relevant information. Both timing relations 240a, 240b correspond to timing start point 250e (FF1/CP) and timing end point 250f (FF2/D). However the launch clock and the capture clock for timing relationship 240a are both clkb(r), indicating the rising edge of clkb is used for both launch clock and capture clock for timing 240a. On the other hand, the launch clock and the capture clock for timing relationship 240b are both clka(r), indicating the rising edge of clka is used for both launch clock and capture clock for timing 240b. Furthermore, the timing relationship 240b is present in mode 1 as a single cycle path as shown in column 230 but not present in mode 2 as

shown in column 235. Similarly, the timing relationship 240a is not present in mode 1 as shown in column 230 but present in mode 2 as a single cycle path as shown in column 235.

[0055] Automatic generation and verification of merged constraints is based on comparison of timing relationships between merged mode and individual modes. A timing relationship is specified between a pair of timing nodes and comprises a launching clock, a capturing clock, and timing state, for example, multicycle path or false path between the start point and end point, a timing node specified as having a constant value, etc. Adding a timing constraint to a circuit configuration may affect one or more timing relationships. For example, a timing constraint "set_max_delay –from A" applies to all paths that start from point A and may affect multiple timing relationships associated with point A. Some timing constraints may add new timing relationships, for example, timing constraints that add clocks to the circuit configuration. Some timing constraints can modify existing timing relationships, for example, a timing constraint that specifies a multicycle path can modify existing timing relationships by changing their timing state. Some timing constraints can eliminate timing relationship, for example, specifying false path between two timing nodes indicates that there is no timing relationship between the two timing nodes.

[0056] Automatic generation of merged mode is performed from individual modes for a netlist. An initial merged mode is generated by combining timing constraints from individual modes into the merged mode. One approach to combining timing constraints is the following. The merged mode comprises an intersection set of all timing constraints of individual modes that remove timing relationships. The merged mode also comprises a union set of all timing constraints of individual modes that add or modify timing relationships. The merged mode comprises an intersection set of all timing constraints of individual modes that do not affect timing relationships. The timing relationships in the merged mode are compared with the timing relationships in the individual modes to identify extraneous timing relationships present in the merged mode. Timing constraints are added to the merged mode to eliminate the extraneous timing relationships.

[0057] Automatic verification of correctness of a merged mode with respect to individual modes is performed by comparing timing relationships present in the merged mode with the timing relationships present in the individual modes. Automatic verification of correctness of merged mode is performed by verifying that a timing relationship that exists in any individual mode is

also present in the merged mode. Furthermore, the automatic verification process verifies that every timing relationship that exists in the merged mode is present in at least one of the individual modes. If there are timing relationships in the merged mode that do not occur in an individual mode or there are timing relationships in an individual mode that are not present in the merged mode, the merged mode does not correctly represent the timing constraint behavior of individual modes. Accordingly, such merged mode is not equivalent to the individual modes and a validation of the merged mode based on timing analysis does not guarantee validation of individual modes.

SYSTEM ARCHITECTURE

[0058] FIG. 3 illustrates an embodiment of the system architecture for automatic verification and generation of merged modes based on individual modes. The computer architecture illustrated in FIG. 3 comprises a computer system 300 comprising a verification module 330, a merged mode generator 350, a comparison module 360, a netlist store 310 and a mode store 320. The netlist store 310 stores the netlists of the circuit configuration being analyzed. The mode store 320 stores the information describing the individual modes as well as the merged mode being processed. The verification module 330 performs automatic verification of the merged mode constraints to determine whether the merged mode is equivalent to the individual modes. The merged mode generator 350 performs automatic generation of the merged mode from a set of individual modes. The comparison module 360 performs three passes to match timing relationships between merged modes and the individual modes. The comparison process of the comparison module 360 is invoked by the verification module 330 to perform verification of merged modes and by the merged mode generator 350 to perform generation of merged modes. The system architecture presented in FIG. 3 is used for performing automatic verification and generation of merged modes.

COMPARISON OF MERGED MODES WITH INDIVIDUAL MODES

[0059] The comparison module 360 ensures that every timing relationship present in any individual mode 130 is present in the merged mode 120 and every timing relationship present in the merged mode 120 is present in at least one of the individual modes 130. If the comparison

module 360 identifies any timing relationship that does not satisfy this criteria, the comparison module 360 flags the timing relationship as a mismatch.

[0060] FIG. 4 illustrates one example of how comparison between a merged mode and individual modes is performed by the comparison module 360 for different sets of start points and end points. As illustrated in FIG. 4, in this example, three different passes are performed by the comparison module 360. In the first pass 410, the comparison module 360 compares the merged mode 120 with the individual modes 130 with respect to timing relationships between all start points and a particular end point. The comparison module 360 repeats this process for each end point. If the comparison process determines a mismatch, the comparison module 360 flags the mismatch. The mismatch may be processed manually by a system administrator or by an automatic process. The processing performed when a mismatch is encountered can be used for different purposes, for example, for generation of merged modes or verification of merged modes. If no mismatch is found for an end point and a match cannot be determined decisively due to ambiguities after the first pass 410, the comparison module 360 performs the second pass 420 with respect to the selected end point.

[0061] In the second pass 420, the comparison module 360 compares the merged mode 120 with the individual modes 130 with respect to timing relationships between a particular start point and a particular end point. The comparison module 360 repeats this process for each pair of a start point and an end point. If a mismatch is found, the comparison module 360 flags the mismatch for the pair of start point and end point. If no mismatch is found for a start point and end point pair and a match cannot be determined decisively due to ambiguities after the second pass 420, the comparison module 360 performs the third pass 430 with respect to the selected start point and end point pair.

[0062] In the third pass 430, the comparison module 360 compares the merged mode 120 and the individual modes 130 with respect to timing relationships associated with a reconvergent point between the selected start point and end point pair. The comparison module 360 compares timing relationships associated with a timing node with an edge into the reconvergent point and the reconvergent point. If a mismatch is found, the comparison module 360 flags the mismatch for the timing relationship associated with timing node feeding into the reconvergent point and the reconvergent point. If no mismatch is found for timing nodes feeding into this reconvergent point, the process is repeated for other reconvergent points between the selected start point and

end point pair. If no mismatch is found for any reconvergent point between the start point and end point pair, the timing relationships associated with the start point and end point pair are determined to be equivalent and the comparison module 360 continues processing other start point and end point pairs.

[0063] Embodiments can perform a subset of the steps shown in FIG. 4. For example, an embodiment can perform the first pass 410 without performing the second pass 420 and the third pass 430, if there is no ambiguity found by the first pass 410 (in case of matches as well as mismatches determined by the first pass 410.) Similarly, an embodiment can perform the first pass 410 and the second pass 420, without performing the third pass 430.

[0001] FIG. 5 illustrates the steps for determining whether each relationship present in every individual mode is present in the merged mode, in accordance with an embodiment. The comparison module 360 selects 510 an individual mode for processing. The comparison module 360 selects 520 a timing relationship R in the selected individual mode 130. The comparison module 360 determines 530 if there is a timing relationship in the merged mode 120 that is equivalent to the selected timing relationship R. If an equivalent relationship is not found 540, the timing relationship R is flagged 550 as a mismatch. If an equivalent relationship is found 540, the comparison module 360 checks 560 whether all timing relationships in the selected individual mode 130 are processes. If there are more timing relationships in the individual mode 130 to be processed, the comparison module 360 selects 520 another timing relationship in the individual mode 130 and repeats the above steps. If all the timing relationships of the selected individual mode 130 are processed, the comparison module 360 checks 570 if there are more individual modes 130 to be processed. If the comparison module 360 finds an individual mode 130 to be processed, the comparison module 360 selects 510 the individual mode and performs the above steps for the individual mode 130. If the comparison module 360 determines that all the individual modes 130 are processed, the comparison module 360 stores 580 the results from the above analysis. In an embodiment, the comparison module 360 stores the results as the above steps are processed.

[0064] FIG. 6 illustrates the steps for determining whether each timing relationship present in the merged mode is present in at least one of the individual modes, in accordance with an embodiment. The comparison module 360 selects 620 a timing relationship in the merged mode 120 for processing. The comparison module 360 determines if a timing relationship equivalent to

the selected relationship exists in any one of the individual modes 130. If the comparison module 360 does not find 640 an equivalent timing relation in any individual mode, the comparison module 360 flags 650 the selected timing relationship as a mismatch for further analysis. If the comparison module 360 finds an equivalent timing relationship in at least one of the individual modes 130, the selected relationship is not flagged as a mismatch. The comparison module 360 checks 660 if all timing relationships in the merged mode have been processed. If more timing relationship in the merged mode needs to be processed, the comparison module 360 selects 620 an unprocessed timing relationship and performs the above steps for the selected timing relationship. If all the timing relationships of the merged mode are processed, the comparison module 360 stores the results of the above analysis. In an embodiment, the comparison module 360 stores the results as the above steps are processed.

[0065] Each pass can be described as determining timing relationships associated with a source set of timing nodes and a sink set of timing nodes. For example, the first pass can be described as determining timing relationships associated with the source set comprising all the start points and the sink set comprising a particular end point. Similarly, the second pass can be described as determining timing relationships associated with the source set comprising a particular end point and the sink set comprising a particular start point.

[0066] Next we illustrate the details of each pass with respect to a graph representation of a circuit. FIG. 7 shows timing paths in an example circuit, which will be used to illustrate various terms and a corresponding graph representation of the circuit. A timing node is a port of the circuit or a pin of a circuit component. For example, a clock pin of a register is a timing node. A timing node where a timing analysis can originate from will be referred to as a start point. A timing node where a timing analysis can terminate at will be referred to as an end point. The netlist shown in FIG. 7(a) is represented as a graph comprising nodes and edges as shown in FIG. 7(b). The nodes in the graph of FIG. 7(b) are represented by squares and correspond to the timing nodes in the circuit of FIG. 7(a). The edges in FIG. 7(b) are represented by arrows and correspond to FIG. 7(b)'s connections between timing nodes. For example, timing nodes CPa and CPb are connected by an edge whereas timing nodes CPa and CPe do not have an edge between them. If two timing paths between a start point and end point follow different paths and then converge at a timing node, the timing node is called a reconvergent point 740. For example, in FIG. 7(b), the two timing paths between start point CPc and end point CPg diverge at timing

node CPd and converge again at timing node CPf. The timing node CPf is called a reconvergent point and the timing node CPd is called a divergent point.

[0067] FIGS. 8(a)-(d) illustrate the steps of computation of aggregate timing relationships at an end point of the circuit with respect to a set of start points, in accordance with an embodiment. The various steps shown in FIG. 8 illustrate the computation of the aggregate timing relationships for the circuit starting from an initial set 810 of timing nodes. The timing nodes N1 and N2 are considered the start points for the circuit shown in FIG. 8 and the timing nodes N9 and N10 are the end points of the circuit shown in FIG. 8. The sets of timing relationships for N1 and N2 as shown in FIG. 8(a) are empty since there are no incoming edges to the timing nodes N1 and N2.

[0068] The set of neighboring timing nodes of set 810 includes N3 and N4. The aggregate sets of timing relationships for N3 and N4 are determined as shown in FIG. 8(b). The timing node N4 has a single incoming edge associated with constraint tr5, resulting in the set of timing relationships S4={tr5}. The timing node N3 is a reconvergent point with two incoming edges from timing nodes N1 and N2. The edge from N2 to N3 is not associated with any constraint. Therefore the set of timing relationships for timing node N3 is S3={tr1}. The timing nodes N3 and N4 are added to the set 810 resulting in the set 820 as the current set of timing nodes.

[0069] As illustrated in FIG. 8(c) the neighboring timing nodes of the set 820 are N6 and N7. Next the aggregate sets of timing relationships for N6 and N7 are determined. The aggregate set S6 of timing relationships for the reconvergent point N6, includes the timing relationships in set S3, set S4, and the timing relationships tr2 and tr3 associated with the incoming edges of N6 resulting in S6={tr1,tr2,tr3,tr5}. The aggregate set of timing relationships S7 associated with N7 is determined to be same as S4 since no timing relationships are added by the edge between N4 and N7. The timing nodes N6 and N7 are added to the set 820 to obtain set 830. Similarly, as illustrated in FIG. 8(d), the remaining timing nodes are processed. The timing nodes N9 and N10 are the end points of the circuit.

[0070] FIGS. 9(a)-(d) illustrate the steps of computation of aggregate timing relationships at a start point of the circuit with respect to an end point of the circuit, in accordance with an embodiment. The computation in the second pass is similar to the first pass except that the source set of timing nodes is a particular end point, the sink set is a particular start point and the direction of traversal is in the reverse direction of the edges. As shown in FIG. 9(a), the source

set of timing nodes includes a single end point N9. As shown in FIG. 9(b), the neighboring timing nodes of the set 910 include N6 and N7, traversing in the reverse direction of the edges. The aggregate sets of timing relationships for N6 is S6={}, i.e., empty set since the edge between N6 and N9 is not associated with any constraint. The edge between N7 and N10 is not associated with any constraint but the edge between N7 and N9 is associated with a constraint tr6, resulting in S7={tr6} being added to S7. The timing nodes N6 and N7 are added to the set of timing nodes 910 to obtain set 920. The above process is continued as shown in FIG. 9(c) to determine aggregate sets of timing relationships for timing nodes N3 and N4 since they are neighboring timing nodes for the set 920. The timing node N4 is a divergent point. The timing nodes N3 and N4 are added to the set 920 to obtain set 930. The aggregate sets of timing relationships for the remaining timing nodes can be determined similar to the examples presented above. The final step illustrated in FIG. 9(d) determines the aggregate sets of timing relationships for each start point with respect to the end point N9. The process is repeated for each end point, for example N10.

[0071] FIG. 10 illustrates how timing relationships associated with reconvergent points from two different modes are compared. As shown in FIG. 10, Nk is a reconvergent point. The aggregate sets of timing relationships Sk and Sk' for the reconvergent point Nk for the two modes have the same elements {tr1, tr2, tr3}. However, if the timing nodes with edges incoming to the reconvergent points are compared, a mismatch can be detected. The aggregate sets of timing relationships for Ni in mode mode1 is Si={tr1,tr2} whereas for mode mode2 it is Si'={tr1,tr3}. Similarly the aggregate sets of timing relationships for the timing node Nj in mode mode1 is Sj={tr3} and in mode mode2 is Sj'={tr2}. Hence a mismatch can be flagged corresponding to the timing nodes feeding into the reconvergent points. The analysis illustrated by FIG. 10 can be applied to a divergent point if the timing paths are traversed in the reverse direction of the edges. Accordingly, timing nodes connected by edges that are out going from the divergent point are compared to identify mismatches.

GENERATION OF MERGED MODE

[0072] The automatic generation of modes for a set of netlists based on individual modes is described. FIG. 11 illustrates the overall process for generating merged mode from individual modes for a netlist, in accordance with an embodiment. The merged mode generator 350 creates

1110 an initial merged mode based on the constraints of the individual modes for the given netlist. The merged mode generator 350 eliminates 1120 interactions between two clocks in the merged mode if the two clocks never co-exist in any individual mode. The merged mode generator 350 detects 1130 timing relationships in merged mode that are extraneous with respect to the individual modes. In an embodiment, the merged mode generator 350 detects 1130 the extraneous timing relationships by comparing the merged mode with the individual modes, for example, by executing a three pass algorithm based on the flowchart illustrated in FIG. 4.

[0073] The merged mode generator 350 disables 1140 these extraneous timing relationships by adding exceptions in the merged mode. The merged mode generator 350 adds exceptions to the merged mode by generating new exceptions in the merged mode or by introducing exceptions from individual modes into the merged mode. The exceptions are added to the merged mode in a manner that preferably ensures that pessimism or optimism does not get added to the merged mode for purposes of signal integrity analysis. An example of pessimism is a merged mode in which there is possibility of crosstalk between two timing nodes even though none of the individual modes allow such crosstalk. An example of optimism is a merged mode in which there is no possibility of crosstalk between two timing nodes even though one or more individual modes allow such crosstalk.

[0074] FIG. 12 shows the details of step 1110 illustrating how timing constraints of an initial merged mode are determined from timing constraints of individual modes based on their impact on timing relationships, in accordance with an embodiment. The merged mode generator 350 identifies 1210 a timing constraint in individual modes for determining whether the constraint should be added to the merged mode. The merged mode generator 350 checks 1220 if the timing constraint adds or modifies timing relationships. If the timing constraint adds or modifies timing relationships, the constraint is added to the merged mode if the timing constraint occurs in at least one individual mode. In other words, a union set computed over all individual modes of timing constraints that add/modify timing relationships is added to the merged mode. Examples of timing constraints that add timing relationships include timing constraints that create a clock, timing constraints that set input delay, timing constraints that set maximum delay and the like. For example, a command “create_clock” creates a new clock for the netlist and thereby introduces new timing relationships between timing nodes where the clock reaches. A command “set_input_delay” specifies the minimum and maximum amount of delay from a clock edge to

the arrival of a signal at a specified input port. A "set_output_delay" command specifies the minimum and maximum amount of delay between the output port and the external sequential device that captures data from that output port. Similarly the commands like set_max_delay and set_min_delay set the maximum and minimum delay constraints for paths. These commands are examples of timing constraints that add timing relationships. An example of a timing constraint that modifies timing relationships is set_multicycle_path that specifies the number of clock cycles required to propagate data from a start point to an end point. In sum, a timing constraint that adds/modifies timing relationships is included in the merged mode if the timing constraint is present in any one of the individual modes.

[0075] The merged mode generator 350 further checks 1240 if the timing constraint removes timing relationships. If the timing constraint removes timing relationships, the constraint is added to the merged mode if the timing constraint occurs in all individual modes. In other words, an intersection set computed over all individual modes of timing constraints that remove timing relationships is added to the merged mode. Examples of timing constraints that remove timing relationships include timing constraints that set false path, timing constraints that disable timing, timing constraints that set constant values at pins or ports, set clock groups and the like. For example, the set_false_path timing constraint excludes a path from timing analysis and therefore removes timing relationships along the specified paths. The set_disable_timing command disables a timing edge and thereby prevents timing analysis through the edge and removes timing relationships associated with the edge. The set_case_analysis command sets the data at a pin or port to a constant value, thereby reducing the interactions between the pin/port and other timing nodes. The set_clock_groups constraint can be used to specify exclusivity between clocks, thereby reducing the timing interactions between the clocks. These timing constraints remove timing relationships from a mode. A timing constraint that removes timing relationships is included in the merged mode if the timing constraint is present in all the individual modes.

[0076] In an embodiment, if all individual modes have a constant value of either 0 or 1 specified at a pin/port, such that at least one individual mode specifies a value of one for that pin/port and at least one individual mode specifies a value of zero for that pin/port, a timing constraint is added to the merged mode that specifies that the value on the pin/port is constant, for example, "set_case_analysis constant port_name." The information that the pin/port always has a constant

value indicates that there are no timing relationships associated with this timing node since the signal on the timing node never changes during any mode of operation.

[0077] If the timing constraint does not affect timing relationships, the merged mode generator 350 adds 1260 the timing constraint to the merged mode if the timing constraint is the same across all the individual modes. In an embodiment, the merged mode generator 350 returns error if a timing constraint that does not affect timing relationships does not occur across all individual modes or is not the same across all the individual modes. If the merged mode generator 350 determines 1270 that more timing constraints of the individual modes need to be processed, the merged mode generator 350 identifies 1210 another timing constraint and processes it, or else, the merged mode generator 350 stores 1280 the results. The results can be stored after or during other steps of the process illustrated in FIG. 12.

[0078] Next, FIG. 13(a) illustrates an example process for eliminating 1120 timing relationships between clocks in the merged mode. The merged mode generator 350 eliminates 1120 interactions between clocks in the merged mode if the clocks do not co-exist in any individual mode with respect to a set of timing nodes. The merged mode generator 350 performs propagation 1310 of clock inputs or case inputs that specify a constant value at a pin/port. The merged mode generator 350 identifies timing nodes where merged mode has more clocks compared to fan-in points. The merged mode generator 350 detects combinations of clocks that reach the timing node but do not coexist in any individual mode. The merged mode generator 350 adds timing constraints to the merged mode to declare these clocks as mutually exclusive.

[0079] For example, if multiple individual modes are combined to a merged mode and clock clka is disabled in all individual modes in which clkb is enabled and clkb is disabled in all individual modes in which clka is enabled, the merged mode generator 350 determines that clka and clkb cannot coexist in any individual mode. Accordingly, the merged mode generator 350 eliminates the interactions between clocks clka and clkb in the merged mode by introducing appropriate constraints, for example, a constraint that specifies that clka and clkb are exclusive (logically or physically).

[0080] FIG. 13(b) shows a circuit diagram to illustrate the process of elimination of timing relationships based on the flowchart shown in FIG. 13(a), in accordance with an embodiment. The timing nodes 1350a and 1350b correspond to pins receiving clocks clka and clkb as inputs respectively. The input pin 1350c provides an input sel which allows selection of one of the

inputs of the multiplexer 1360. The selected clock input reaches the output 1350d of the multiplexer 1360 and is provided as the input clock for the flip-flops 1370a and 1370b. There can be two individual modes, mode1 and mode2. Assume that in mode1 the input at timing node 1350c (sel) is set to a constant value 1, thereby selecting the clock clka at the input pin 1350a for the multiplexer 1360. This can be specified using a timing constraint "set_case_analysis 1 [get_ports sel]" in mode1. In mode2 the value for timing node 1350c (sel) value is set to a constant 0, thereby selecting the clock clkb at the input pin 1350b for the multiplexer 1360. This can be specified using a timing constraint "set_case_analysis 0 [get_ports sel]."

[0081] The merged mode contains both the create_generated_clock timing constraints described above, since these constraints add timing relationship and are added to the merged mode in step 1220, 1230 of the flowchart in FIG. 12. However, in any individual mode, only one clock input reaches the timing node 1350d and the clocks do not co-exist in any individual modes. Therefore, a timing constraint is added 1340 to disable any interactions between the two clocks by declaring the two clocks as mutually exclusive, for example, "set_clock_groups -physically_exclusive –group clka –group clkb." The set_clock_groups command defines groups of clocks that are mutually exclusive with respect to each other. Specifying the two clocks clka and clkb to be physically exclusive indicates that there is no need to perform any crosstalk analysis between the clock nets. Therefore, any timing analysis performed using the merged mode does not check paths that start from a clock in one group (say clka) and end at a clock in another group (say clkb).

[0082] Next, the steps of detection 1130 of extraneous timing relationships in merged mode for disabling 1140 the timing relationships is described in detail. FIG. 14 shows the flowchart for illustrating the steps for detecting and eliminating extraneous timing relationships in a merged mode, in accordance with an embodiment. The flowchart shown in FIG. 14 performs three passes similar to the flowchart illustrated in FIG. 4. The first pass eliminates 1410 extraneous timing relationships from the merged mode by disabling clock paths that reach a timing node in the merged mode but do not reach the same timing node in any individual mode. The second pass eliminates 1420 extraneous timing relationships by disabling data paths in merged modes if a timing relationship is present between two timing nodes but there is no corresponding timing relationship between the timing nodes in any individual modes. The third pass eliminates 1430

extraneous relationships by disabling data paths feeding inputs to reconvergent timing nodes. The details of the three passes are further described herein.

[0083] FIG. 15(a) illustrates the details of the first pass 1410 for eliminating extraneous timing relationships from the merged mode by disabling clock paths, in accordance with an embodiment. The merged mode generator 350 propagates the constraints from all start points to an end point for each individual mode and the merged mode. This step is performed in the manner illustrated by example in FIG. 8. The merged mode generator 350 analyzes the aggregated timing relationships for the end point to identify capture clocks that reach the end point in the merged mode and individual modes. In particular, the merged mode generator 350 detects 1520 capture clocks that reach the end point in the merged mode but are not capture clocks in any individual mode for that end point. The merged mode generator 350 adds timing constraints to the merged mode to disable the clock path from reaching the timing node. For example, a clock clk1 can be disabled by adding a constraint "set_clock_sense –stop_propagation –clock clk1 pin_name." In an embodiment, an intermediate timing node connecting the clock input to the end point is selected for disabling the clock path. The steps illustrated in FIG. 15 are repeated for all end points.

[0084] FIG. 15(b) illustrates the process of FIG. 15(a) by an example. The inputs 1550a and 1550b are provided to an OR gate 1580. There are two individual modes modeA and modeB associated with the netlist shown in FIG. 15. In modeA, the input 1550a is set to constant value 1 and 1550b is set to a constant value 0. In modeB, the input 1550a is set to a constant value 0 and 1550b is set to a constant value 1. As a result, the output 1550c of the gate 1580 is determined to be 1 in modeA as well as modeB. The value at timing node 1550c corresponds to the select input value for the multiplexer 1570. Accordingly, only the clock input clkA on pin 1550e reaches the output 1550f of the multiplexer 1570 for the individual modes. The clkB input on pin 1550d never reaches the output 1550f of the multiplexer 1570 for the modes modeA and modeB. The output 1550f of the multiplexer 1570 acts as the clocks for the flip-flops 1560a and 1560b. Accordingly, in none of the individual modes, modeA and modeB, the clock input clkA at pin 1550d reaches the end points corresponding to the flip-flops 1560a and 1560b. Therefore, the clock clkA is disabled in the merged mode by adding 1530 a timing constraint, for example, "set_clock_sense –stop_propagation –clock clkA [get_pin mux1/Z]" wherein mux1

corresponds to the multiplexer 1570 and the pin returned by "get_pin mux1/Z" corresponds to the timing node 1550f.

[0085] FIG. 16(a) illustrates the details of the second pass 1420 for eliminating extraneous timing relationships by disabling data paths in merged modes, in accordance with an embodiment. The merged mode generator 350 propagates 1610 the timing relationships from an end point to a start point as illustrated in FIG. 9. The merged mode generator 350 detects 1620 extraneous timing relationships between a pair of start point and end point and adds 1630 timing constraints to the merged mode to disable extraneous paths between the start point and end point. The steps illustrated in FIG. 16(a) are repeated for all start points that can reach an end point, and for each end point that couldn't be resolved in pass1 alone.

[0086] FIG. 16(b) illustrates the process of FIG. 16(a) by an example. Assume there are two individual modes modeX and modeY for the netlists shown in FIG. 16(b). The input SE provided via pin 1650a reaches the output 1650b of the buffer BUF and is provided as an input to the multiplexer 1670. The mode modeX specifies the input SE to the pin 1650a to be constant 0, for example, by specifying "set_case_analysis 0 [get_port SE]." The mode modeY specifies the input SE to the pin 1650a to be constant 1, for example, by specifying "set_case_analysis 1 [get_port SE]." These two timing constraints are of the type that remove timing relationships since specifying a constant value for a pin ensures that there are no timing relationships associated with the pin. Therefore, these timing constraints can be added to the initial merged mode only if they are the same across all individual modes. However, since these timing constraints are not the same for modeX and modeY, these are not added to the merged mode. Accordingly extraneous timing relationships of data paths that start from the input SE and reach timing nodes associated with the flip-flop 1660 are disabled by adding appropriate timing constraints to the merged mode. For example, a timing constraint "set_false_path –from SE" can be added to the merged mode to disable paths from input 1650a (port SE) to the end point associated with the timing node 1650f (input of flip-flop 1660).

[0087] FIG. 17 illustrates the details of the third pass 1430 for eliminating extraneous timing relationships in merged mode by disabling data paths feeding inputs to reconvergent timing nodes, in accordance with an embodiment. The merged mode generator 350 detects 1710 reconvergent points between a start point and an end point. This check is performed if there are multiple timing relationships between a start point and end point pair caused by reconvergent

points between the start point and end point. For example, FIG. 17(b) shows an example netlist with a reconvergent point 1750c. The gate 1760 has two inputs, 1750a and 1750b. The input 1750a is provided as input A to the buffer BUF and the output Z of the buffer BUF is fed in input 1750b to the gate 1760. Assume there are two individual modes, modeP and modeQ for the circuit in FIG. 17(b). The individual mode modeP specifies a timing constraint "set_false_path –through [get_pin BUF/A]." The individual mode modeQ specifies a timing constraint "set_false_path –through [get_pin BUF/Z]." Accordingly, the input to the gate 1760 via the timing node 1750b is set to false path in both individual modes. Two timing relationships are detected at the reconvergent point 1750c, one associated with input X of the gate 1760 and another associated with input Y of the gate 1760.

[0088] Since there are multiple timing relationships reaching the reconvergent point 1750c, the appropriate timing relationships are not eliminated in pass 1 (illustrated in FIG. 15) and pass 2 (illustrated in FIG. 16). However, each input of the reconvergent point is compared between the merged mode and the individual modes to determine if a path can be disabled. In FIG. 17(b), the merged mode generator 350 determines that in both the individual modes modeP and modeQ, the input Y received at timing node 1750b is disabled. Accordingly a timing constraint is added 1730 to the merged mode to disable the path arriving at the input Y of gate 1760, for example, "set_false_path –through [get_pin U1/Y]."

VERIFICATION OF MERGED MODES

[0089] FIG. 18 illustrates an embodiment of an overall process for verifying equivalence between a merged mode and individual modes corresponding to a netlist. The verification module 330 performs the steps illustrated in the flowchart of FIG. 18 to verify the equivalence between the merged mode and the individual modes with respect to a set of start points and a set of end points of the circuit configuration.

[0090] The verification module 330 determines 1810 the timing relationships encountered between the set of start points and the set of end points in each individual modes 130 as well as the merged mode 120. In an embodiment, the determination of the timing relationships comprises aggregating timing relationships by performing a graph traversal from the set of start points to the set of end points as described herein. The verification module 330 verifies 1820 whether a timing relationship present in an individual mode 130 is present in the merged mode

120. If a timing relationship is present in an individual mode 130 but no equivalent timing relationship is found in the merged mode 120, the timing relationship is flagged as a mismatch. A mismatch can be flagged by logging a message or by presenting a message to a user. A timing relationship from one mode is equivalent to a timing relationship from another mode if the two timing relationships are associated with the same pair of timing nodes and have equivalent launching clock and capture clock as well as equivalent state as defined by the associated timing constraints. A mismatch found between the merged mode and an individual mode can be further analyzed automatically or by a system administrator.

[0091] The verification module 330 verifies 1830 for each timing relationship in the merged mode 120, whether there is an equivalent timing relationship in at least one of the individual modes 130. If a timing relationship is present in the merged mode 120 but there is no equivalent timing relationship in any of the individual modes 130, the verification module 330 flags the timing relationship as a mismatch. This timing relationship can be further analyzed automatically or by a system administrator.

[0092] Accordingly, the verification module 330 ensures that every timing relationship present in any individual mode 130 is present in the merged mode 120 and every timing relationship present in the merged mode 120 is present in at least one of the individual modes 130. If the verification module 330 identifies any timing relationship that does not satisfy this criteria, the verification module 330 flags the timing relationship as a mismatch.

[0093] The verification process can be performed using the three pass process illustrated in FIG. 19 which is similar to the three pass process illustrated in FIG. 4. The three passes can be performed by the verification module 330 by invoking the comparison module 360. In the first pass 1910, the verification module 330 verifies the equivalence between the merged mode 120 with the individual modes 130 with respect to timing relationships between all start points and a particular end point based on the process illustrated in FIG. 18. The verification module 330 repeats this process for each end point. If the verification process determines a mismatch, the verification module 330 flags the mismatch. The mismatch may be fixed manually by a system administrator or by an automatic process. If no mismatch is found for an end point and a match cannot be determined decisively due to ambiguities after the first pass 1910, the verification module 330 performs the second pass 1920 with respect to the selected end point.

[0094] In the second pass 1920, the verification module 330 verifies the equivalence between the merged mode 120 with the individual modes 130 with respect to timing relationships between a particular start point and a particular end point based on the process illustrated in FIG. 18. The verification module 330 repeats this process for each pair of a start point and an end point. If a mismatch is found, the verification module 330 flags the mismatch for the pair of start point and end point. The step 1920 is repeated for each start point with respect to the selected end point. If no mismatch is found for a start point and end point pair and a match cannot be determined decisively due to ambiguities after the second pass 1920, the verification module 330 performs the third pass 1930 with respect to the selected start point and end point pair.

[0095] In the third pass 1930, the verification module 330 verifies the equivalence between the merged mode 120 and the individual modes 130 with respect to timing relationships associated with a reconvergent point between the selected start point and end point pair. The verification module 330 verifies equivalence for timing relationships associated with a timing node with an edge into the reconvergent point and the reconvergent point. If a mismatch is found, the verification module 330 flags the mismatch for the timing relationship associated with timing node feeding into the reconvergent point and the reconvergent point. If no mismatch is found for timing nodes feeding into this reconvergent point, the process is repeated for other reconvergent points between the selected start point and end point pair. If no mismatch is found for any reconvergent point between the start point and end point pair, the timing relationships associated with the start point and end point pair are determined to be equivalent and the verification module 330 continues processing other start point and end point pairs.

[0096] Embodiments can perform a subset of the steps shown in FIG. 19. For example, an embodiment can perform the first pass 1910 without performing the second pass 1920 and the third pass 1930, if there is no ambiguity found by the first pass 1910 (in case of matches as well as mismatches determined by the first pass 1910.) Similarly, an embodiment can perform the first pass 1910 and the second pass 1920, without performing the third pass 1930.

[0097] If the steps 1910, 1920, 1930 find no mismatch, the verification module 330 determines the merged mode 120 to be equivalent to the individual modes 130. In an embodiment, any mismatches found are flagged. The mismatches can be fixed via an automatic process or by a system administrator by modifying the merged mode 120. The above verification process can be

repeated for the modified merged mode 120 to determine if the modified merged mode 120 is equivalent to the individual modes 130.

[0098] A process similar to that illustrated in FIG. 5 can be used for verifying whether each relationship present in every individual mode is present in the merged mode. Similarly, a process similar to that illustrated in FIG. 6 can be used for verifying whether each timing relationship present in the merged mode is present in at least one of the individual modes. Accordingly the verification process can be performed to determine if a merged mode is equivalent to several individual modes.

[0099] In sum, embodiments allow efficient generation of merged modes from individual modes for a set of netlists and verification of equivalence between merged mode and individual modes for the set of netlists. Timing constraints are added to the merged mode in a manner that does not add pessimism to the merged mode. Errors are reported during merging if necessary identifying timing nodes causing errors. The merged mode generated is determined with high accuracy and confidence and can be used for implementation stages of design as well as sign-off stages.

COMPARING TIMING CONSTRAINTS OF ELECTRONIC CIRCUITS

[00100] A process similar to that illustrated in FIG. 4 can be used for comparison of timing constraints of two different circuits. FIG. 20 is a high-level block diagram illustrating comparison of timing constraints of two circuit configurations. The two circuit configurations 2010a and 2010b are inputs to a circuit configuration comparator 2040. The circuit configuration 2010a includes a netlist specification 2020a and a timing constraints specification 2030a. The circuit configuration 2010b includes a netlist specification 2020b and a timing constraints specification 2030b. The circuit configuration comparator 2040 generates circuit configuration differences 2050 based on differences in the timing constraints 2030. For example, if there are no effective timing constraint differences between circuit configurations 2010a and 2010b, the circuit configuration differences 2050 indicate the two circuit configurations are equivalent. On the other hand, if there are timing constraint differences between circuit configurations 2010a and 2010b, the circuit configuration differences 2050 might specify the details of the netlists or inputs/outputs that correspond to timing constraints differences as well as the corresponding timing constraints that do not match.

[00101] If conflicting timing constraints are applied to a given path, a particular timing constraint is selected based on a set of predetermined precedence rules. For example, one precedence rule might be that a set_false_path constraint takes precedence over the set_max_delay path constraint. Then, if there are two constraints satisfied for a path, "set_max_delay –to A 5" and "set_false_path –to A," based on this precedence rule, the set_false_path constraint is dominant over the set_max_delay command. The precedence rules may be based on various criteria, for example, a specific type of constraint may always take precedence over another type of constraint, a constraint that provides more specific details may take precedence over a less specific constraint, or specific precedence rules may be specified by users.

[00102] A timing constraint may be applicable to multiple timing paths. For example, constraint "set_max_delay –from A" applies to all paths that start from point A, whereas constraint "set_false_path –to B" applies to all paths that end at B. However if there is a path from A to B, then the two constraints "set_max_delay –from A" and "set_false_path –to B" are both satisfied but will conflict. Based on the above precedence rule, the set_false_path constraint is dominant for paths that begin with A and end at B, but the set_max_delay constraint may still be applicable to paths that begin with A but do not end in B.

INTERACTION OF TIMING CONSTRAINTS

[00103] Conventional techniques for comparing timing constraints of two circuit configurations (for example, iterative pairwise comparison of individual constraints) do not take into consideration interactions between timing constraints. The circuit configuration comparator 2040 can attempt to pair each timing constraint in 2030a with a corresponding timing constraint in 2030b and determine whether the paired constraints are equivalent. The iterative pairwise comparison may be performed for all constraints of a particular type together, for example, the comparison of all set_case_analysis and then clocks, input/output delays, exceptions, followed by comparisons of clock latencies, followed by comparisons of clock uncertainty, and so on.

[00104] However, pairwise comparison of constraints does not take into consideration various interactions between timing constraints. As a result, pairwise comparison of constraints may not detect all mismatches or may report false mismatches. FIGS. 21 and 22 illustrate two examples.

[00105] FIG. 21 illustrates two circuit configurations 2110a and 2110b for a circuit being compared to detect timing constraints mismatches. The netlist specifications 2120a and 2120b for the two circuit configuration 2110 are identical. However, the timing constraints 2130a include a constraint 2115 "set_multicycle_path 3 -through [get_pins inv/Z]," whereas the timing constraints in 2130b include a constraint 2135 "set_multicycle_path 3 -from [get_pins rA/CP]." The constraint 2125 in timing constraints 2130a is identical to the timing constraint 2145 in timing constraints 2130b. An iterative pairwise comparison may conclude that the constraint 2115 is equivalent to constraint 2135 and there is no mismatch between the timing constraints 2130a and 2130b.

[00106] However, the timing constraints associated with the path from register rA through pin inv/Z through pin and/Z to register rY may be resolved differently for circuit configurations 2110a and 2110b based on precedence rules. For the path from register rA through pin inv/Z through pin and/Z to register rY in circuit configuration 2110a both constraints 2115 and 2125 are applicable and the precedence rules may determine that the resulting constraint is 2125 with the most constraining multi-cycle path of 2 cycles. On the other hand, in circuit configuration 2110b, again both constraints 2135 and 2145 are applicable to the path from register rA through pin inv/Z through pin and/Z to register rY and the precedence rules may determine that the resulting constraint is 2135 with a multi-cycle path of 3 cycles because 2135 is a multicycle path that originates from a clock pin. As a result, due to the interaction of different constraints, a comparison of timing constraints for the two circuit configurations should flag a mismatch for pin inv/Z. However, an iterative pairwise comparison of the two circuit configurations 2110 does not consider the interactions between the constraints for a circuit configuration and may determine that there is no mismatch between the two circuits.

[00107] FIG. 22 shows an example of circuit configurations that can falsely be flagged as a timing mismatch by a circuit configuration comparator 2040 that uses iterative pairwise configuration. FIG. 22 shows two circuit configurations 2210a and 2210b with identical netlists 2220a and 2220b respectively. The timing constraints specification 2230a includes one command whereas timing constraints specification 2230b includes two commands. The constraint 2215 in timing constraints specification 2230a is the same as constraint 2235 in timing constraints specification 2230b. An iterative pairwise comparison of the two timing constraints specifications 2230a and 2230b may attempt to match the above pairs of constraints and

determine that the constraint 2245 does not have a matching constraint in 2230a. As a result, an iterative pairwise comparison of the two circuit configurations may report a mismatch in the timing constraints of the two circuit configurations 2210a and 2210b.

[00108] However, the constraints 2235 and 2245 apply to the same timing paths and precedence rules can be used to determine if a constraint can be eliminated. For example, if precedence rules decide that a set_false_path command overrides a set_multicycle_path command, the timing constraints 2245 can be eliminated. Accordingly, the two sets of timing constraints 2230a and 2230b are identical and there is no timing mismatch between the two circuit configurations 2210a and 2210b. Therefore, a false mismatch can be reported by iterative pairwise comparison of timing constraints even if there is no real mismatch as determined by considering interactions between timing constraints.

[00109] False mismatches reported by a tool may require further analysis of the circuits causing unnecessary delays in completing the design process. On the other hand, failure to report timing mismatches may result in faulty signoff and incorrect operation of the circuit.

TIMING BEHAVIOR ANALYSIS

[00110] Timing behavior analysis is performed by aggregating timing relationships associated with timing nodes along a timing path while traversing from a start point to an end point. Alternatively, the timing relationships can be aggregated while traversing from an end point to a start point, in the reverse direction of the edges. An edge in the timing path can be associated with zero or more timing relationships.

[00111] An embodiment of the invention compares corresponding (start point, end point) pairs between two circuit configurations by comparing the aggregate sets of timing relationships obtained at each end point while traversing from the corresponding start point. The aggregate sets of timing relationships for start points obtained by traversing in the reverse direction of the edges starting from the end points can also be compared to similarly identify timing relationship mismatch between circuit configurations.

[00112] The set of timing nodes from where the traversal originates for aggregating timing relationships is referred to as a source set. The set of timing nodes where the traversal terminates for aggregating timing relationships is called a sink set. The direction of traversal is from the source set to the sink set. If the source set comprises start points and the sink set comprises end

points, the direction of traversal is in the direction of the edges. If the source set comprises end points and the sink set comprises start points, the direction of traversal is against the direction of the edges.

[00113] In an embodiment, the timing relationships are aggregated by starting the traversal from a source set of timing nodes and ending the traversal at a sink set of timing nodes. The aggregate timing relationships associated with the sink set can be compared for two circuit configurations. Precedence rules can be applied to eliminate redundant constraints associated with timing nodes. Timing relationships associated with different timing nodes in the same sink set are not combined together but are treated separately. Aggregate timing relationships for two sink sets of timing nodes from two circuit configurations are compared by comparing the aggregate timing relationships for individual timing nodes in the sets. If the aggregate timing relationships obtained for a set of end points starting from a set of start points shows a mismatch, the end points can be flagged as a mismatch.

[00114] A timing node of a source set can be called a source timing node and a timing node of a sink set can be called a sink timing node. The timing nodes of a source set can be all start points or all end points but not a mixture of both. Similarly timing nodes in a sink set can be all start points or all end points but not both. If the source set comprises start points, the sink set comprises end points. Similarly if the source set comprises end points, the sink set comprises start points.

[00115] The aggregation of the timing relationships for all the timing paths from a source timing node to a sink timing node allows interactions between constraints to be considered. For example, precedence rules can be applied to an aggregate set of timing relationships for a timing node to determine an effective set of constraint that is applicable to the timing node. A timing relationship may override another timing relationship associated with one or more timing nodes. For example, two sets of timing relationships Sg = {t1, t2, t4, t5, t6} and Sg'= {t1, t2, t4, t5} may be determined to have a mismatch. However, if the timing relationship t5 overrides the timing relationship t6 based on precedence rules, the two sets Sg and Sg' are determined to be equivalent. On the other hand, based on precedence rules, if it is determined that the timing relationship t6 overrides the timing relationship t5, the two sets are determined as not equivalent and a mismatch may be flagged.

[00116] Even if the aggregate timing relationships match for corresponding end points, there can be a mismatch for intermediate timing nodes encountered while traversing from a start point to the end point (or in the reverse direction). In one embodiment, if the timing relationship match for corresponding end points between two circuit configurations, further analysis is performed to identify if the aggregate timing relationship sets for the other timing nodes in the timing paths match or not. In one embodiment, if the timing relationship between two circuit configurations are determined to be matching for the corresponding end points a subset of nodes along the timing paths is further analyzed. For example, only the reconvergent/divergent nodes along the timing paths are analyzed.

COMPARING CIRCUIT CONFIGURATIONS

[00117] FIG. 23 shows an embodiment of the overall process for comparing timing relationships of two circuit configurations CC1 and CC2. The process involves three passes through the circuit, each pass identifying timing mismatches between the two circuit configurations. The first pass 2310 computes aggregate timing relationships for each end point starting from the complete set of start points. The aggregate timing relationships for the corresponding end points between the two circuit configurations are compared to identify mismatches. If a mismatch is detected for an end point, the end point is flagged as a mismatch. The set of end points that are not flagged as mismatches are further processed in the second pass 2320.

[00118] The second pass 2320 compares aggregate timing relationships obtained for each start point by traversing from each end point in the reverse direction of the edges. The corresponding (start point, end point) pairs for the two circuit configurations are compared to identify mismatches. If a (start point, end point) pair for the two circuit configurations shows a mismatch, the mismatch is flagged. The (start point, end point) pairs for the two circuit configurations that do not show any mismatch in the second pass are further processed in the third pass. The third pass 2330 compares aggregate timing relationships of timing nodes feeding into reconvergent points between a (start point, end point) pair. This process is continued until either mismatches are detected or the (start point, end point) pair is determined not to have any mismatches.

[00119] In one embodiment, the first pass 2310 and the second pass 2320 are implemented by a breadth first graph traversal of the circuit associated with the circuit configuration. In one embodiment, the breadth first algorithm starts with a set of timing nodes and treats it as the current set of timing nodes. At each step, all timing nodes that neighbor the current set are processed. A timing node neighbors a set of timing nodes if all incoming edges of the timing node originate at nodes within the set (and the timing node does not belong to the set). For example, if a timing node has three incoming edges and all three incoming edges are coming from nodes within the set, the timing node neighbors the set. However, if two of the incoming edges are coming from nodes within the set and one edge comes from a node outside the set, the timing node does not neighbor the set. The above description of a neighboring node is based on a graph being processed in the direction of the edges. If a graph is processed in the reverse direction of edges, a neighboring node of a set of timing nodes is defined as the timing node such that all outgoing edges from the timing node connect to a node within the set.

[00120] Processing a neighboring timing node comprises computing the aggregate timing relationships for the neighboring timing node based on the timing nodes in the current set. After the set of neighboring timing nodes is processed, the neighboring timing nodes are added to the current set and the above process repeated. The process is continued until all timing nodes are processed or until a desired timing node, say an end point is reached.

[00121] The details of the steps used in the various passes shown in FIG. 23 are further described as flowcharts. FIG. 24 shows a flow chart illustrating the steps for determining the aggregate timing relationships for a set of end points with respect to a set of start points. The set S of timing nodes is initialized 2410 as the set of all start points. The set N of all neighboring timing nodes of the timing nodes in set S is determined 2420. The aggregate timing relationships for the timing nodes in the set N are computed 2430. The timing nodes of set N are added 2435 to the set S. If it is determined 2440 that all timing nodes outside set S are processed, the processing stops by storing the data of the set S. Otherwise, the above steps 2420, 2430, 2435, and 2440 are repeated. The computation of the second pass 2320 is also illustrated by the flowchart shown in FIG. 24, where the set S is initialized to an individual end point and the edges are traversed in the reverse direction to determine the neighboring timing nodes with respect to set S.

[00122] FIG. 25 shows how the process shown in FIG. 24 is used for comparing timing behavior of two circuit configurations. The two circuit configurations CC1 and CC2 that are being compared are read S10. In some embodiments, the circuit configurations are made available as input files stored on the disk. The aggregate timing relationships for each end point with respect to the complete set of start points is determined 2520 as described by the steps in FIG. 24. The results obtained for the two circuit configurations are compared to determine if there are mismatches. An end point is selected 2530 from CC1 and the corresponding end point is selected 2530 from CC2. The aggregate timing relationships of the corresponding end points from CC1 and CC2 are compared 2540. If a mismatch is found 2550, the mismatch is flagged 2560. Two sets of timing relationships can be determined to have a mismatch if the timing relationships in one set are all different from the timing relationships of the other set. If the two sets are singleton sets and the corresponding elements match, the two sets are determined to match. If no mismatch is found between the two end points, the end-point is further processed 2570 using the second pass. If a subset of the set of timing relationships matches but the remaining elements show a mismatch, the second pass can be used to further analyze the subset that matches. If all end points are processed 2580, the results may be stored and the process stops. Otherwise, the next pair of end points is selected 2530 from CC1 and CC2 and the process above repeated.

[00123] FIG. 26 shows a flow chart illustrating the steps of the second pass 2320 for matching (start point, end point) pairs according to an embodiment. An end point is selected 2610 from the circuit configuration CC1 and the corresponding end point selected 2610 from circuit configuration CC2. As described above, the second pass is executed if the aggregate timing relationships for the corresponding end points from the circuit configurations match. The aggregate timing relationships are determined for each start point with respect to the end point for each circuit configuration by following the process described in FIG. 24 from the end point and traversing the edges in the reverse direction. A (start point, end point) pair is selected 2630 from each circuit configuration such that there is at least one timing path between the start point and the end point. The aggregate timing relationships of the corresponding start points from the two circuit configurations are compared 2640. If a mismatch is found 2650, the mismatch is flagged 2660. Otherwise, the (start point, end point) pairs from the two circuit configurations are processed 2670 using the third pass 2330. If all (start point, end point) pairs are determined 2680

to be processed, the results are stored 2690 and the process stops. Else the next (start point, end point) pair is selected 2630 and the above steps repeated.

[00124] FIG. 27 shows a flow chart illustrating the steps for comparing reconvergent/divergent points of two circuit configurations according to an embodiment. The following description applies to reconvergent points but can be extended to divergent points. A (start point, end point) pair (SP1, EP1) is selected 2710 from circuit configuration CC1 and a corresponding (start point, end point) pair (SP2, EP2) is selected 2720 from circuit configuration CC2 such that the aggregate timing relationships of SP1 and SP2 match as determined by the second pass 2320. A reconvergent point MP1 that is encountered while traversing from SP1 to EP1 is selected 2730 from the pair (SP1, EP1) and a corresponding reconvergent point MP2 that is encountered while traversing from SP2 to EP2 is selected 2730 from the pair (SP2, EP2). The timing nodes that feed edges into MP1 and MP2 are compared 2740 (i.e., a timing node feeds an edge into a reconvergent point if an edge connects the timing node to the reconvergent point). If MP1 and MP2 are divergent points, the timing nodes connected to MP1, MP2 via outgoing edges are matched 2740.

[00125] If a mismatch is found 2750, the mismatch is flagged 2760. If the reconvergent points are found to match, the remaining circuit paths associated with the (start point, end point) pairs are checked to see if all reconvergent points are processed 2780. If all reconvergent points are determined 2780 to be processed, the processing of the (start point, end point) pair is completed, and the results can be stored 2790. Otherwise, the next unprocessed reconvergent point is selected 2730 and processed as described above.

[00126] The above processes perform comparison of timing nodes based on aggregate sets of timing relationships associated with timing nodes determined by traversing from one or more source timing nodes to a target timing node. The computation of aggregate sets of timing relationships associated with timing nodes allows processing of the sets of timing relationships based on precedence rules or other criteria in order to obtain a normalized set of timing relationships. Obtaining a normalized set of timing relationships simplifies the process of matching corresponding timing nodes from two circuit configurations and also reduces the possibility of false positives. As a result the disclosed timing relationship comparison provides improved results over existing approaches, for example, iterative pairwise comparison of constraints for the two circuit configuration. For example, false timing constraint mismatches

that may be reported by iterative pairwise comparison of constraints as described in FIG. 22 are not reported by the disclosed techniques. Besides, timing constraint mismatches that exist in the circuits due to interaction between constraints but are not reported by iterative pairwise comparison of constraints as described in FIG. 21 are reported by the disclosed techniques.

COMPUTER ARCHITECTURE

[00127] FIG. 28 is a high-level block diagram illustrating an example computer 2800 that can be used for processing the steps of the processes described herein. The computer 2800 includes at least one processor 2802 coupled to a chipset 2804. The chipset 2804 includes a memory controller hub 2820 and an input/output (I/O) controller hub 2822. A memory 2806 and a graphics adapter 2812 are coupled to the memory controller hub 2820, and a display 2818 is coupled to the graphics adapter 2812. A storage device 2808, keyboard 2810, pointing device 2814, and network adapter 2816 are coupled to the I/O controller hub 2822. Other embodiments of the computer 2800 have different architectures.

[00128] The storage device 2808 is a non-transitory computer-readable storage medium such as a hard drive, compact disk read-only memory (CD-ROM), DVD, or a solid-state memory device. The memory 2806 holds instructions and data used by the processor 2802. The pointing device 2814 is a mouse, track ball, or other type of pointing device, and is used in combination with the keyboard 2810 to input data into the computer system 2800. The graphics adapter 2812 displays images and other information on the display 2818. The network adapter 2816 couples the computer system 2800 to one or more computer networks.

[00129] The computer 2800 is adapted to execute computer program modules for providing functionality described herein. As used herein, the term "module" refers to computer program logic used to provide the specified functionality. Thus, a module can be implemented in hardware, firmware, and/or software. In one embodiment, program modules are stored on the storage device 2808, loaded into the memory 2806, and executed by the processor 2802. The types of computers 2800 used can vary depending upon the embodiment and requirements. For example, a computer may lack displays, keyboards, and/or other devices shown in FIG. 28.

[00130] Some portions of above description describe the embodiments in terms of algorithmic processes or operations. These algorithmic descriptions and representations are commonly used by those skilled in the data processing arts to convey the substance of their work effectively to

others skilled in the art. These operations, while described functionally, computationally, or logically, are understood to be implemented by computer programs comprising instructions for execution by a processor or equivalent electrical circuits, microcode, or the like. Furthermore, it has also proven convenient at times, to refer to these arrangements of functional operations as modules, without loss of generality. The described operations and their associated modules may be embodied in software, firmware, hardware, or any combinations thereof.

[00131] As used herein any reference to "one embodiment" or "an embodiment" means that a particular element, feature, structure, or characteristic described in connection with the embodiment is included in at least one embodiment. The appearances of the phrase "in one embodiment" in various places in the specification are not necessarily all referring to the same embodiment.

[00132] Some embodiments may be described using the expression "coupled" and "connected" along with their derivatives. It should be understood that these terms are not intended as synonyms for each other. For example, some embodiments may be described using the term "connected" to indicate that two or more elements are in direct physical or electrical contact with each other. In another example, some embodiments may be described using the term "coupled" to indicate that two or more elements are in direct physical or electrical contact. The term "coupled," however, may also mean that two or more elements are not in direct contact with each other, but yet still co-operate or interact with each other. The embodiments are not limited in this context.

[00133] As used herein, the terms "comprises," "comprising," "includes," "including," "has," "having" or any other variation thereof, are intended to cover a non-exclusive inclusion. For example, a process, method, article, or apparatus that comprises a list of elements is not necessarily limited to only those elements but may include other elements not expressly listed or inherent to such process, method, article, or apparatus. Further, unless expressly stated to the contrary, "or" refers to an inclusive or and not to an exclusive or. For example, a condition A or B is satisfied by any one of the following: A is true (or present) and B is false (or not present), A is false (or not present) and B is true (or present), and both A and B are true (or present).

[00134] In addition, use of the "a" or "an" are employed to describe elements and components of the embodiments herein. This is done merely for convenience and to give a general sense of

the disclosure. This description should be read to include one or at least one and the singular also includes the plural unless it is obvious that it is meant otherwise.

[00135] Upon reading this disclosure, those of skill in the art will appreciate still additional alternative structural and functional designs for a system and a process for comparing timing constraints of circuits. Thus, while particular embodiments and applications have been illustrated and described, it is to be understood that the present invention is not limited to the precise construction and components disclosed herein and that various modifications, changes and variations which will be apparent to those skilled in the art may be made in the arrangement, operation and details of the method and apparatus disclosed herein without departing from the spirit and scope as defined in the appended claims.

What is claimed is:

1. A computer-implemented method for comparing a first circuit configuration with a second circuit configuration, wherein each circuit configuration comprises a set of timing nodes, edges between timing nodes, and timing constraints, the method comprising:

identifying a first source set of at least one timing node and a first sink set of at least one timing node in the first circuit configuration;

identifying a second source set of at least one timing node and a second sink set of at least one timing node in the second circuit configuration, the second source set corresponding to the first source set and the second sink set corresponding to the first sink set;

determining a first set of timing relationships, comprising aggregating all timing relationships encountered in a traversal of all timing paths from the first source set to the first sink set;

determining a second set of timing relationships, comprising aggregating all timing relationships encountered in a traversal of all timing paths from the second source set to the second sink set;

determining whether the first set of timing relationships is equivalent to the second set of timing relationships; and

responsive to determining that the first and second sets of timing relationships are not equivalent, indicating a mismatch between the first circuit configuration and the second circuit configuration.

2. The computer-implemented method of claim 1, wherein:

the first sink set consists of a single end point; and

the first source set consists of all start points with at least a timing path from the source timing node to the single end point.

3. The computer-implemented method of claim 1, wherein:

the first source set consists of a single end point; and

the first sink set consists of a single start point, the start point having a timing path to the single end point.

4. The computer-implemented method of claim 3, further comprising, responsive to determining that the first and second sets of timing relationships are equivalent:

identifying a first reconvergent point on at least two timing paths between the first source set and the first sink set, and identifying a corresponding second reconvergent point on at least two timing paths between the second source set and the second sink set;

identifying a first timing node connected by a first edge to the first reconvergent point, and identifying a corresponding second timing node connected by a second edge to the second reconvergent point;

determining a third set of timing relationships, comprising aggregating all timing relationships encountered in a traversal of all timing paths from the first source set to the first timing node;

determining a fourth set of timing relationships, comprising aggregating all timing relationships encountered in a traversal of all timing paths from the second source set to the second timing node;

determining whether the third set of timing relationships is equivalent to the fourth set of timing relationships; and

responsive to determining that the third and fourth sets of timing relationships are not equivalent, indicating a mismatch between the first circuit configuration and the second circuit configuration.

5. The computer-implemented method of claim 1, wherein aggregating all timing relationships encountered in a traversal of all timing paths from the first source set to the first sink set comprises:

identifying edges connecting a first timing node to a second timing node on timing paths from the first source set to the first sink set; and

for the second timing node, determining an aggregate set of timing relationships associated with the second timing node as a union of a set of timing

relationships associated with the first timing node and a set of timing relationships associated with the edge.

6. The computer-implemented method of claim 1, wherein aggregating all timing relationships encountered in a traversal of all timing paths from the first source set to the first sink set comprises:

identifying first timing nodes connected by first edges to reconvergent points and second timing nodes connected by second edges to reconvergent points; and

for the reconvergent points, determining an aggregate set of timing relationships associated with the reconvergent point as a union of a set of timing relationships associated with the first timing node, a set of timing relationships associated with the second timing node, a set of timing relationships associated with the first edge, and a set of timing relationships associated with the second edge.

7. The computer-implemented method of claim 1, wherein determining a first set of timing relationships comprises eliminating a first timing relationship from the first set if a second timing relationship is determined to override the first timing relationship based on precedence rules.

8. The computer-implemented method of claim 1, wherein aggregating all timing relationships encountered in a traversal of all timing paths from the first source set to the first sink set comprises:

determining a set of timing relationships for a set of timing nodes consisting of the first source set;

expanding the set of timing nodes to include neighboring timing nodes, and updating the set of timing relationships by aggregating timing relationships associated with expanding the set of timing nodes;

repeating the previous step until the set of timing nodes includes the first sink set.

9. The computer-implemented method of claim 1, wherein the traversal of all timing paths from the first source set is a breadth first traversal.

10. The computer-implemented method of claim 1, wherein the first source set consists of at least one start point and the first sink set consists of at least one end point.

11. The computer-implemented method of claim 1, wherein the first source set consists of at least one end point and the first sink set consists of at least one start point.

12. A computer-implemented method for comparing a first circuit configuration with a second circuit configuration, wherein each circuit configuration comprises a set of timing nodes and timing relationships, the method comprising:

identifying a first source set of at least one timing node and a first sink set of at least one timing node in the first circuit configuration;

identifying a second source set of at least one timing node and a second sink set of at least one timing node in the second circuit configuration, the second source set corresponding to the first source set and the second sink set corresponding to the first sink set;

determining a first aggregate set of timing relationships for timing paths from the first source set to the first sink set, comprising accounting for interactions between timing relationships ;

determining a second aggregate set of timing relationships for timing paths from the second source set to the second sink set; and

determining whether the first aggregate set of timing relationships is equivalent to the second aggregate set of timing relationships.

13. The computer-implemented method of claim 12, wherein accounting for interactions between timing relationships from different timing paths comprises identifying dominant timing relationships satisfied on the timing paths based on precedence rules

14. The computer-implemented method of claim 12, wherein accounting for interactions between timing relationships for timing paths comprises resolving redundancies in timing relationships.

15. A computer-implemented method for generating merged mode corresponding to a plurality of individual modes for a circuit comprising netlists, each mode comprising timing constraints, the method comprising:

generating a merged mode by combining timing constraints from individual modes, wherein each mode comprises a set of timing constraints and each mode is associated with a set of timing relationships;

determining a set of timing relationships for the merged mode and a set of timing relationship for each individual modes, wherein each timing relationship is associated with a first timing node and a second timing node;

comparing timing relationships for the merged mode with timing relationships for the individual modes to identify an extraneous timing relationship present in the merged mode, wherein the extraneous timing relationship is not present in any individual mode; and

adding a timing constraint to the merged mode such that addition of the timing constraint eliminates the extraneous timing relationship.

16. The computer-implemented method of claim 15, further comprising:

determining that a clock is a capture clock of a timing node;

determining that the clock is not the capture clock of the timing node in each of the individual modes; and

disabling the capture clock of the end point in the merged mode by adding the timing constraint to the merged mode.

17. The computer-implemented method of claim 15, wherein the extraneous timing relationship is determined to exist between a start point and an end point in the merged mode and no timing relationship is determined to exist between the start point and end point in any individual mode, and wherein the timing constraint added to the merged mode disables the path between the start point and the end point.

18. The computer-implemented method of claim 15, wherein a plurality of timing relationship exist between a start point and an end point in the merged mode, the method further comprising:

determining a reconvergent node between the start point and end point;

comparing timing relationships between data paths connected to the reconvergent node;

determining a data path connected to the reconvergent node such that a timing relationship for the data path exists in the merged mode and no timing relationship exists for the data path in any individual mode; and

disabling the data path by adding the timing constraint to the merged mode.

19. The computer-implemented method of claim 15, wherein each timing relationship comprises a launch clock and a capture clock.

20. The computer-implemented method of claim 19, wherein each timing relationship further comprises one or more timing constraints associated with at least one of the start point and the end point.

21. The computer-implemented method of claim 15, the method further comprising:

determining first clock and a second clock reaching a timing node in the merged mode such that the first clock is disabled in every individual mode in which the second clock is enabled and the second clock is disabled in every individual mode in which the first clock is enabled; and

disabling interactions between the first clock and the second clock in the merged mode by adding the timing constraint to the merged mode .

22. The computer-implemented method of claim 15, wherein combining timing constraints from individual modes to the merged mode comprises:

determining whether the timing constraint is of a type that adds a new timing relationship to any mode; and

responsive to determining that the timing constraint is present in at least one of the individual modes, adding the timing constraint to the merged mode.

23. The computer-implemented method of claim 15, wherein combining timing constraints from individual modes to the merged mode comprises:

determining whether the timing constraint is of a type that modifies an existing timing relationship in any mode; and

responsive to determining that the timing constraint is present in at least one of the individual modes, adding the timing constraint to the merged mode.

24. The computer-implemented method of claim 15, wherein combining timing constraints from individual modes to the merged mode comprises:

determining whether the timing constraint is of a type that removes an existing timing relationship in any mode; and

responsive to determining that the timing constraint is present in all of the individual modes, adding the timing constraint to the merged mode.

25. The computer-implemented method of claim 15, further comprising:

identifying a timing node for which each individual mode specifies a constant value of one or zero and adding a timing constraint to the merged mode indicating the value at the timing node is constant.

26. The computer-implemented method of claim 15, wherein combining timing constraints from individual modes to the merged mode comprises:

comparing the set of timing constraints that do not affect timing relationships across all the individual modes; and

responsive to determining that the set of timing constraints are same across all individual modes, adding the set of timing constraints to the merged mode.

27. The computer-implemented method of claim 15, further comprising:

comparing the set of timing constraints that do not affect timing relationships across all the individual modes; and

responsive to determining that the set of timing constraints that do not affect timing relationships are not same across all individual modes, flagging an error.

28. A computer-implemented system for generating a merged mode corresponding to a plurality of individual modes for a circuit comprising netlists, each mode comprising timing constraints, the system comprising:

a computer processor; and

a computer-readable storage medium storing computer program modules configured to execute on the computer processor, the computer program modules comprising:

a merge mode generator module configured to:

generate a merged mode by combining timing constraints from individual modes, wherein each mode comprises a set of timing constraints and each mode is associated with a set of timing relationships;

determine a set of timing relationships for the merged mode and a set of timing relationship for each individual modes, wherein each timing relationship is associated with a first timing node and a second timing node;

compare timing relationships for the merged mode with timing relationships for the individual modes to identify an extraneous timing relationship present in the merged mode, wherein the extraneous timing relationship is not present in any individual mode; and

add a timing constraint to the merged mode such that addition of the timing constraint eliminates the extraneous timing relationship.

29. A computer readable storage medium storing a computer program product including computer instructions configured to cause a processor of a computer to perform a computer-implemented method for generating a merged mode corresponding to a plurality of

individual modes for a circuit comprising netlists, each mode comprising timing constraints, the computer program product comprising:

a merge mode generator module configured to:

generate a merged mode by combining timing constraints from individual modes, wherein each mode is associated with a set of timing relationships;

determine a set of timing relationships for the merged mode and a set of timing relationship for each individual modes, wherein each timing relationship is associated with a first timing node and a second timing node;

compare timing relationships for the merged mode with timing relationships for the individual modes to identify an extraneous timing relationship present in the merged mode, wherein the extraneous timing relationship is not present in any individual mode; and

add a timing constraint to the merged mode such that addition of the timing constraint eliminates the extraneous timing relationship.

30. A computer-implemented method for verifying equivalence between a merged mode and a plurality of individual modes for a circuit comprising netlists, each mode comprising timing constraints, the method comprising:

identifying a source set of timing nodes and a sink set of timing nodes in a set of netlists, wherein a merged mode and a plurality of individual modes are defined for the netlists, the merged mode and the individual modes comprising timing constraints for netlists in the set of netlists;

determining an aggregate set of timing relationships encountered by traversing from the source set of timing nodes to the sink set of timing nodes for each individual mode and the merged mode;

comparing timing relationships in the aggregate set from the merged mode with timing relationships in aggregate sets from the individual modes; and

responsive to identifying a mismatch between timing relationships in the aggregate set from the merged mode and timing relationships in aggregate sets from the individual modes, presenting an error message.

31. The computer-implemented method of claim 30, wherein identifying the mismatch comprises identifying a timing relationship present in an aggregate set from an individual mode, such that the timing relationship is absent in the aggregate set from the merged mode.

32. The computer-implemented method of claim 30, wherein identifying the mismatch comprises identifying a timing relationship present in the aggregate set from the merged mode, such that the timing relationship is absent in all the aggregate sets from the individual modes.

33. The computer-implemented method of claim 30, wherein:

the sink set consists of a single end point; and

the source set consists of all start points with at least a timing path from the timing node in the source set to the single end point.

34. The computer-implemented method of claim 30, wherein:

the source set consists of a single end point; and

the sink set consists of a single start point, the single start point having a timing path to the single end point.

35. The computer-implemented method of claim 30, wherein:

the sink set consists of a reconvergent node on at least two timing paths from a start point to an end point; and

the source set consists of a timing node on one of the timing paths, the timing node connected to the reconvergent node by an edge.

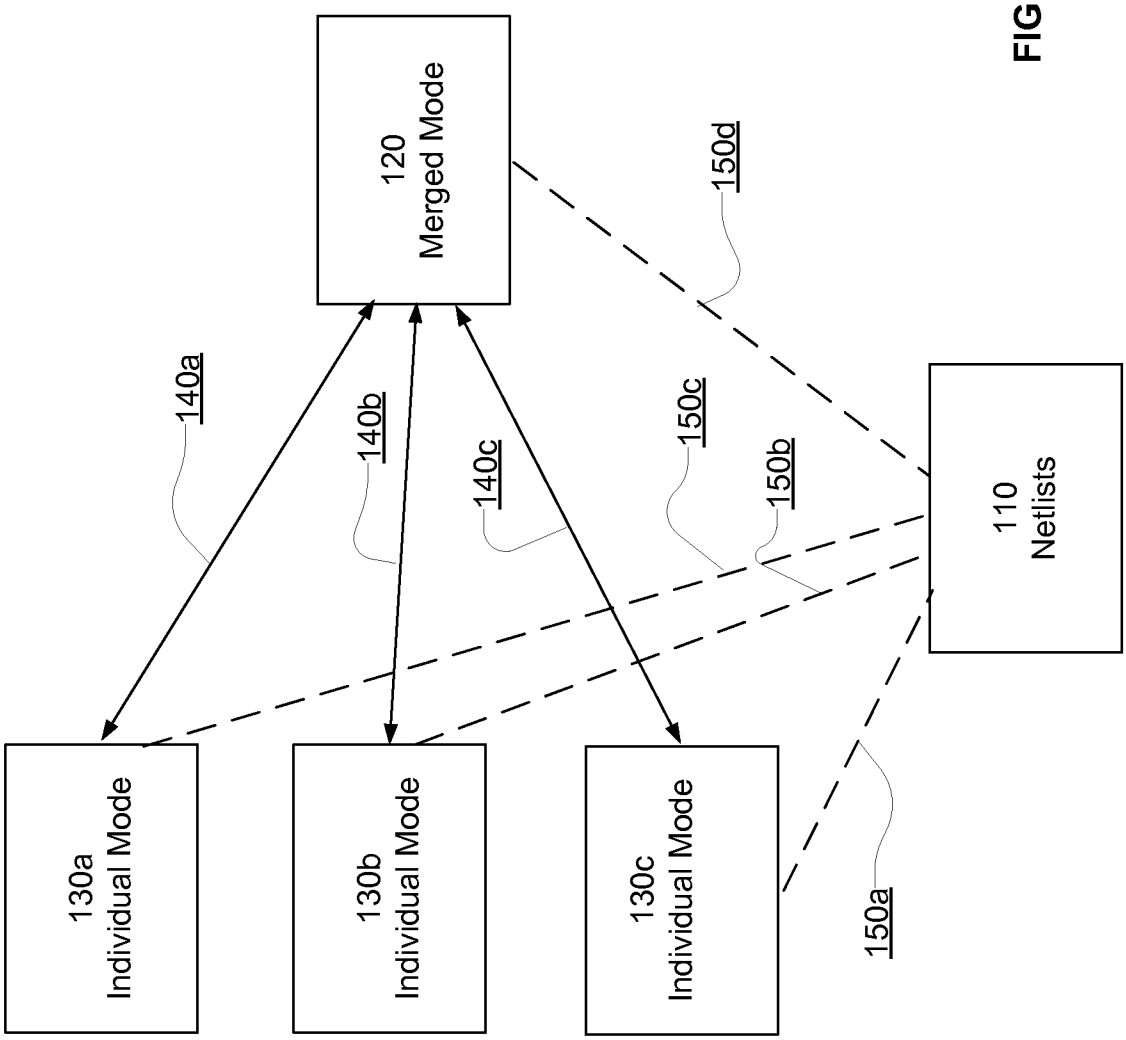

120
Merged Mode
110
Netlists
130a
Individual Mode
130b
Individual Mode
130c
Individual Mode
140a
140b
140c
150a
150b
150c
150d

FIG. 1

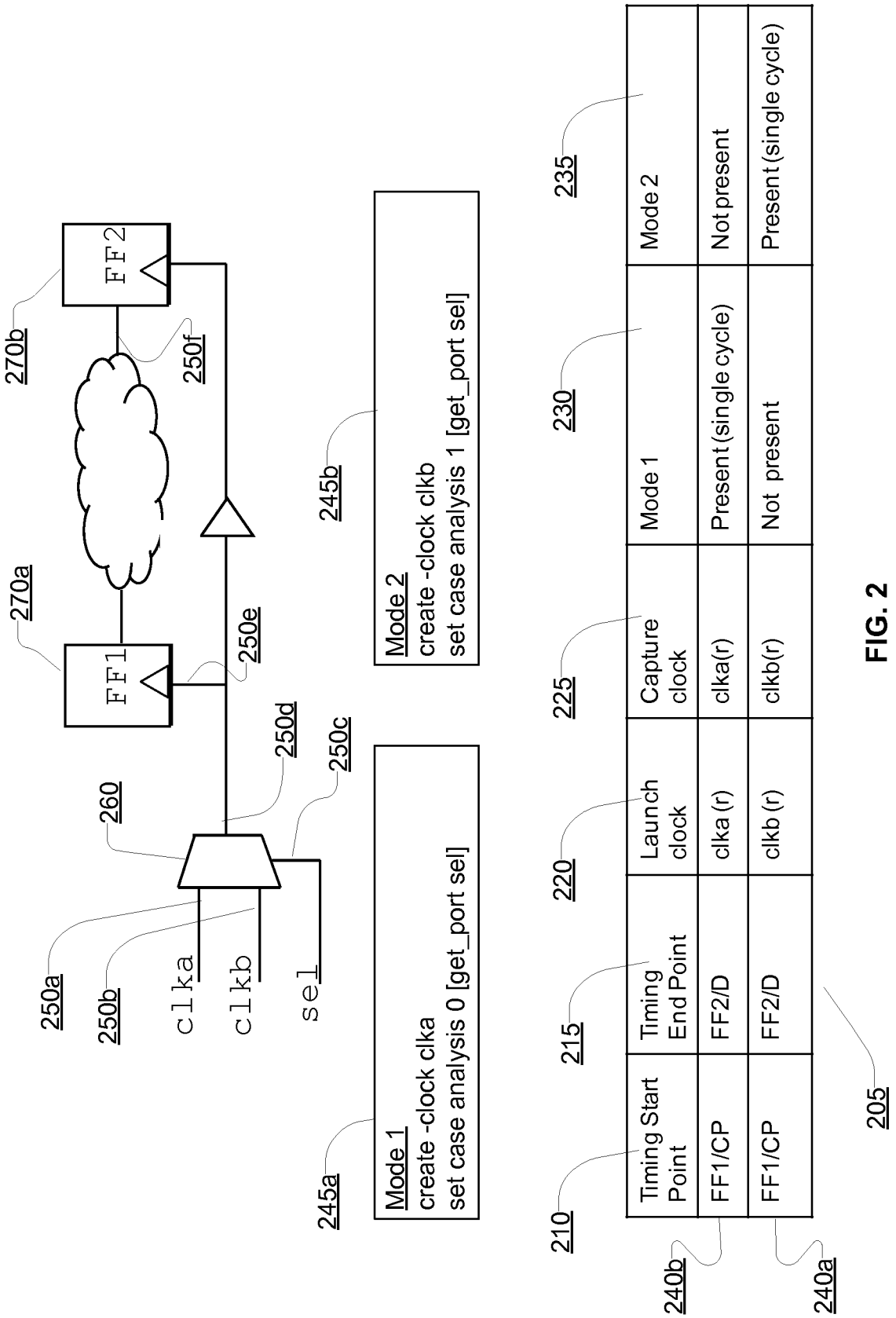

270b
FF2
250f
270a
FF1
250e
260
250d
250c
250a
250b
clka
clkb
sel
245b
Mode 2
create -clock clkb
set case analysis 1 [get_port sel]
245a
Mode 1
create -clock clka
set case analysis 0 [get_port sel]
235
230
225
220
215
210
Timing Start Point
Timing End Point
Launch clock
Capture clock
Mode 1
Mode 2
FF1/CP
FF2/D
clka (r)
clka(r)
Present (single cycle)
Not present
FF1/CP
FF2/D
clkb (r)
clkb(r)
Not present
Present (single cycle)
240b
240a
205

FIG. 2

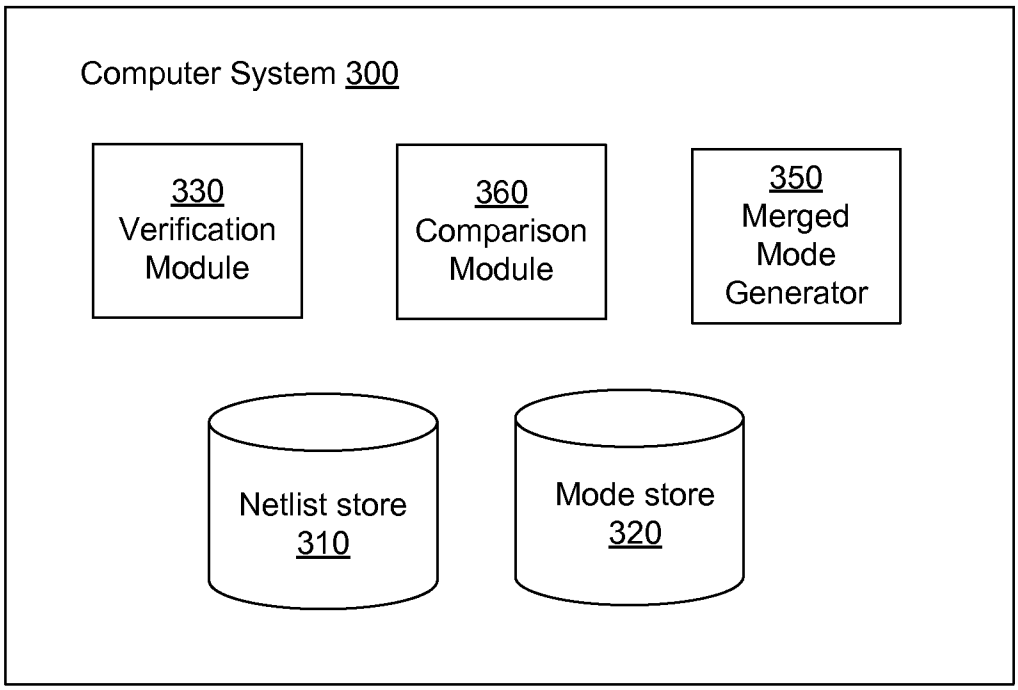

Computer System 300
330
Verification
Module
360
Comparison
Module
350
Merged
Mode
Generator
Netlist store
310
Mode store
320

FIG. 3

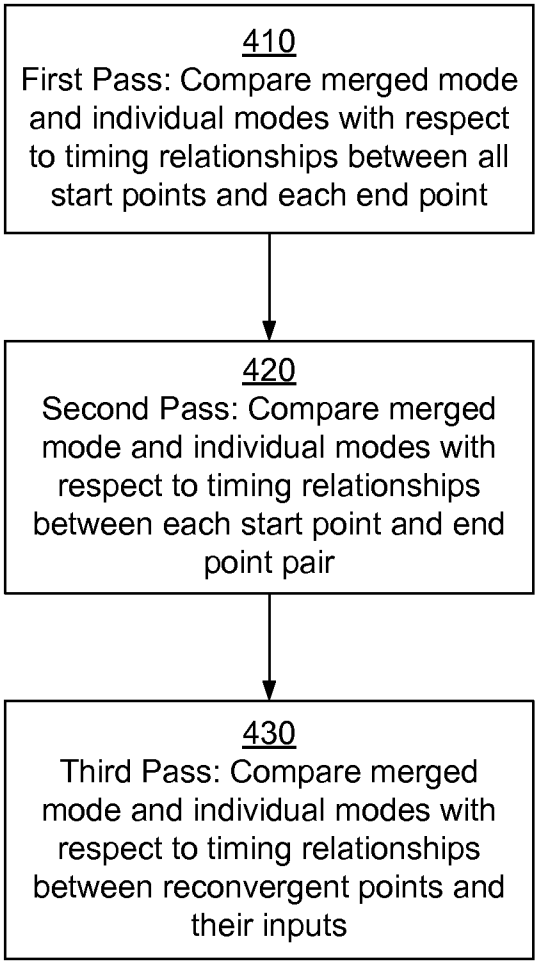

410
First Pass: Compare merged mode and individual modes with respect to timing relationships between all start points and each end point
420
Second Pass: Compare merged mode and individual modes with respect to timing relationships between each start point and end point pair
430
Third Pass: Compare merged mode and individual modes with respect to timing relationships between reconvergent points and their inputs

FIG. 4

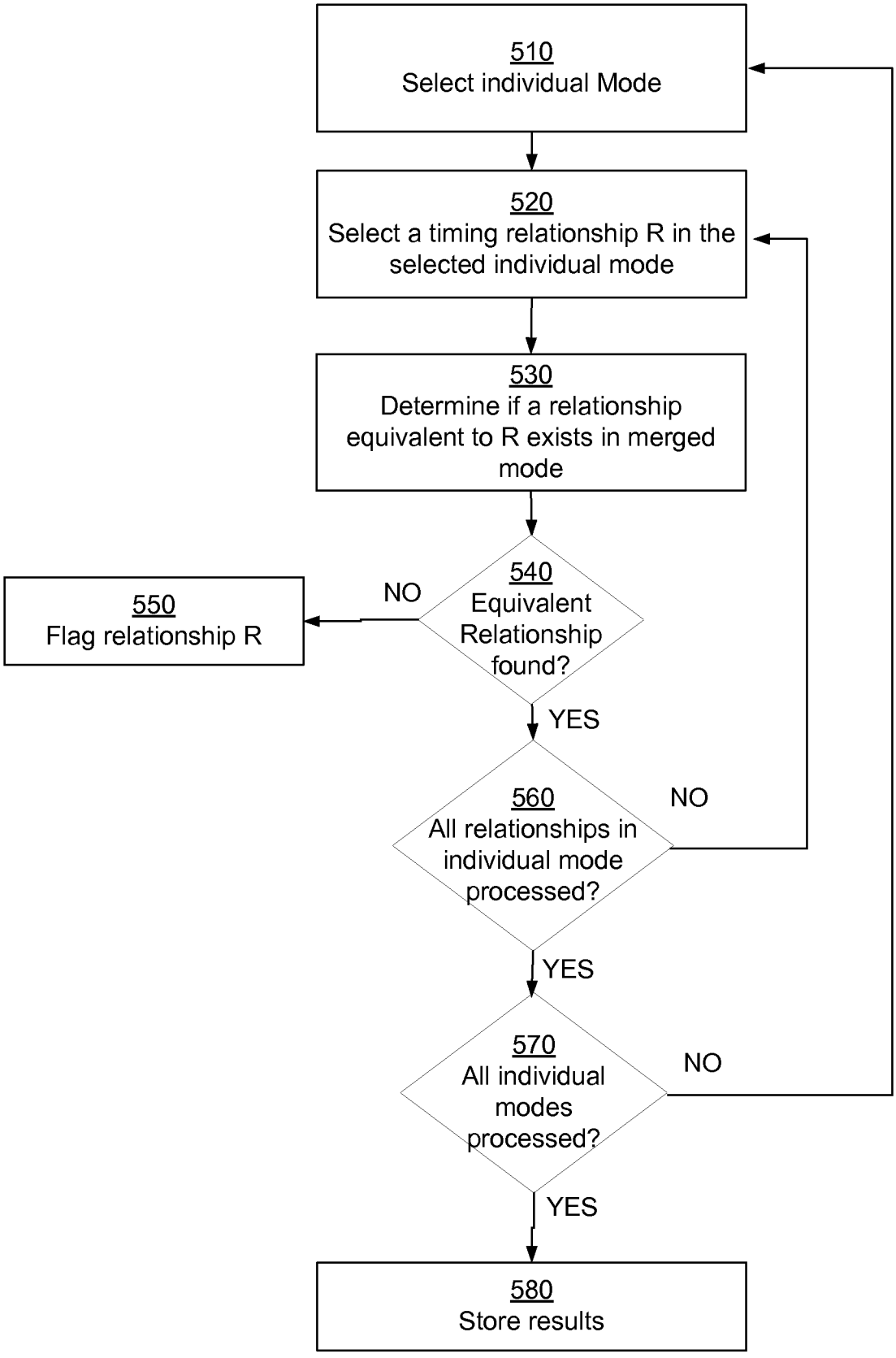

510
Select individual Mode
520
Select a timing relationship R in the selected individual mode
530
Determine if a relationship equivalent to R exists in merged mode
540
Equivalent Relationship found?
NO
550
Flag relationship R
YES
560
All relationships in individual mode processed?
NO
YES
570
All individual modes processed?
NO
YES
580
Store results

FIG. 5

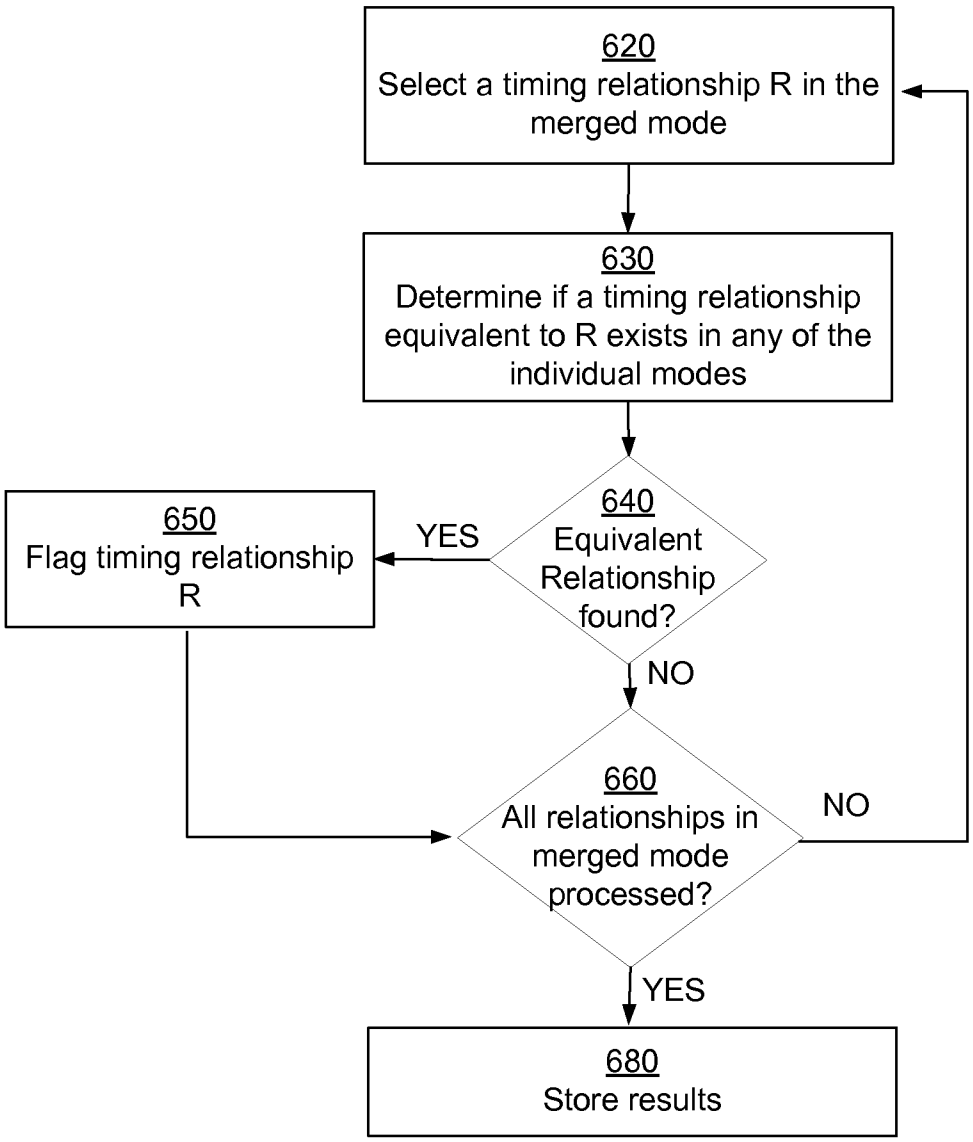

620
Select a timing relationship R in the merged mode
630
Determine if a timing relationship equivalent to R exists in any of the individual modes
640
Equivalent Relationship found?
YES
650
Flag timing relationship R
NO
660
All relationships in merged mode processed?
NO
YES
680
Store results

FIG. 6

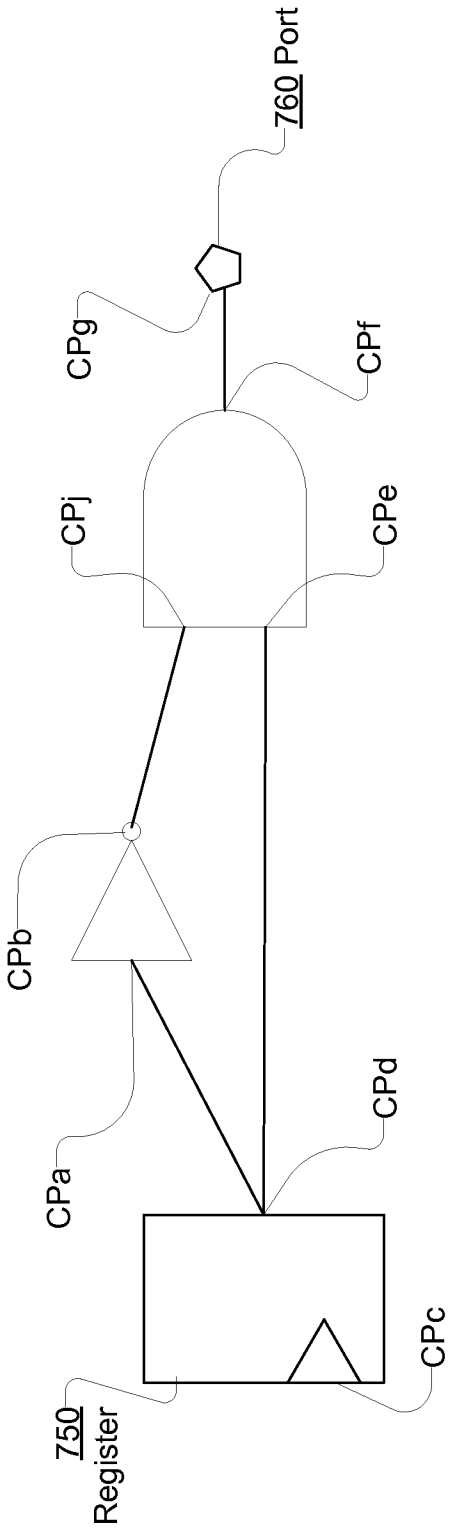

760 Port
CPg
CPf
CPj
CPe
CPb
CPa
CPd
CPc
750 Register

FIG. 7(a)

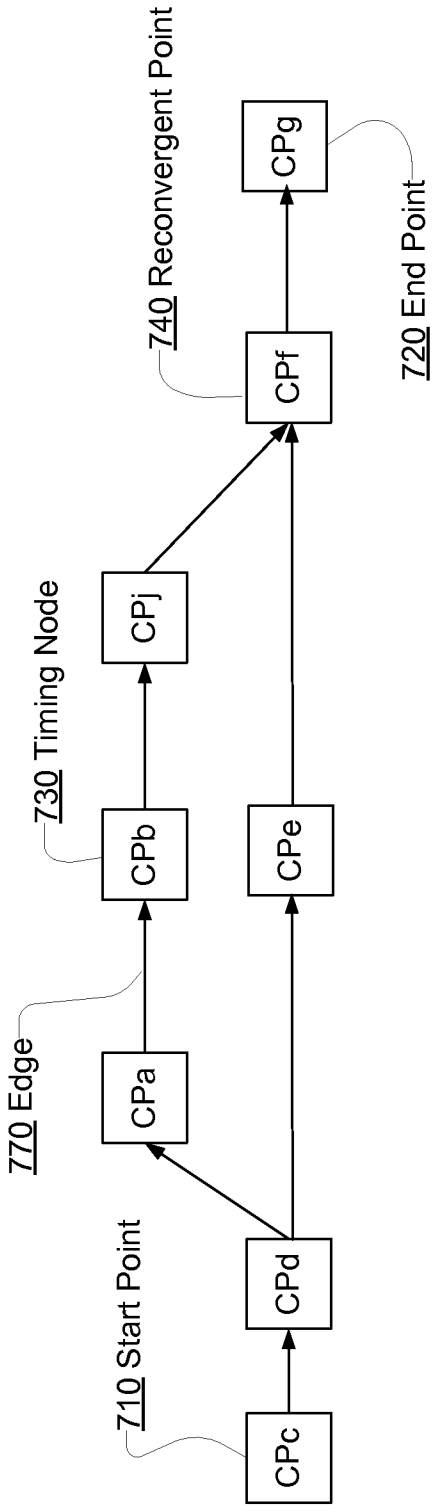

740 Reconvergent Point
CPg
CPf
720 End Point
730 Timing Node
CPj
CPb
CPe
770 Edge
CPa
710 Start Point
CPd
CPc

FIG. 7(b)

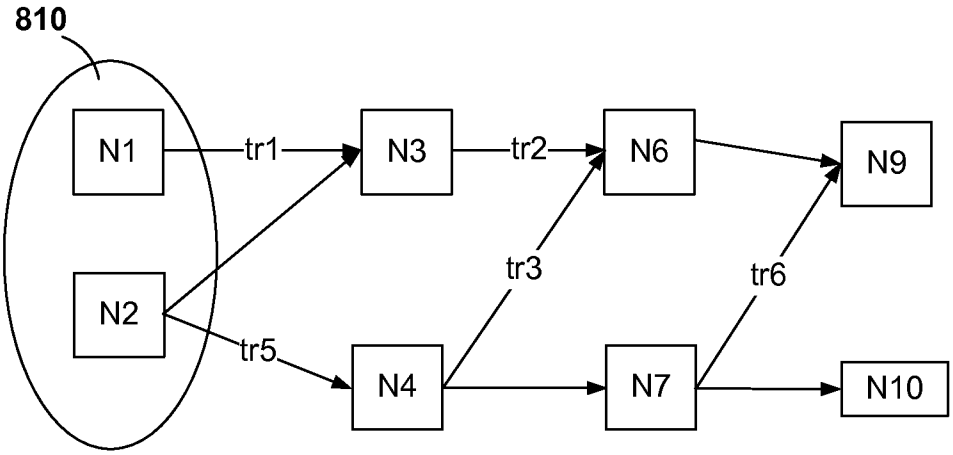

810
N1
tr1
N3
tr2
N6
N9
N2
tr3
tr6
tr5
N4
N7
N10

FIG. 8(a)

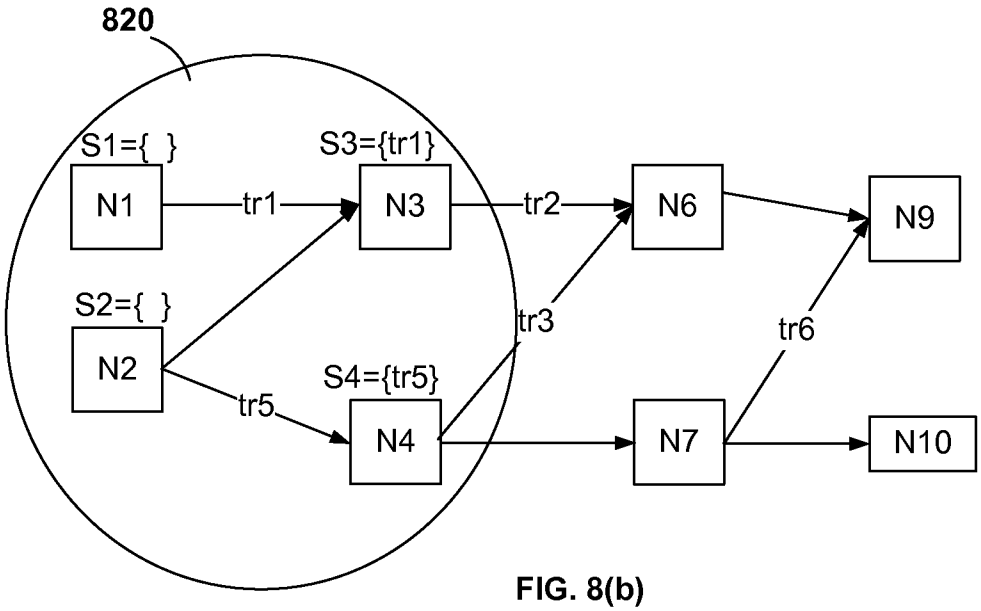

820
S1={ }
S3={tr1}
N1
tr1
N3
tr2
N6
N9
S2={ }
tr3
tr6
N2
S4={tr5}
tr5
N4
N7
N10

FIG. 8(b)

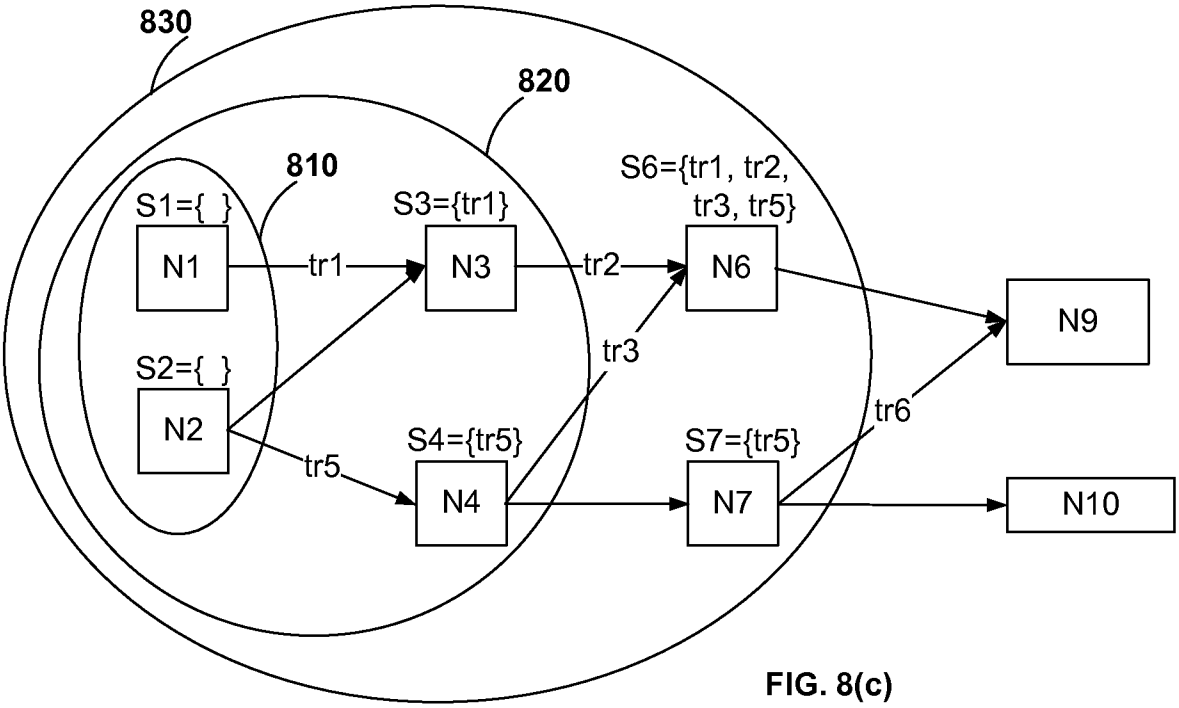

830
820
810
S1={ }
S3={tr1}
S6={tr1, tr2, tr3, tr5}
N1
tr1
N3
tr2
N6
N9
tr3
S2={ }
N2
tr6
S4={tr5}
S7={tr5}
tr5
N4
N7
N10

FIG. 8(c)

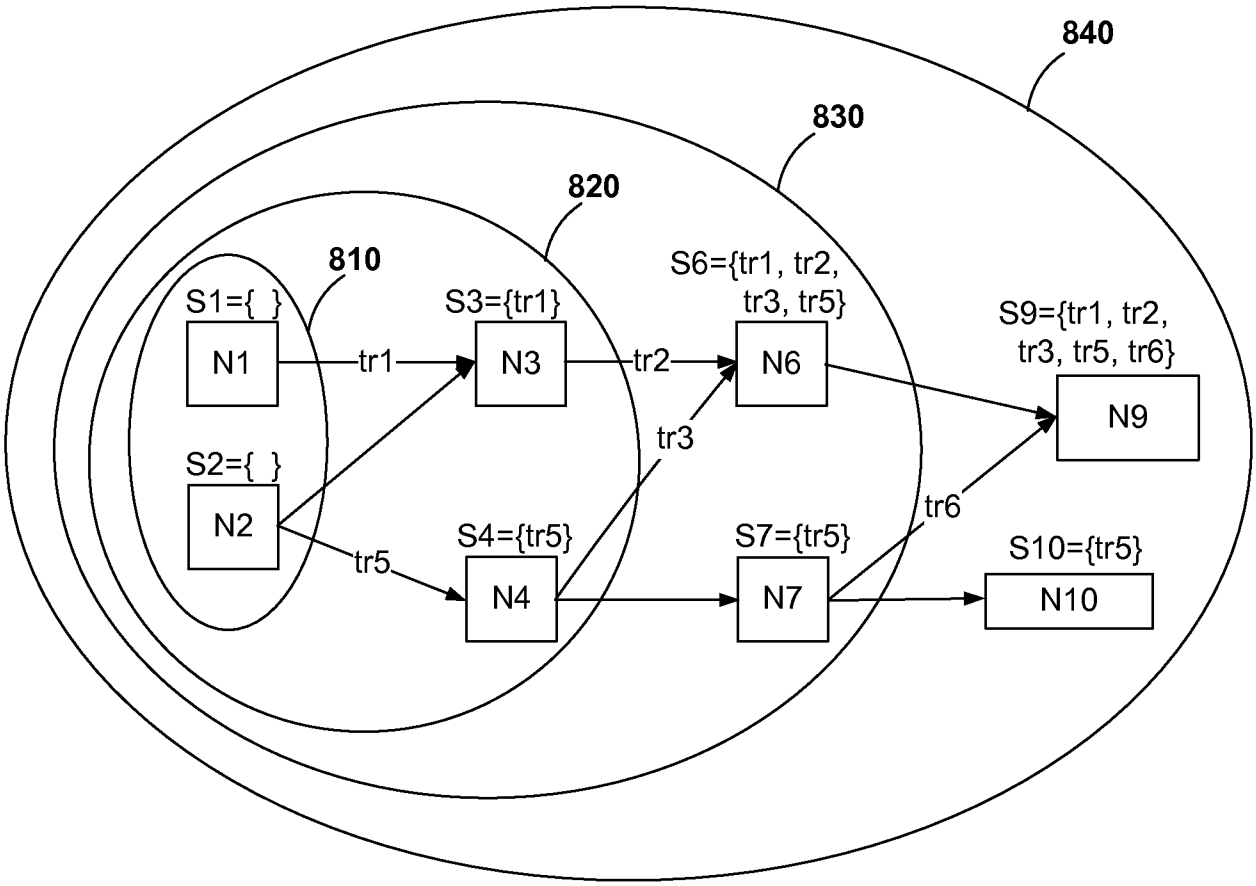

840
830
820
810
S1={ }
S3={tr1}
S6={tr1, tr2, tr3, tr5}
S9={tr1, tr2, tr3, tr5, tr6}
N1
tr1
N3
tr2
N6
N9
tr3
S2={ }
N2
tr6
S4={tr5}
S7={tr5}
S10={tr5}
tr5
N4
N7
N10

FIG. 8(d)

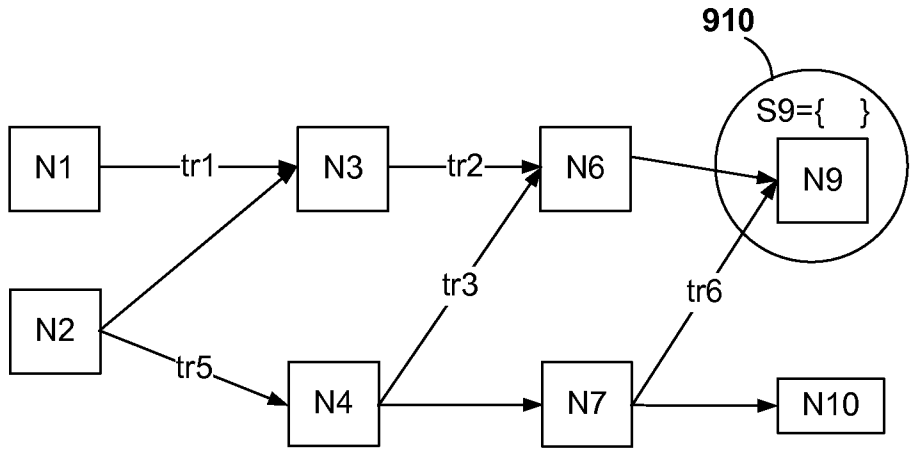

910
S9={ }
N1
tr1
N3
tr2
N6
N9
tr3
tr6
N2
tr5
N4
N7
N10

FIG. 9(a)

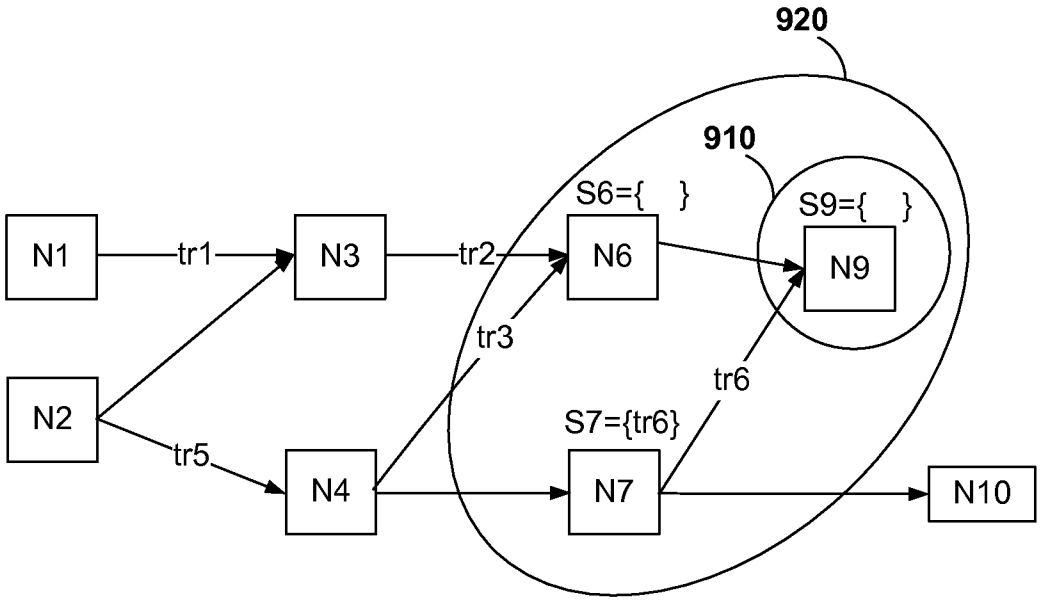

920
910
S6={ }
S9={ }
N1
tr1
N3
tr2
N6
N9
tr3
tr6
N2
S7={tr6}
tr5
N4
N7
N10

FIG. 9(b)

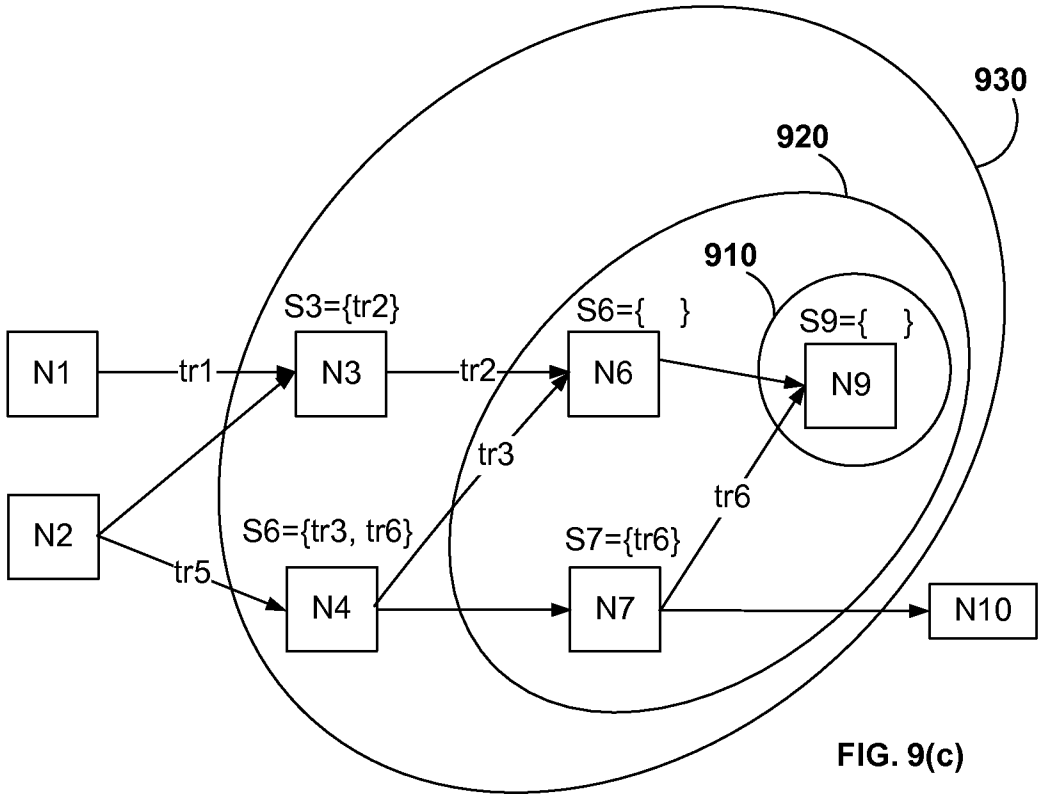

930
920
910
S3={tr2}
S6={ }
S9={ }
N1
tr1
N3
tr2
N6
N9
tr3
tr6
N2
S6={tr3, tr6}
S7={tr6}
tr5
N4
N7
N10

FIG. 9(c)

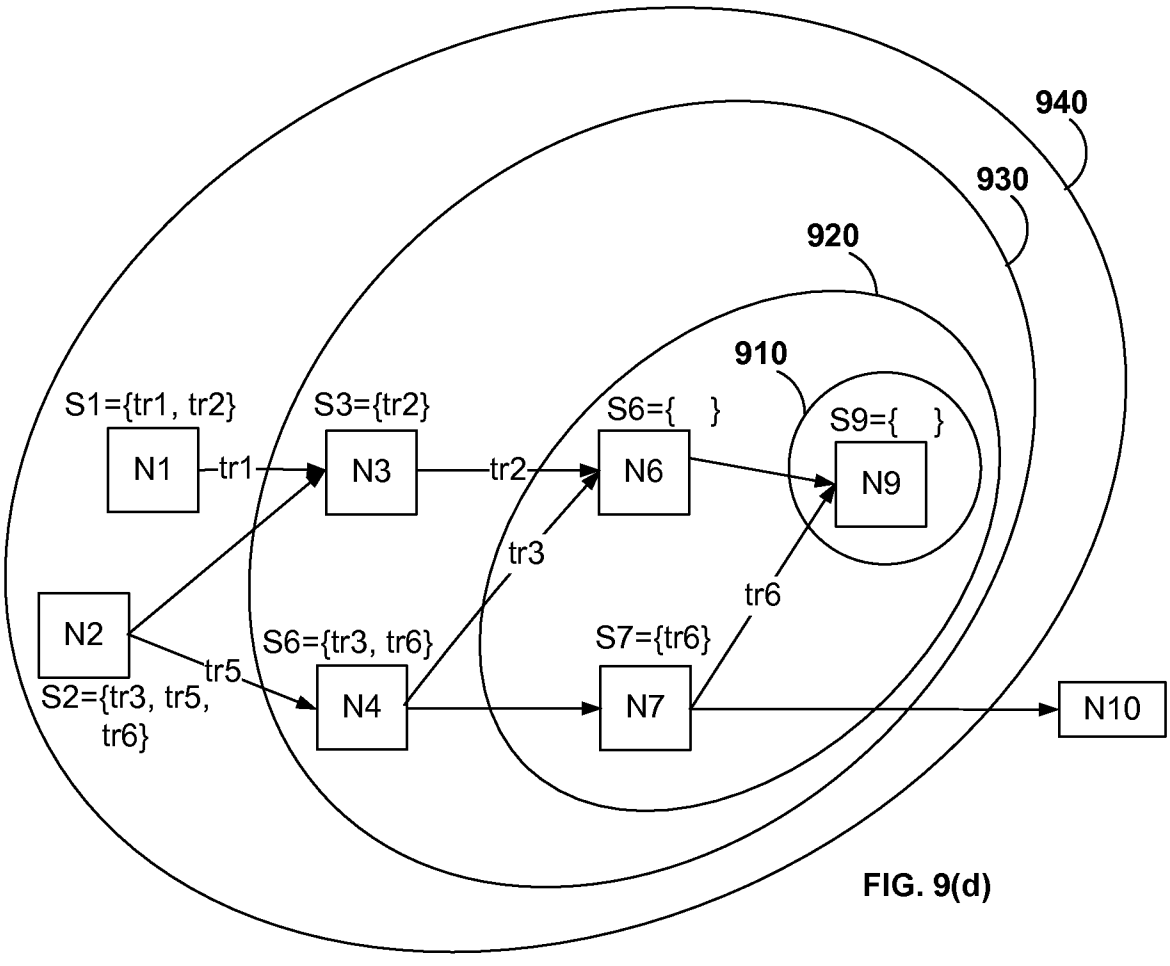
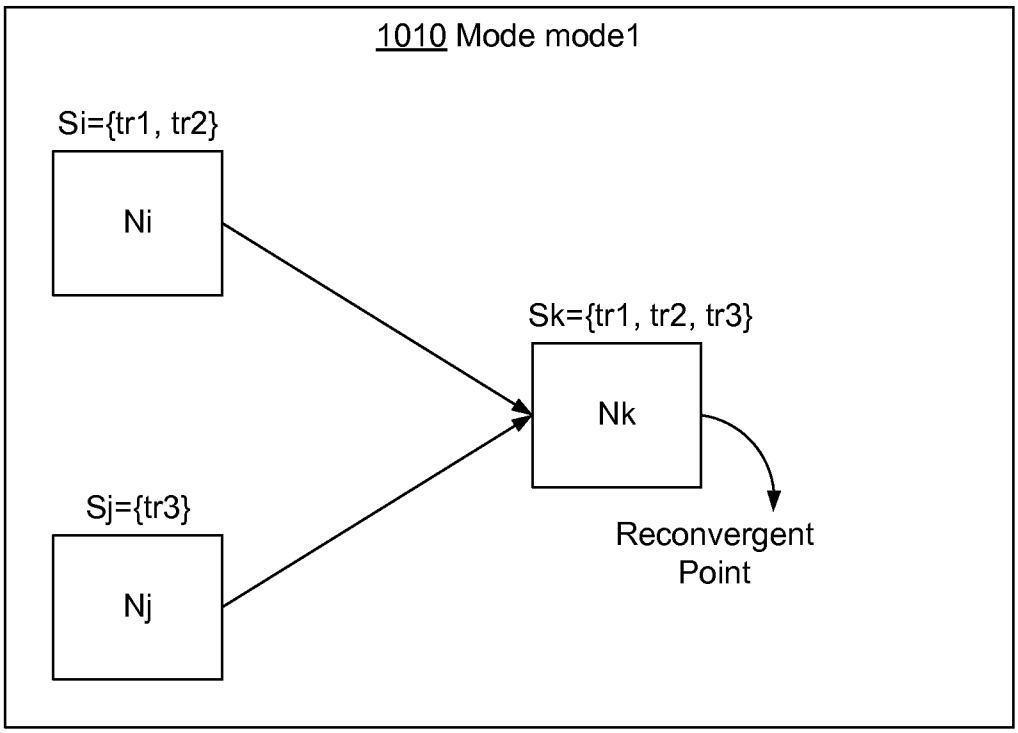

940
930
920
910
S1={tr1, tr2}
S3={tr2}
S6={ }
S9={ }
N1
tr1
N3
tr2
N6
N9
tr3
tr6
N2
S6={tr3, tr6}
S7={tr6}
tr5
S2={tr3, tr5, tr6}
N4
N7
N10

FIG. 9(d)


1010 Mode mode1
Si={tr1, tr2}
Ni
Sk={tr1, tr2, tr3}
Nk
Sj={tr3}
Nj
Reconvergent Point

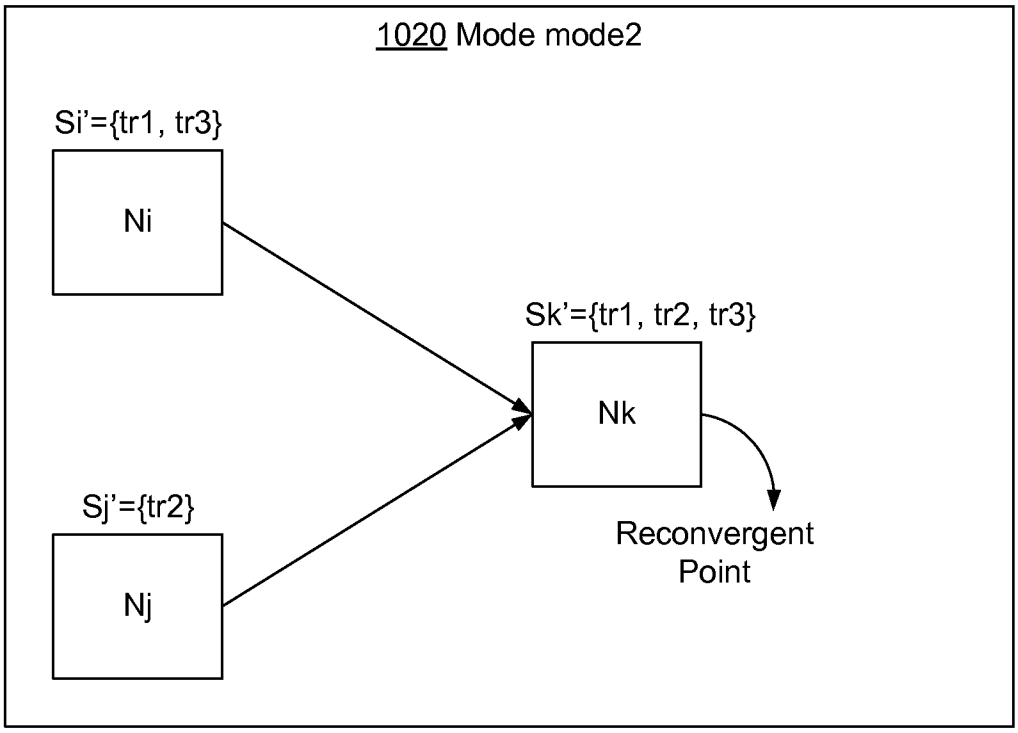

1020 Mode mode2
Si'={tr1, tr3}
Ni
Sk'={tr1, tr2, tr3}
Nk
Sj'={tr2}
Nj
Reconvergent Point

FIG. 10

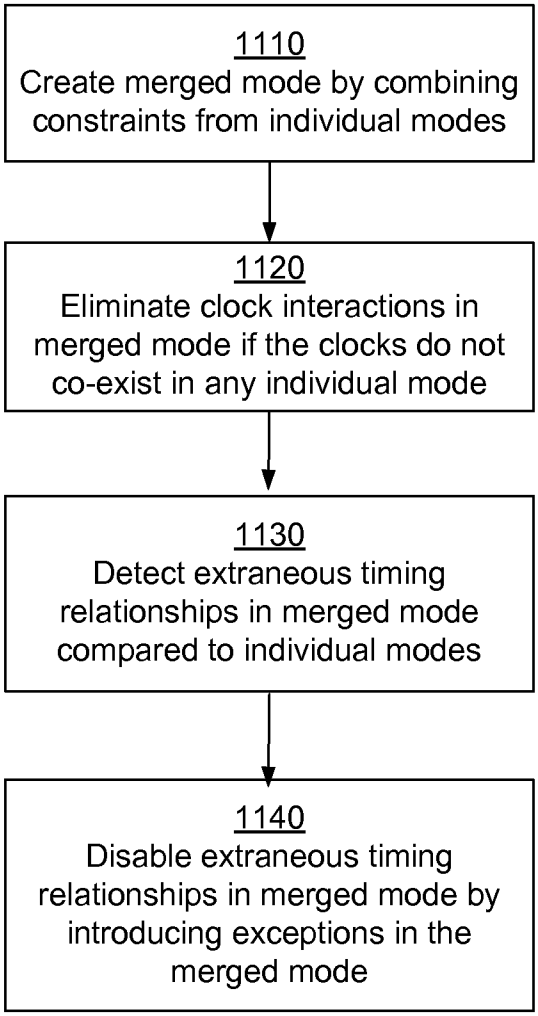

1110
Create merged mode by combining constraints from individual modes
1120
Eliminate clock interactions in merged mode if the clocks do not co-exist in any individual mode
1130
Detect extraneous timing relationships in merged mode compared to individual modes
1140
Disable extraneous timing relationships in merged mode by introducing exceptions in the merged mode

FIG. 11

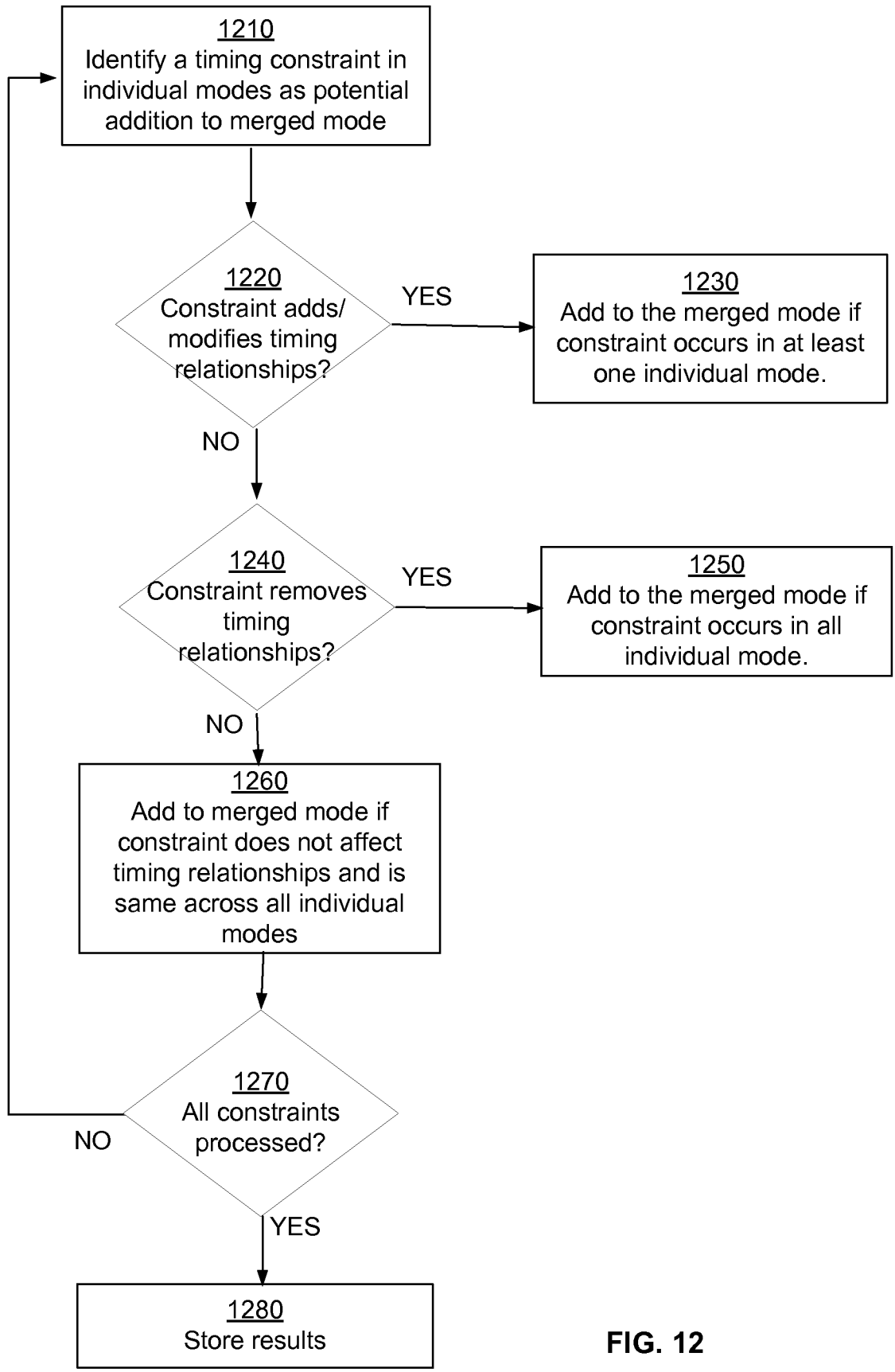

1210
Identify a timing constraint in individual modes as potential addition to merged mode
1220
Constraint adds/ modifies timing relationships?
YES
1230
Add to the merged mode if constraint occurs in at least one individual mode.
NO
1240
Constraint removes timing relationships?
YES
1250
Add to the merged mode if constraint occurs in all individual mode.
NO
1260
Add to merged mode if constraint does not affect timing relationships and is same across all individual modes
1270
All constraints processed?
NO
YES
1280
Store results

FIG. 12

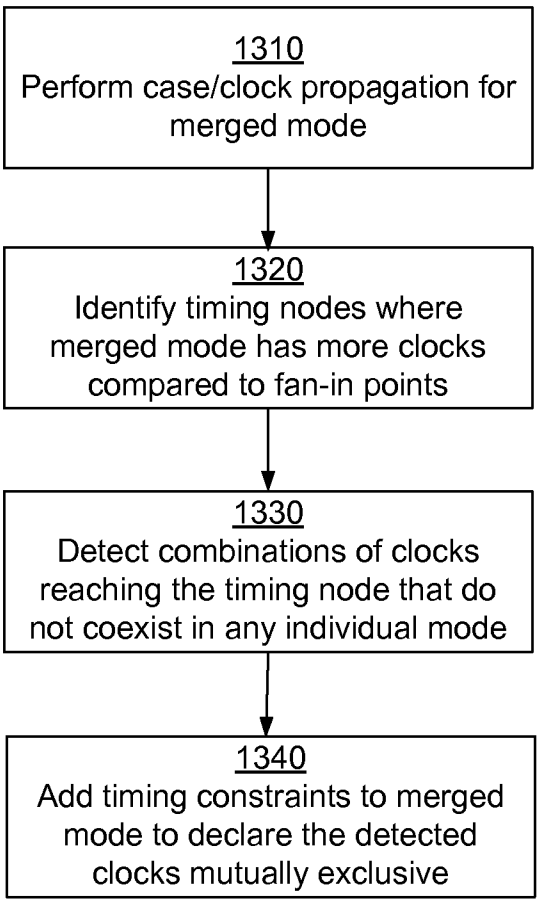


FIG. 13(a)

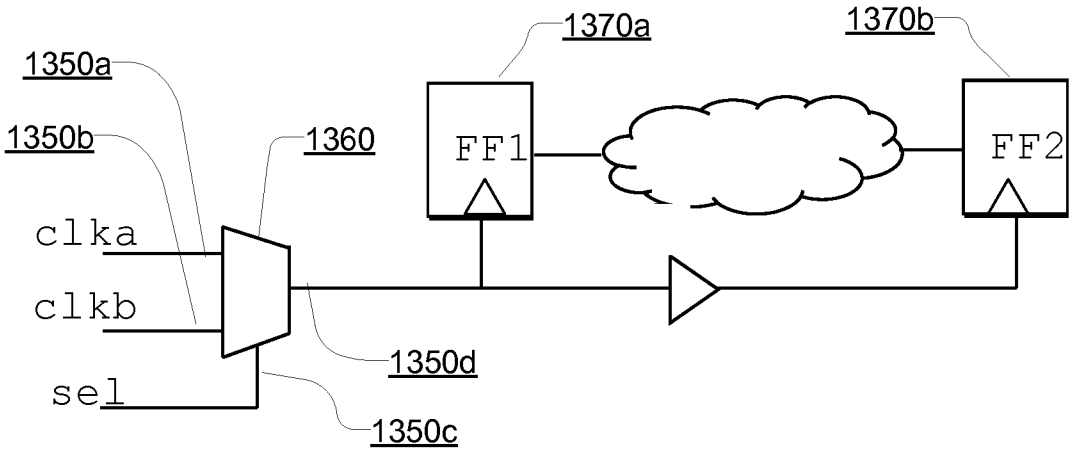


FIG. 13(b)

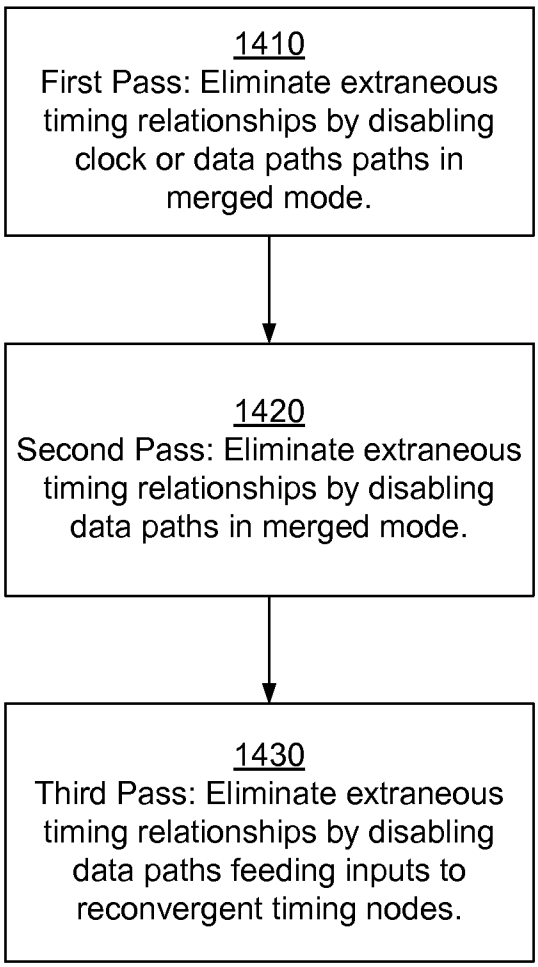

1410
First Pass: Eliminate extraneous timing relationships by disabling clock or data paths paths in merged mode.
1420
Second Pass: Eliminate extraneous timing relationships by disabling data paths in merged mode.
1430
Third Pass: Eliminate extraneous timing relationships by disabling data paths feeding inputs to reconvergent timing nodes.

FIG. 14

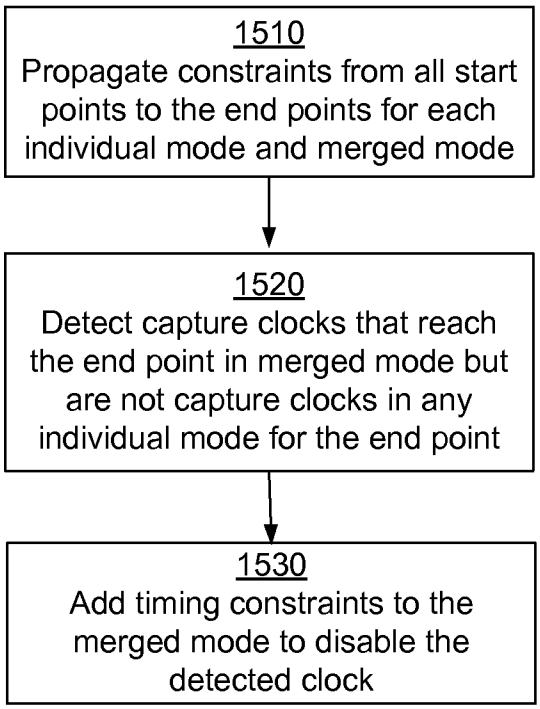


FIG. 15(a)

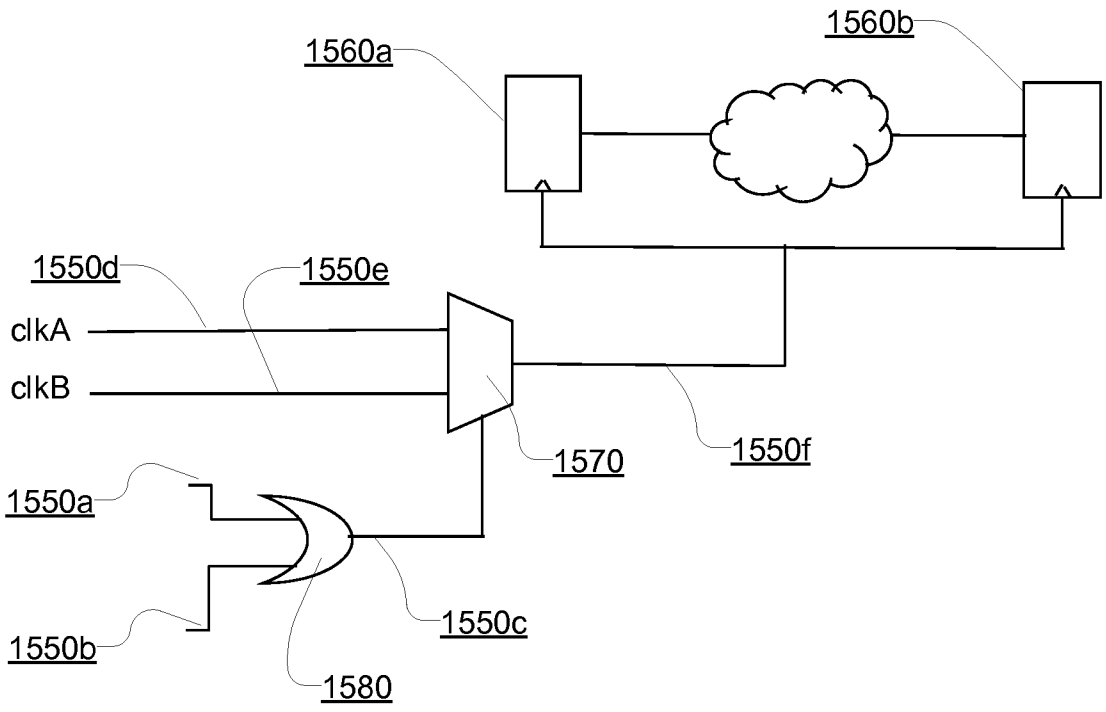


FIG. 15(b)

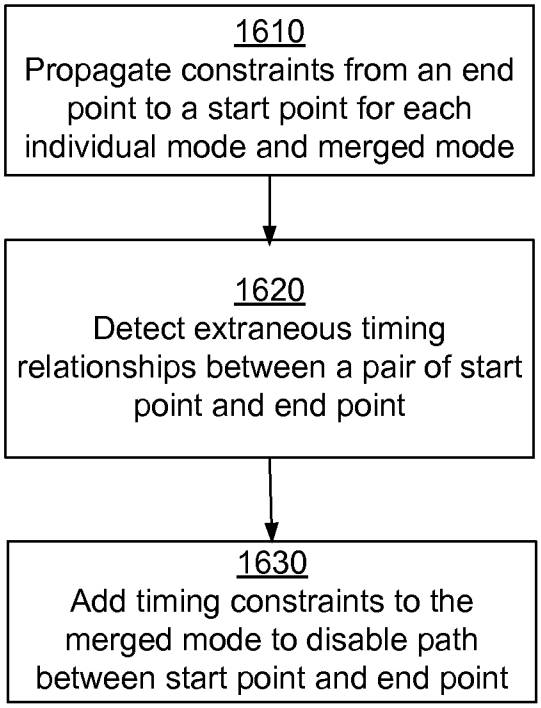


FIG. 16(a)

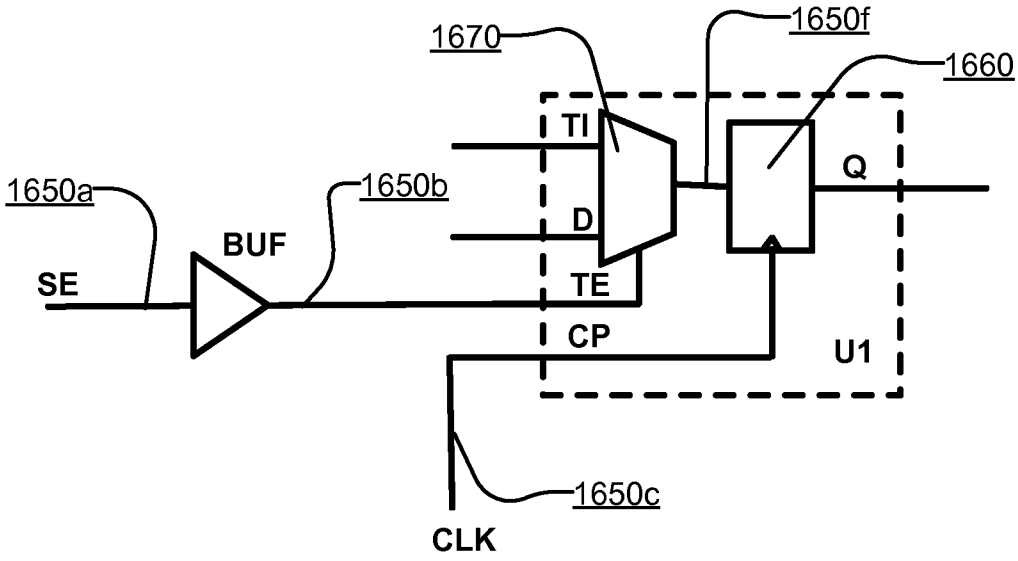


FIG. 16(b)

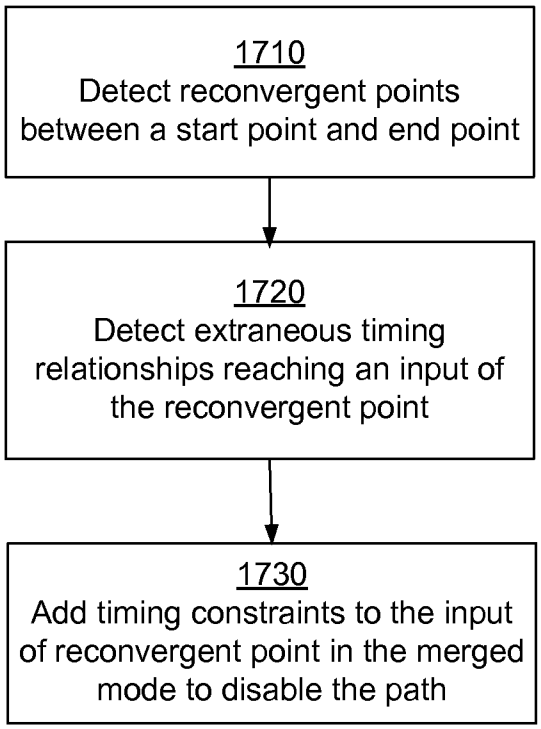


FIG. 17(a)

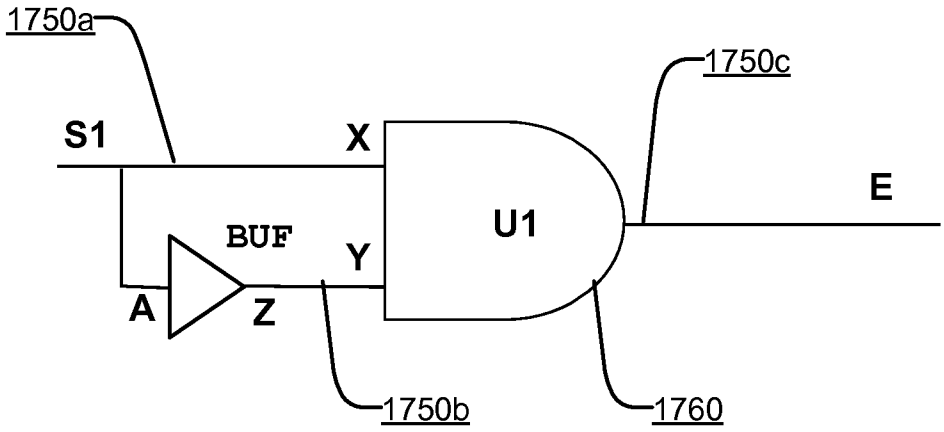


FIG. 17(b)

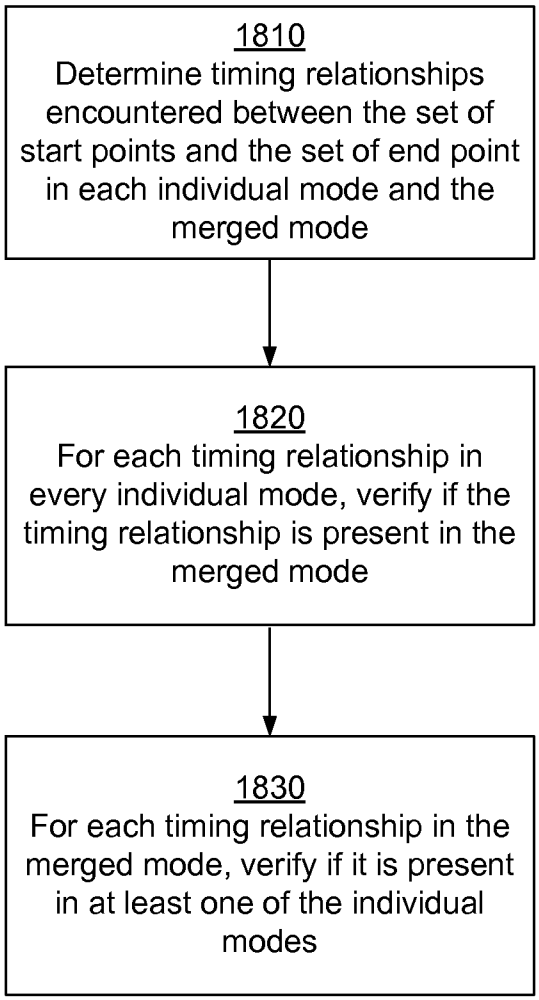

1810
Determine timing relationships encountered between the set of start points and the set of end point in each individual mode and the merged mode
1820
For each timing relationship in every individual mode, verify if the timing relationship is present in the merged mode
1830
For each timing relationship in the merged mode, verify if it is present in at least one of the individual modes

FIG. 18

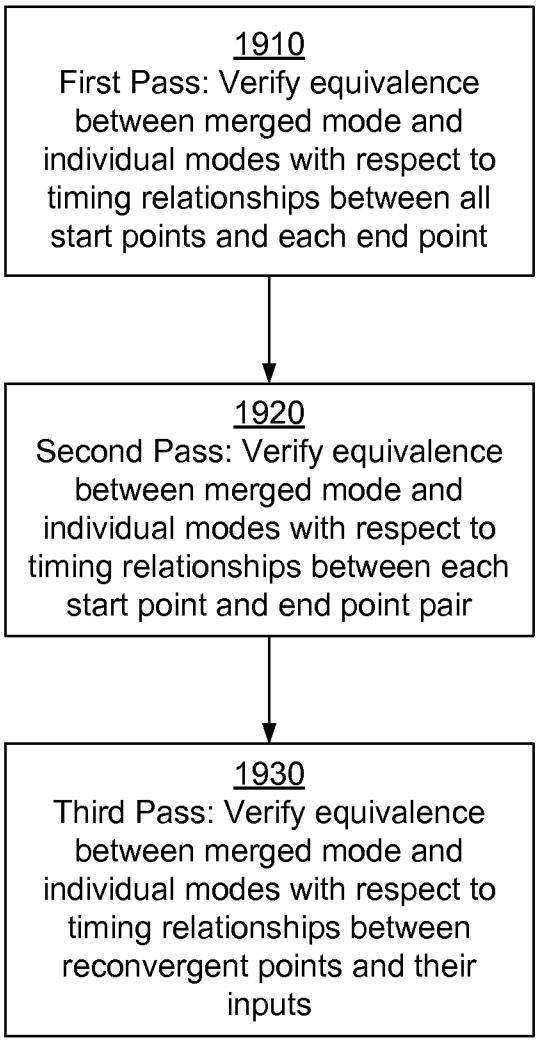

1910
First Pass: Verify equivalence between merged mode and individual modes with respect to timing relationships between all start points and each end point
1920
Second Pass: Verify equivalence between merged mode and individual modes with respect to timing relationships between each start point and end point pair
1930
Third Pass: Verify equivalence between merged mode and individual modes with respect to timing relationships between reconvergent points and their inputs

FIG. 19

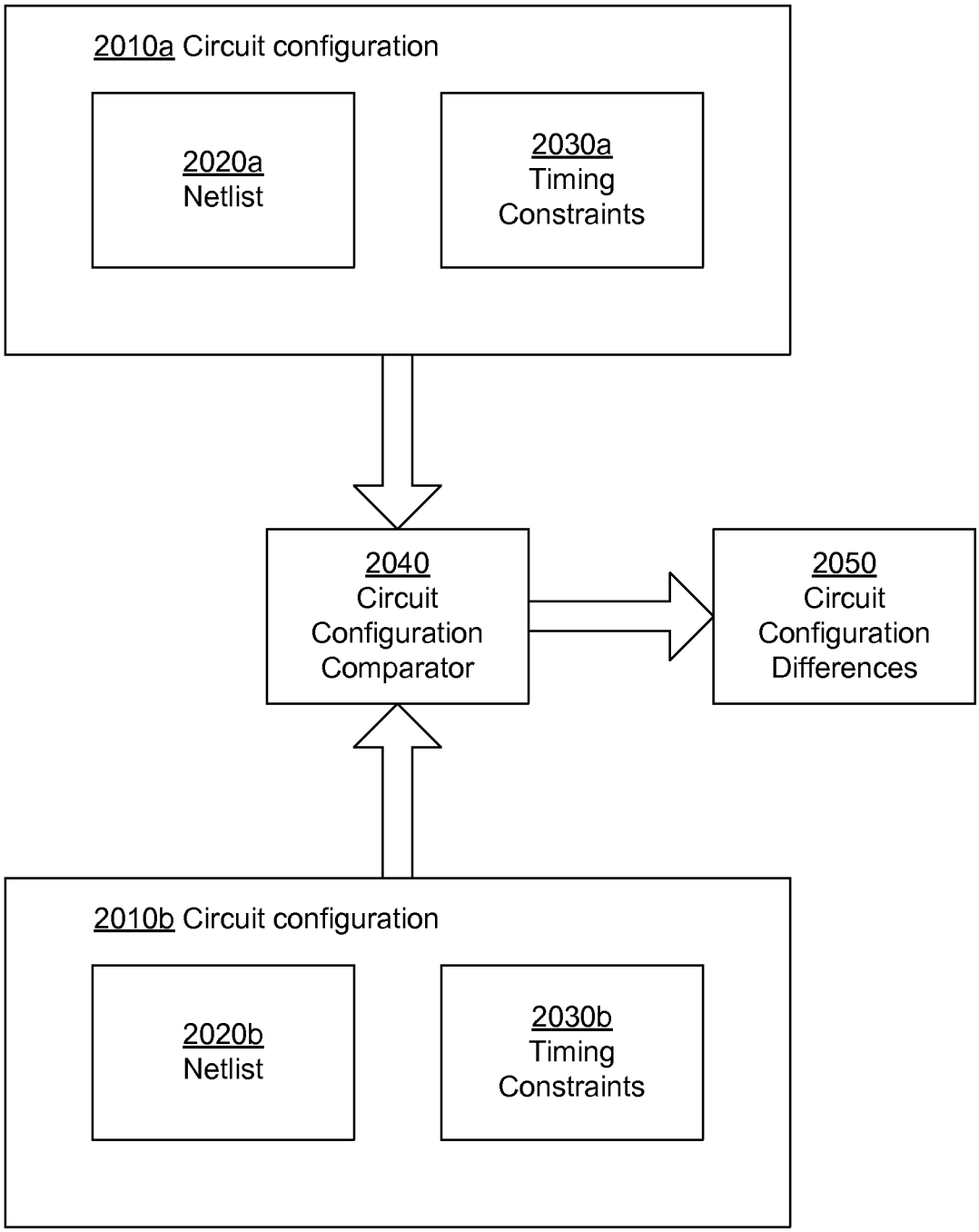

2010a Circuit configuration
2020a Netlist
2030a Timing Constraints
2040 Circuit Configuration Comparator
2050 Circuit Configuration Differences
2010b Circuit configuration
2020b Netlist
2030b Timing Constraints

FIG. 20

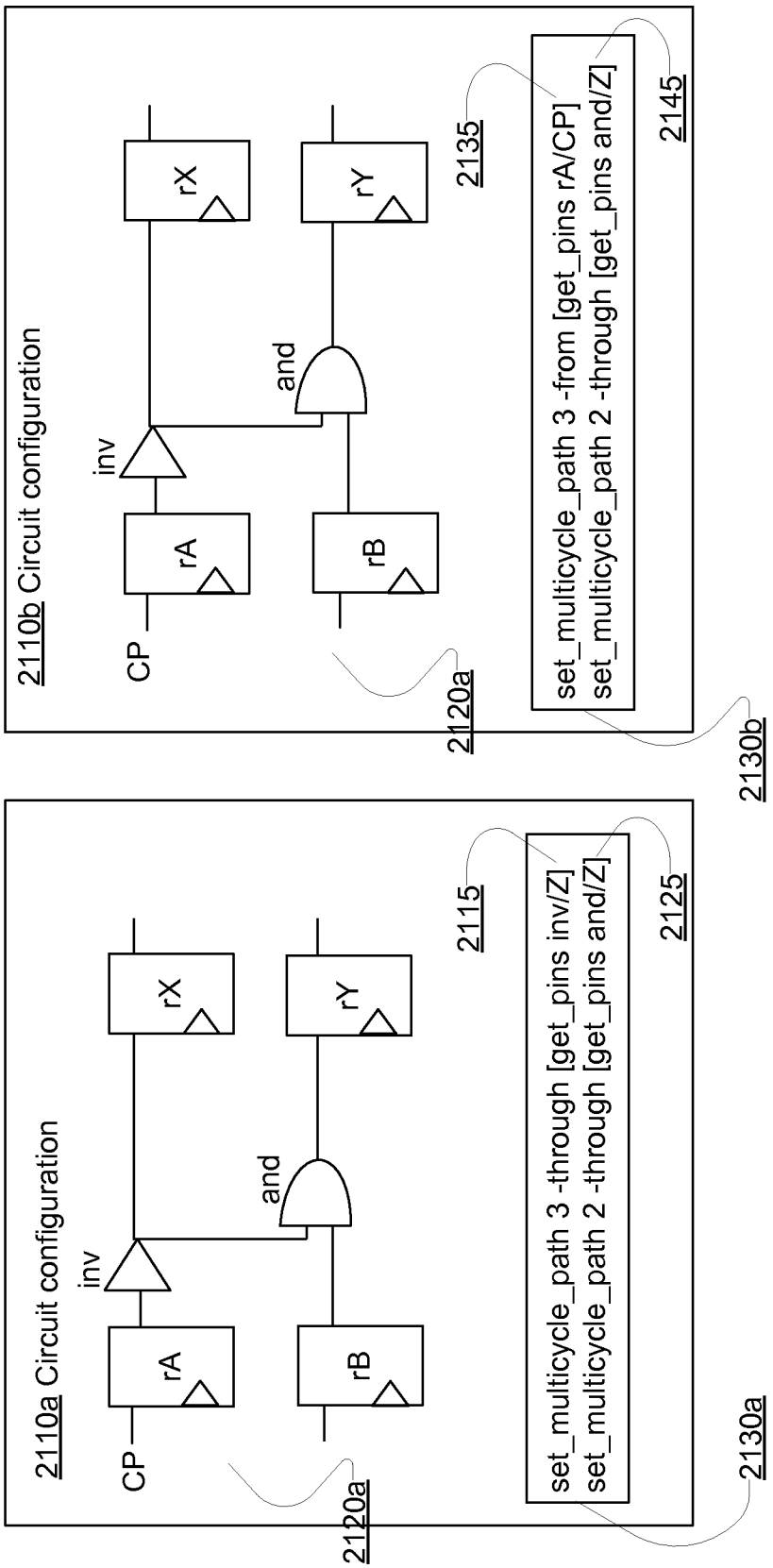

2110a Circuit configuration
CP
rA
rB
inv
and
rX
rY
2120a
2115
set_multicycle_path 3 -through [get_pins inv/Z]
set_multicycle_path 2 -through [get_pins and/Z]
2125
2130a
2110b Circuit configuration
CP
rA
rB
inv
and
rX
rY
2120a
2135
set_multicycle_path 3 -from [get_pins rA/CP]
set_multicycle_path 2 -through [get_pins and/Z]
2145
2130b

FIG. 21

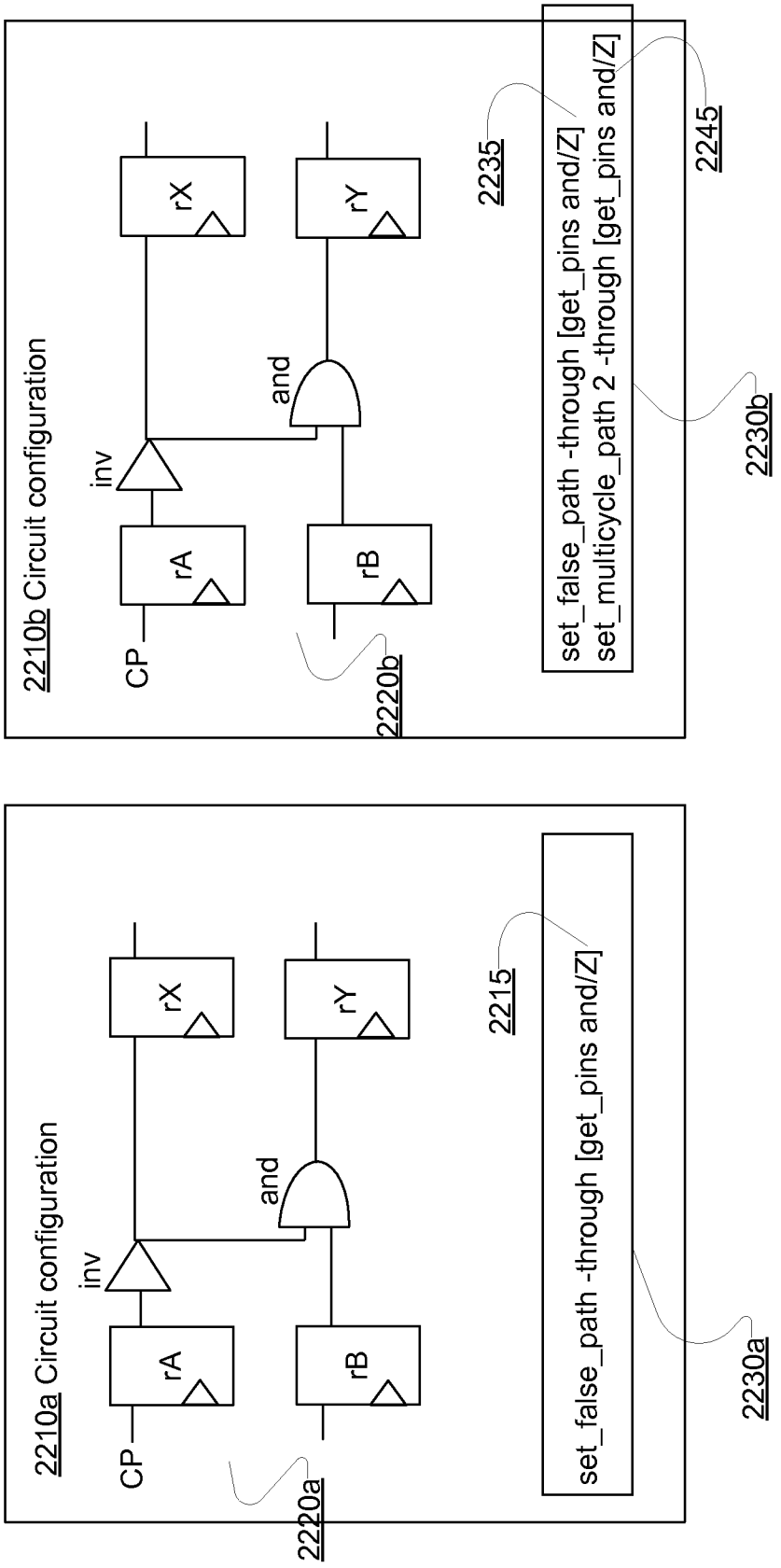

2210b Circuit configuration
rX
rY
inv
and
rA
rB
CP
2220b
2235
2245
set_false_path -through [get_pins and/Z]
set_multicycle_path 2 -through [get_pins and/Z]
2230b
2210a Circuit configuration
rX
rY
inv
and
rA
rB
CP
2220a
2215
set_false_path -through [get_pins and/Z]
2230a

FIG. 22

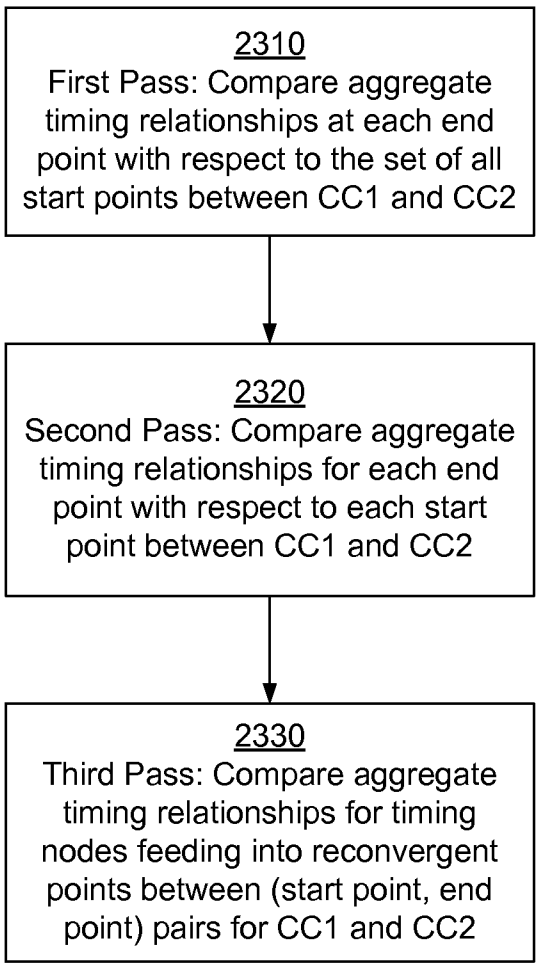

2310
First Pass: Compare aggregate timing relationships at each end point with respect to the set of all start points between CC1 and CC2
2320
Second Pass: Compare aggregate timing relationships for each end point with respect to each start point between CC1 and CC2
2330
Third Pass: Compare aggregate timing relationships for timing nodes feeding into reconvergent points between (start point, end point) pairs for CC1 and CC2

FIG. 23

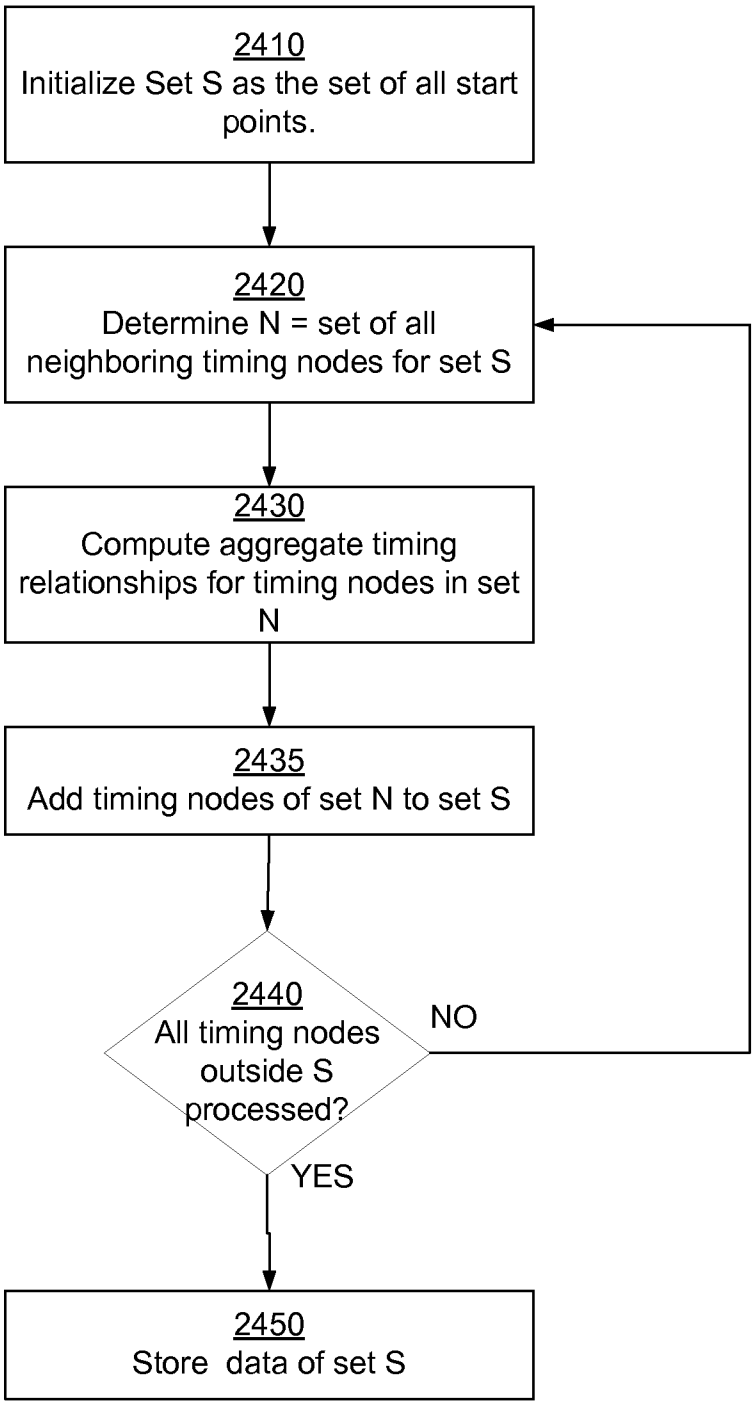

2410
Initialize Set S as the set of all start points.
2420
Determine N = set of all neighboring timing nodes for set S
2430
Compute aggregate timing relationships for timing nodes in set N
2435
Add timing nodes of set N to set S
2440
All timing nodes outside S processed?
NO
YES
2450
Store data of set S

FIG. 24

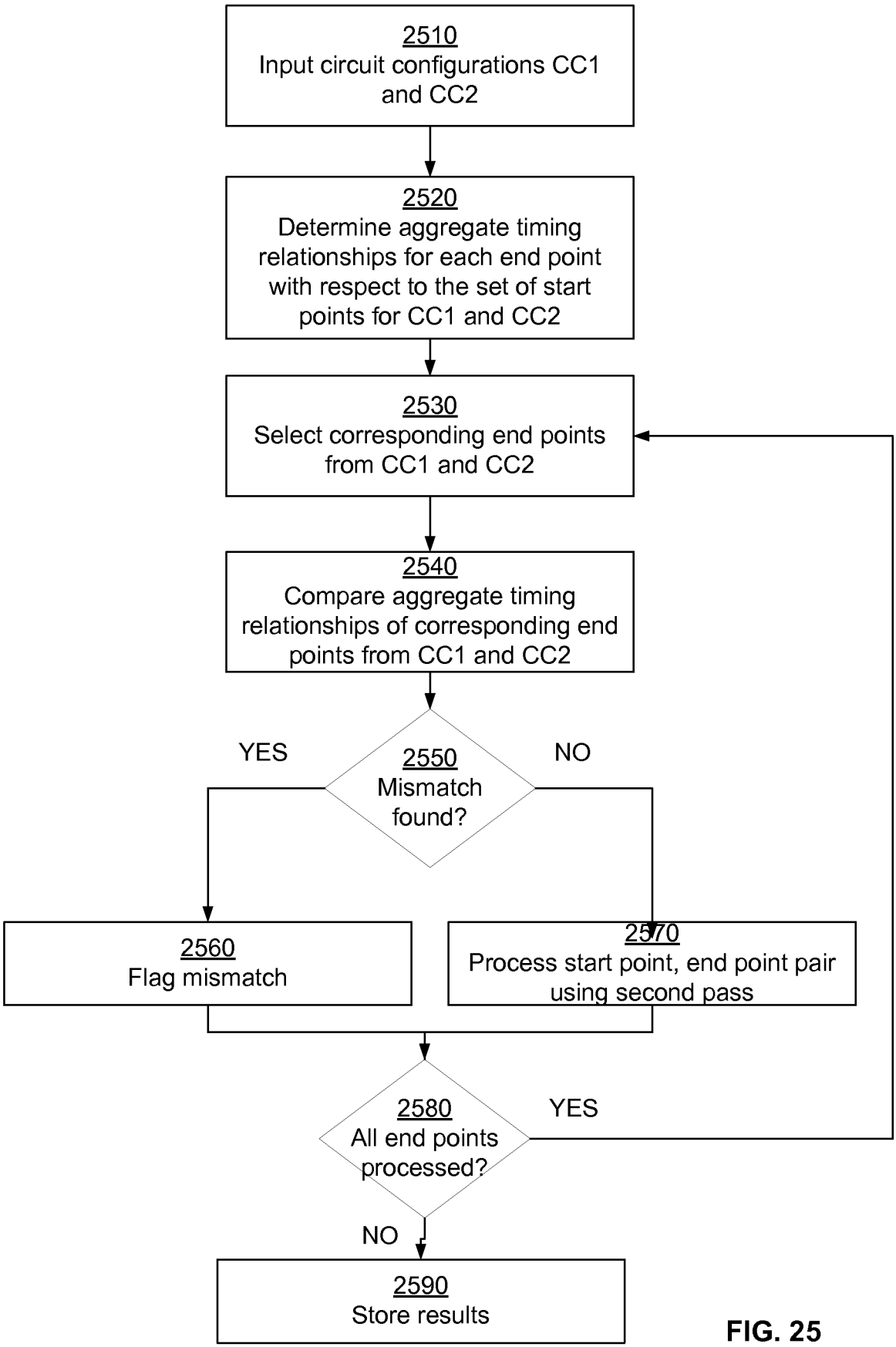

2510
Input circuit configurations CC1 and CC2
2520
Determine aggregate timing relationships for each end point with respect to the set of start points for CC1 and CC2
2530
Select corresponding end points from CC1 and CC2
2540
Compare aggregate timing relationships of corresponding end points from CC1 and CC2
2550
Mismatch found?
YES
NO
2560
Flag mismatch
2570
Process start point, end point pair using second pass
2580
All end points processed?
YES
NO
2590
Store results

FIG. 25

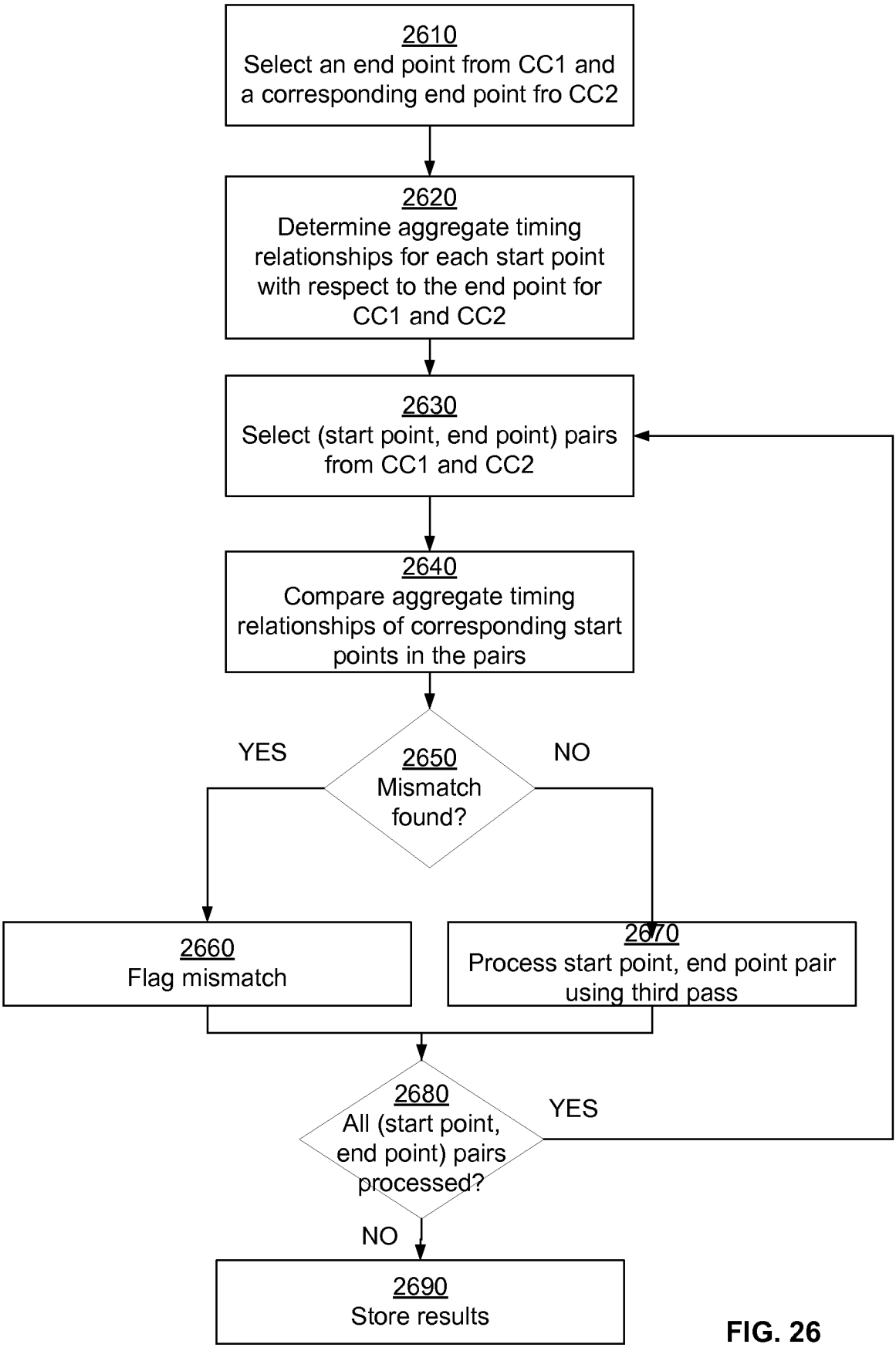

2610
Select an end point from CC1 and a corresponding end point fro CC2
2620
Determine aggregate timing relationships for each start point with respect to the end point for CC1 and CC2
2630
Select (start point, end point) pairs from CC1 and CC2
2640
Compare aggregate timing relationships of corresponding start points in the pairs
2650
Mismatch found?
YES
NO
2660
Flag mismatch
2670
Process start point, end point pair using third pass
2680
All (start point, end point) pairs processed?
YES
NO
2690
Store results

FIG. 26

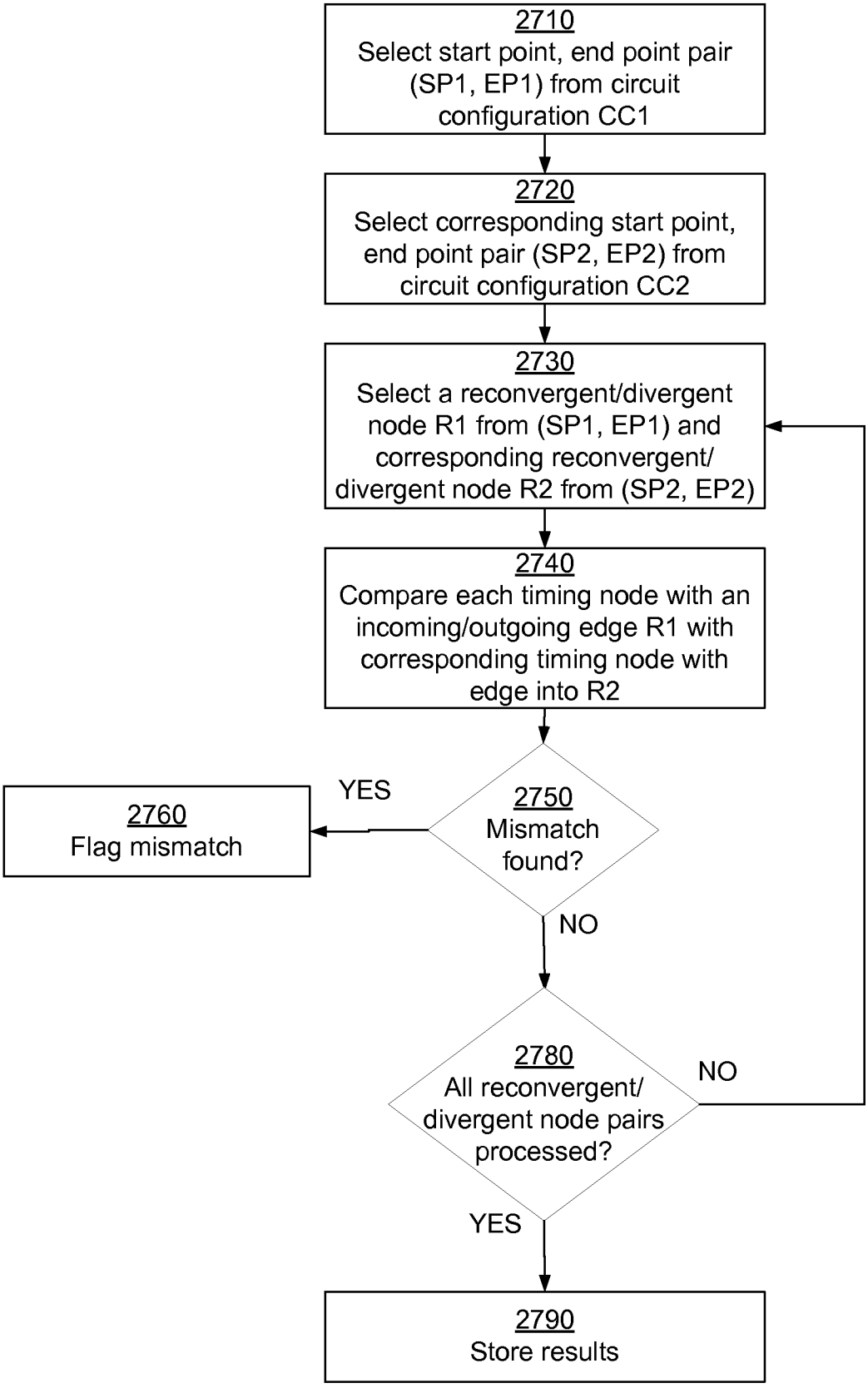

2710
Select start point, end point pair (SP1, EP1) from circuit configuration CC1
2720
Select corresponding start point, end point pair (SP2, EP2) from circuit configuration CC2
2730
Select a reconvergent/divergent node R1 from (SP1, EP1) and corresponding reconvergent/ divergent node R2 from (SP2, EP2)
2740
Compare each timing node with an incoming/outgoing edge R1 with corresponding timing node with edge into R2
2750
Mismatch found?
YES
2760
Flag mismatch
NO
2780
All reconvergent/ divergent node pairs processed?
NO
YES
2790
Store results

FIG. 27

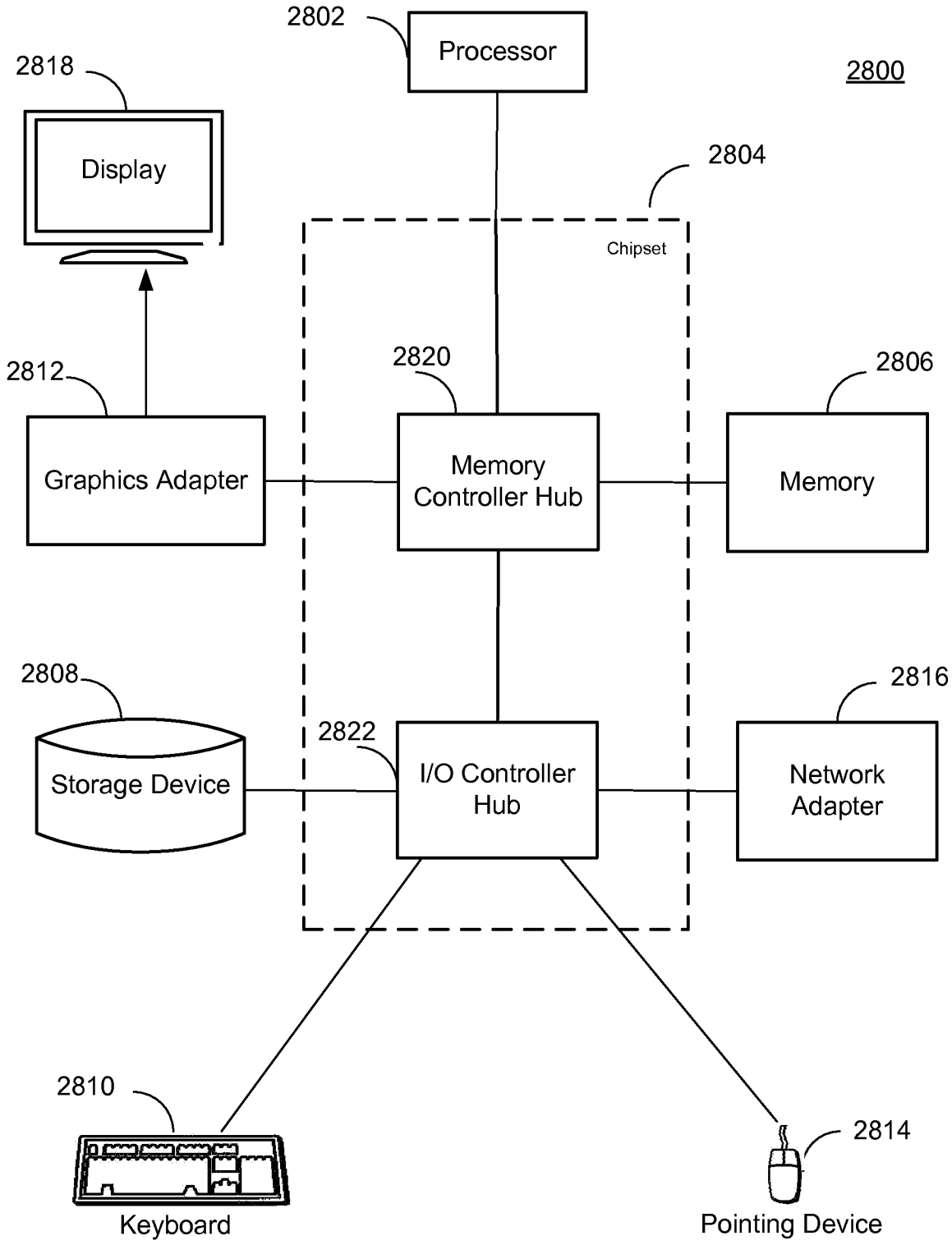

2802
Processor
2800
2818
Display
2804
Chipset
2812
2820
2806
Graphics Adapter
Memory Controller Hub
Memory
2808
2816
2822
Storage Device
I/O Controller Hub
Network Adapter
2810
2814
Keyboard
Pointing Device

FIG. 28