US 20250258986A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2025/0258986 A1**

HEMMETT et al. (43) **Pub. Date: Aug. 14, 2025**

(54) **AUTOMATED SIGNAL ELECTROMIGRATION ANALYSIS FOR LOGIC CIRCUIT DESIGN**

(71) Applicant: **INTERNATIONAL BUSINESS MACHINES CORPORATION**, Armonk, NY (US)

(72) Inventors: **Jeffrey HEMMETT**, St. George, VT (US); **Vasant RAO**, Fishkill, NY (US); **Xin ZHAO**, Hopewell Junction, NY (US); **Ravi Chander LEDALLA**, Fishkill, NY (US)

(21) Appl. No.: **18/438,760**

(22) Filed: **Feb. 12, 2024**

Publication Classification

(51) **Int. Cl.**

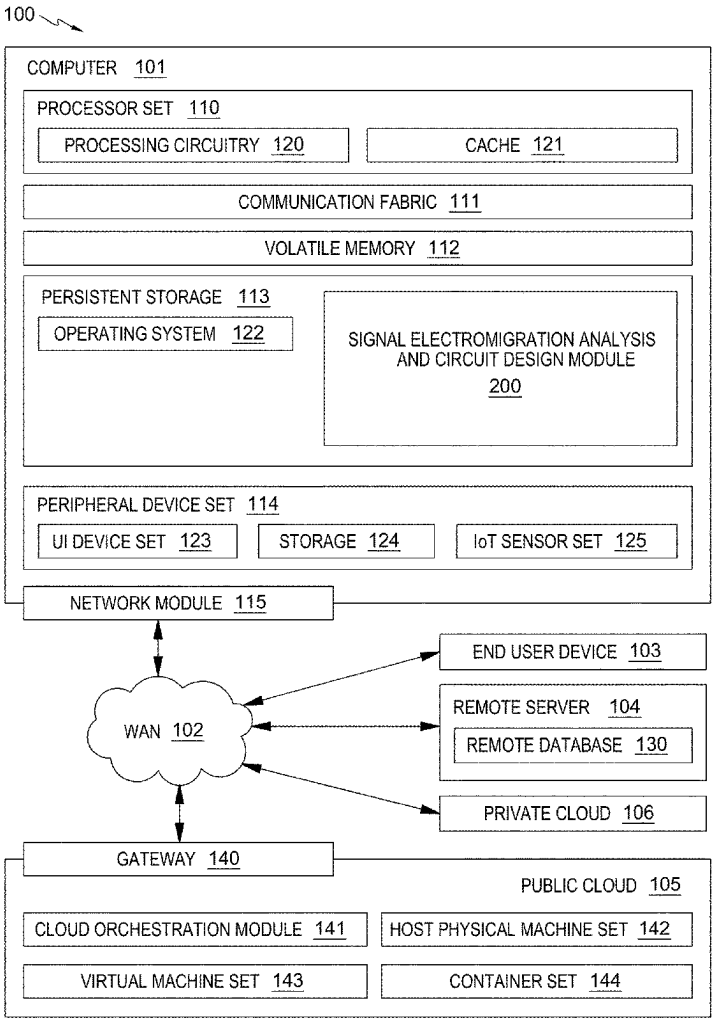
G06F 30/3308	(2020.01)
G06F 30/337	(2020.01)
G06F 119/02	(2020.01)

(52) **U.S. Cl.**

CPC ***G06F 30/3308*** (2020.01); ***G06F 30/337*** (2020.01); *G06F 2119/02* (2020.01)

(57) **ABSTRACT**

Signal electromigration analysis is performed on a logic circuit design by producing a state-analysis-based transition matrix using state analysis data for the logic circuit design, and defining transition submatrices by partitioning the transition matrix. In addition, performing signal electromigration analysis includes determining per simulation weighting factors for the signal electromigration analysis using the transition submatrices and switching factor data, and applying the determined per simulation weighting factors to weight, for signal electromigration analysis, electrical current data collected from a plurality of timing simulations of a transient simulator for the logic circuit design. In addition, performing the signal electromigration analysis includes using the weighted electrical current data in assessing signal electromigration impact on circuit performance of the logic circuit design, and based on the assessing of the circuit performance, the process further includes facilitating generating a modified logic circuit design through modifying of the logic circuit design.



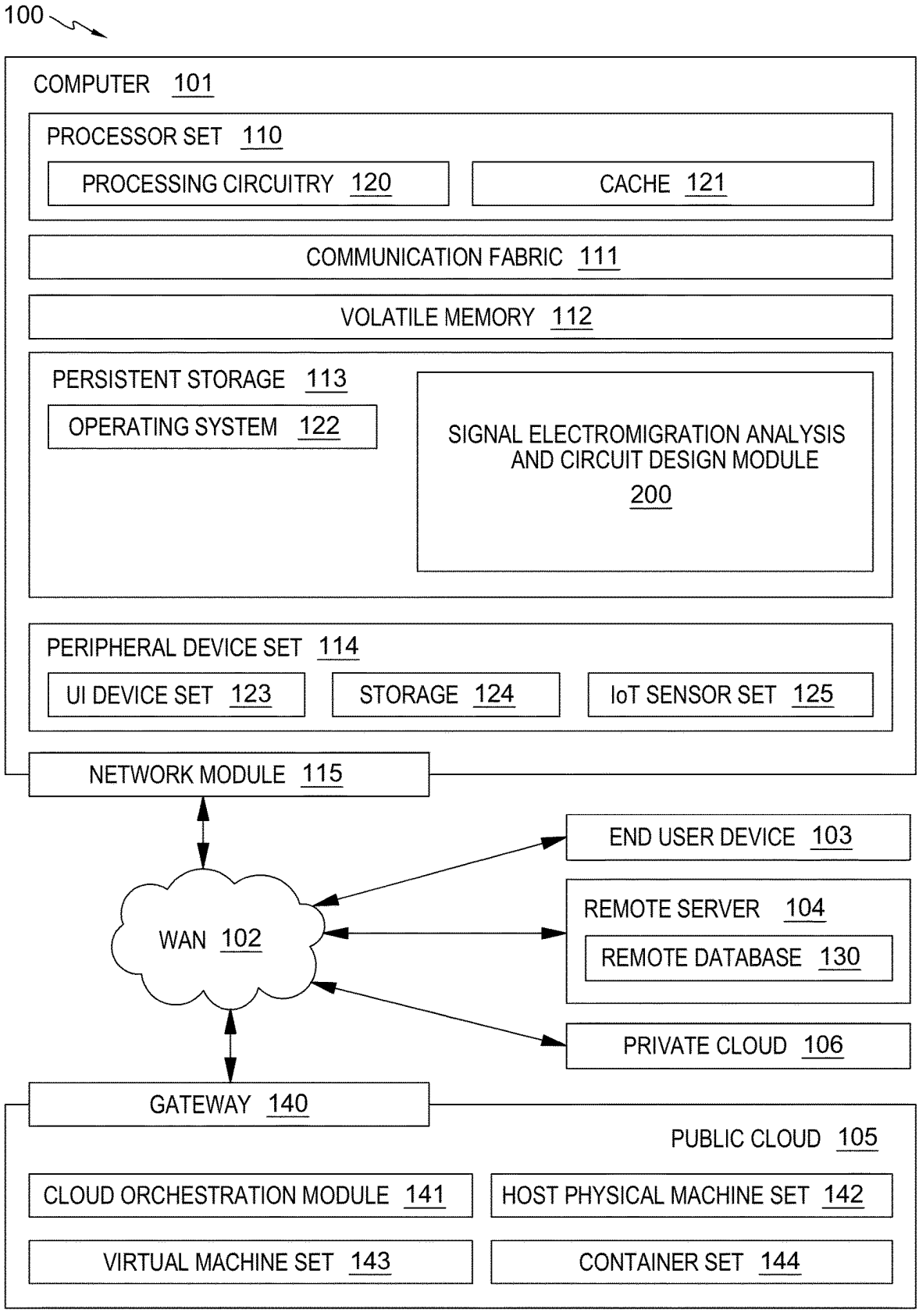

100
COMPUTER 101
PROCESSOR SET 110
PROCESSING CIRCUITRY 120
CACHE 121
COMMUNICATION FABRIC 111
VOLATILE MEMORY 112
PERSISTENT STORAGE 113
OPERATING SYSTEM 122
SIGNAL ELECTROMIGRATION ANALYSIS AND CIRCUIT DESIGN MODULE 200
PERIPHERAL DEVICE SET 114
UI DEVICE SET 123
STORAGE 124
IoT SENSOR SET 125
NETWORK MODULE 115
WAN 102
END USER DEVICE 103
REMOTE SERVER 104
REMOTE DATABASE 130
PRIVATE CLOUD 106
GATEWAY 140
PUBLIC CLOUD 105
CLOUD ORCHESTRATION MODULE 141
HOST PHYSICAL MACHINE SET 142
VIRTUAL MACHINE SET 143
CONTAINER SET 144

FIG. 1

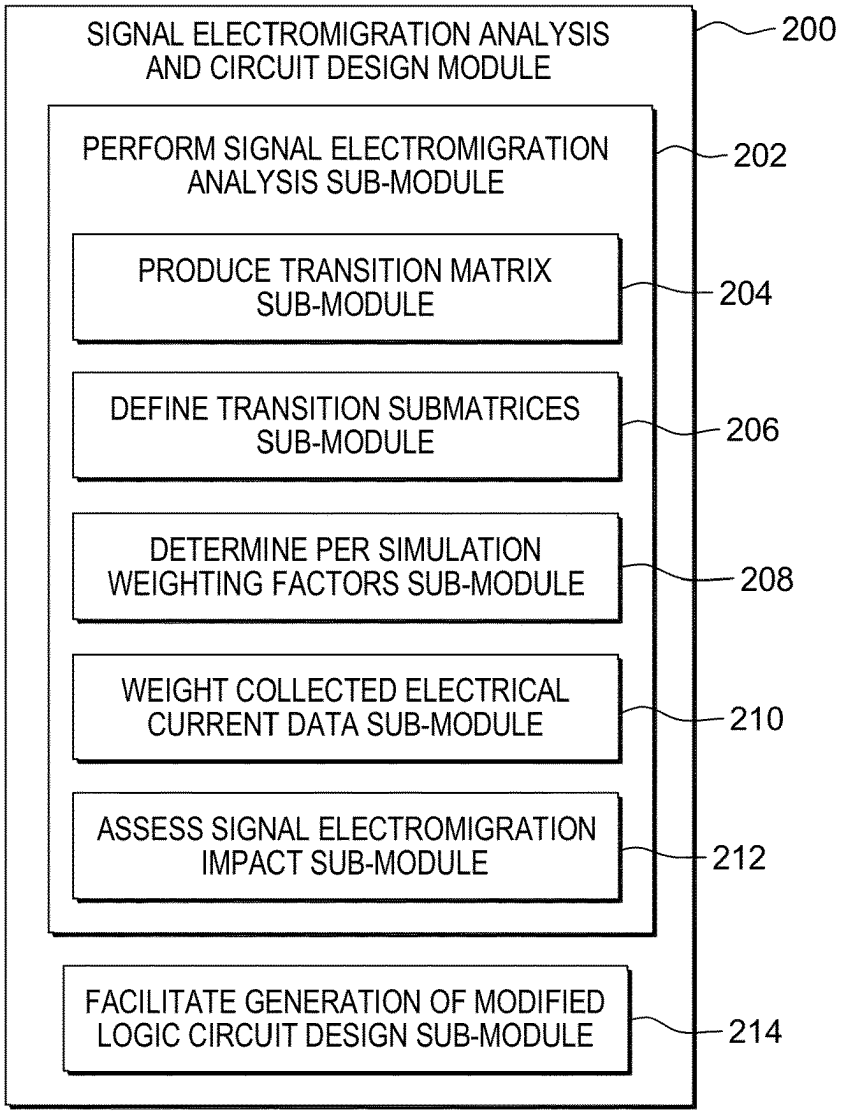

SIGNAL ELECTROMIGRATION ANALYSIS AND CIRCUIT DESIGN MODULE
200
PERFORM SIGNAL ELECTROMIGRATION ANALYSIS SUB-MODULE
202
PRODUCE TRANSITION MATRIX SUB-MODULE
204
DEFINE TRANSITION SUBMATRICES SUB-MODULE
206
DETERMINE PER SIMULATION WEIGHTING FACTORS SUB-MODULE
208
WEIGHT COLLECTED ELECTRICAL CURRENT DATA SUB-MODULE
210
ASSESS SIGNAL ELECTROMIGRATION IMPACT SUB-MODULE
212
FACILITATE GENERATION OF MODIFIED LOGIC CIRCUIT DESIGN SUB-MODULE
214

FIG. 2A

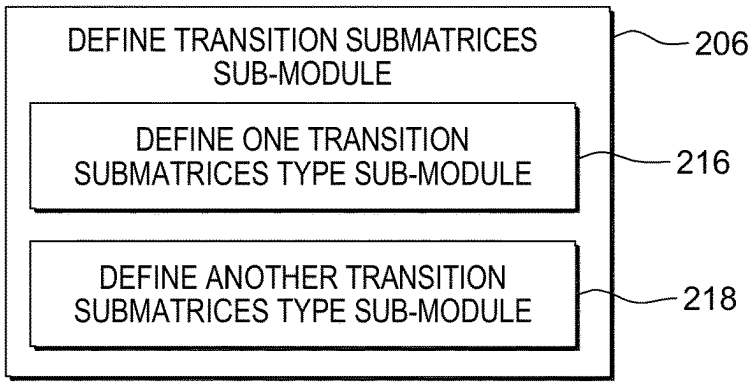

DEFINE TRANSITION SUBMATRICES SUB-MODULE
206
DEFINE ONE TRANSITION SUBMATRICES TYPE SUB-MODULE
216
DEFINE ANOTHER TRANSITION SUBMATRICES TYPE SUB-MODULE
218

FIG. 2B

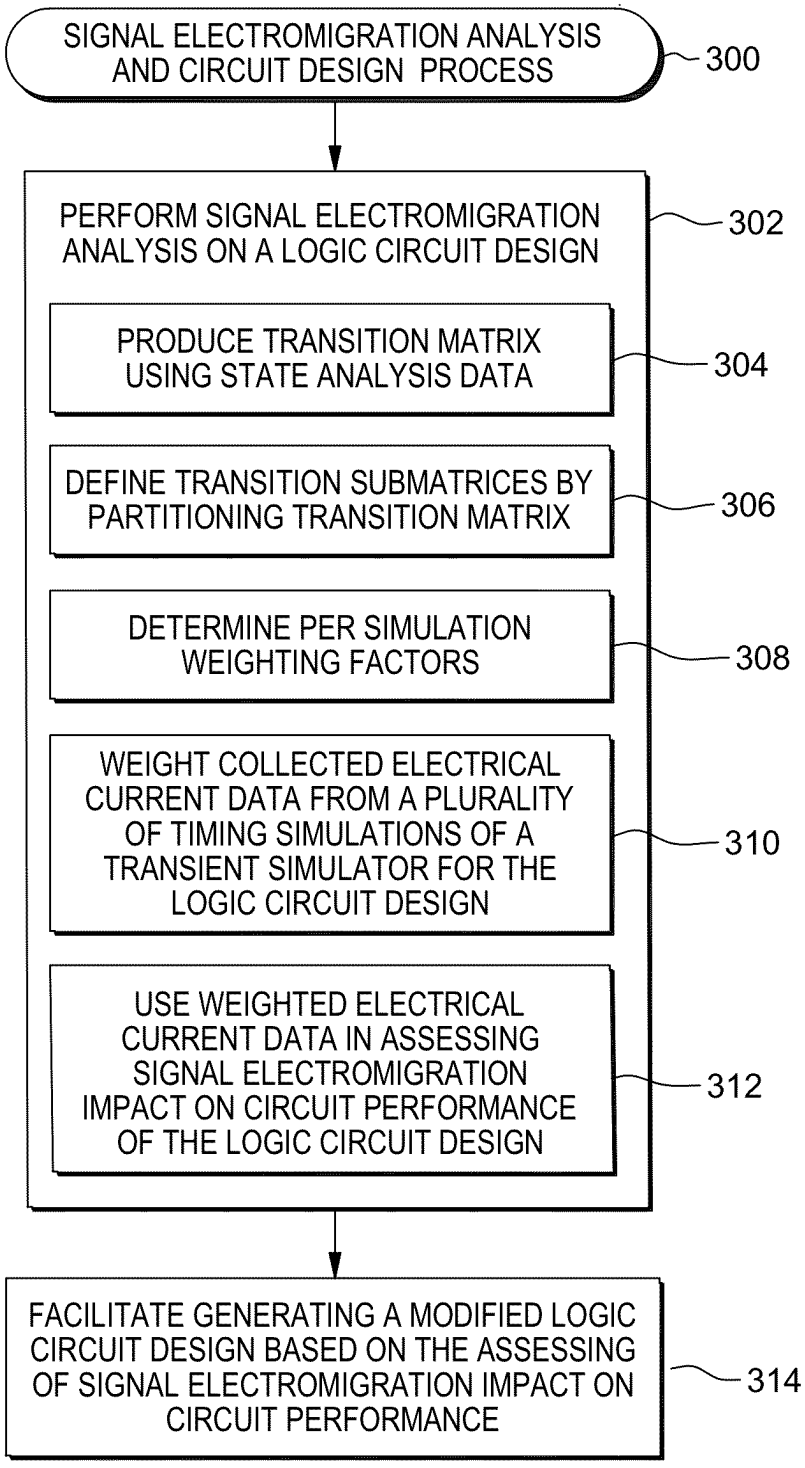

SIGNAL ELECTROMIGRATION ANALYSIS AND CIRCUIT DESIGN PROCESS
300
PERFORM SIGNAL ELECTROMIGRATION ANALYSIS ON A LOGIC CIRCUIT DESIGN
302
PRODUCE TRANSITION MATRIX USING STATE ANALYSIS DATA
304
DEFINE TRANSITION SUBMATRICES BY PARTITIONING TRANSITION MATRIX
306
DETERMINE PER SIMULATION WEIGHTING FACTORS
308
WEIGHT COLLECTED ELECTRICAL CURRENT DATA FROM A PLURALITY OF TIMING SIMULATIONS OF A TRANSIENT SIMULATOR FOR THE LOGIC CIRCUIT DESIGN
310
USE WEIGHTED ELECTRICAL CURRENT DATA IN ASSESSING SIGNAL ELECTROMIGRATION IMPACT ON CIRCUIT PERFORMANCE OF THE LOGIC CIRCUIT DESIGN
312
FACILITATE GENERATING A MODIFIED LOGIC CIRCUIT DESIGN BASED ON THE ASSESSING OF SIGNAL ELECTROMIGRATION IMPACT ON CIRCUIT PERFORMANCE
314

FIG. 3

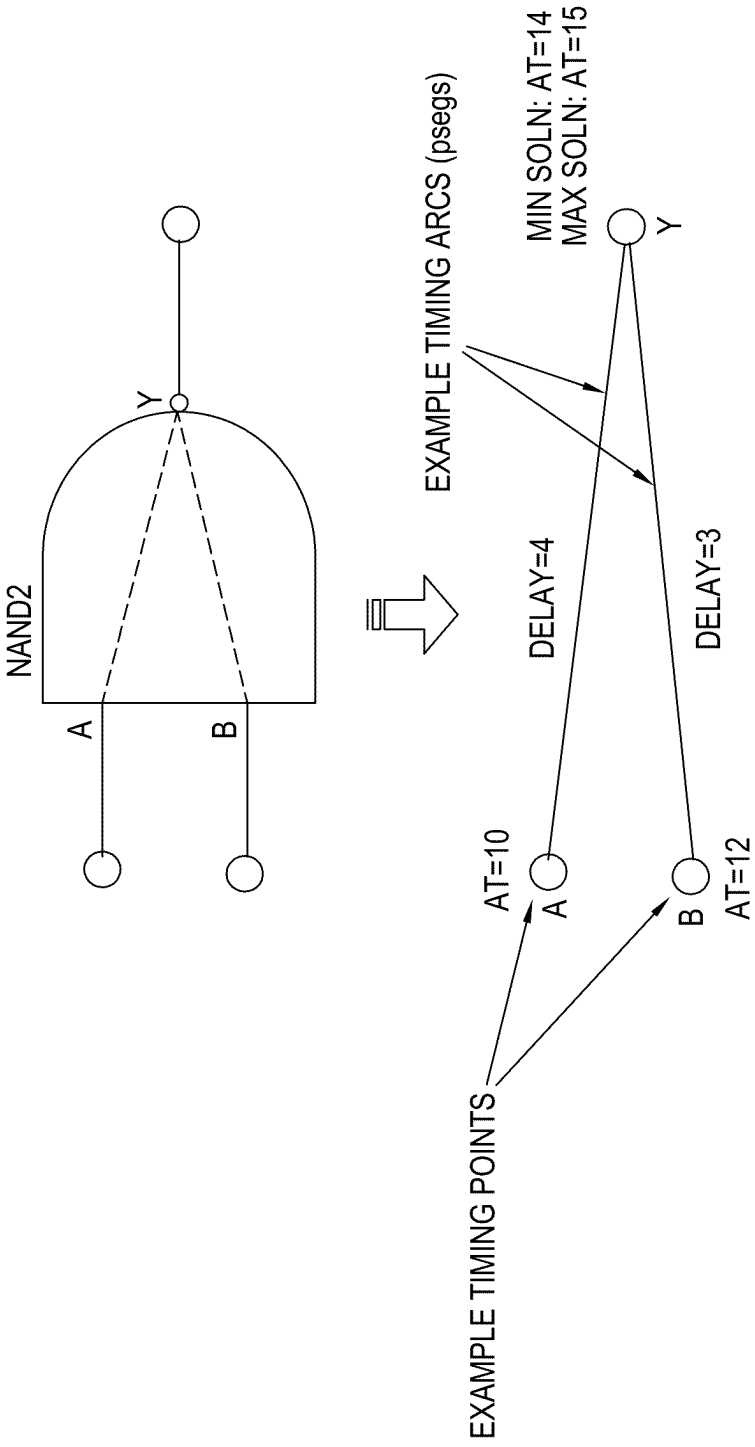

NAND2
A
B
Y
EXAMPLE TIMING ARCS (psegs)
MIN SOLN: AT=14
MAX SOLN: AT=15
DELAY=4
DELAY=3
AT=10
AT=12
EXAMPLE TIMING POINTS

FIG. 4

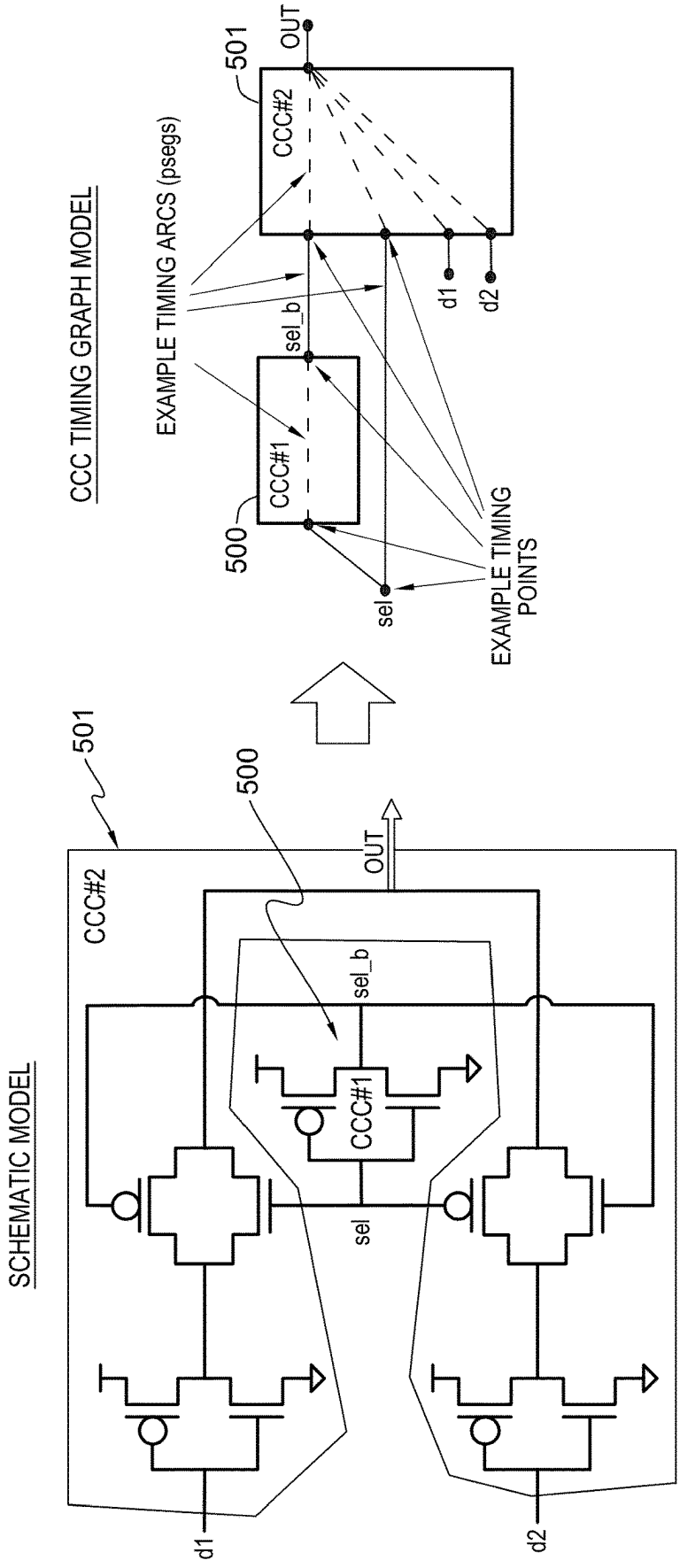

CCC TIMING GRAPH MODEL
EXAMPLE TIMING ARCS (psegs)
501
OUT
CCC#2
d1
d2
sel_b
CCC#1
500
sel
EXAMPLE TIMING POINTS
SCHEMATIC MODEL
501
500
OUT
CCC#2
sel_b
CCC#1
sel
d1
d2

FIG. 5

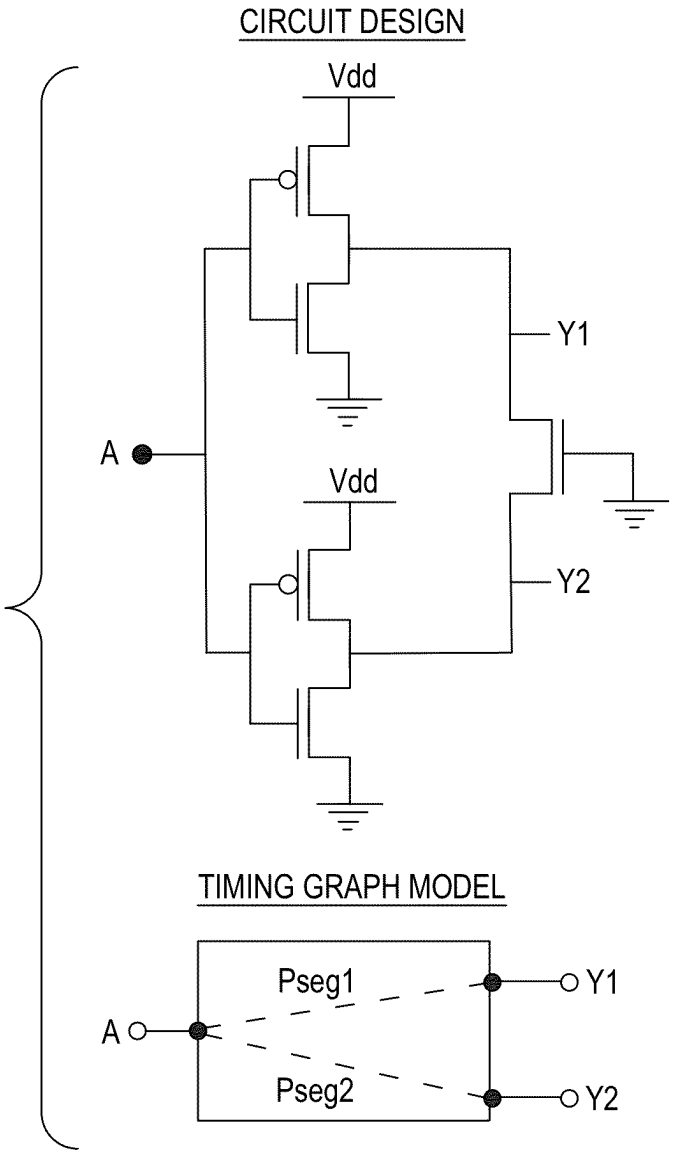

CIRCUIT DESIGN
Vdd
Y1
A
Vdd
Y2
TIMING GRAPH MODEL
Pseg1
Y1
A
Pseg2
Y2

FIG. 6A

	SIM #1	SIM #2	SIM #3	SIM #4	SIM #5	SIM #6	SIM #7	SIM #8
	EARLY	EARLY	LATE	LATE	EARLY	EARLY	LATE	LATE
A:R	1	0	1	0	1	0	1	0
A:F	0	1	0	1	0	1	0	1
Y1:R	0	1	0	1	0	0	0	0
Y1:F	1	0	1	0	0	0	0	0
Y2:R	0	0	0	0	0	1	0	1
Y2:F	0	0	0	0	1	0	1	0

FIG. 6B

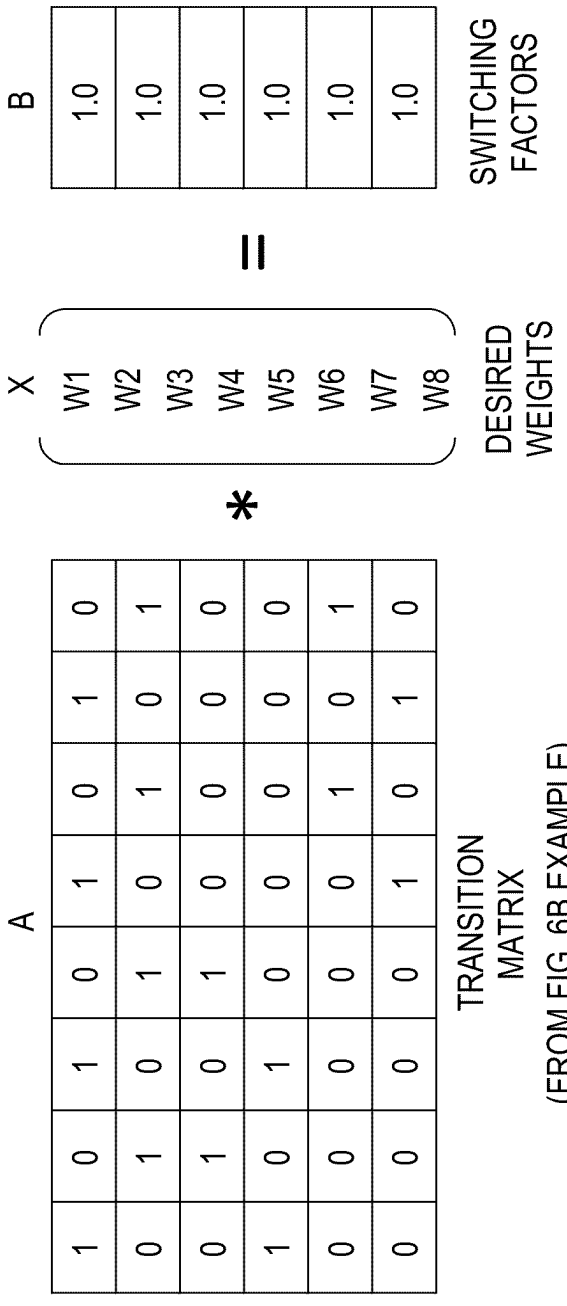

A
1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1
0 1 0 1 0 0 0 0
1 0 1 0 0 0 0 0
0 0 0 0 0 1 0 1
0 0 0 0 1 0 1 0
TRANSITION MATRIX
(FROM FIG. 6B EXAMPLE)
*
X
W1
W2
W3
W4
W5
W6
W7
W8
DESIRED WEIGHTS
=
B
1.0
1.0
1.0
1.0
1.0
1.0
SWITCHING FACTORS

FIG. 7

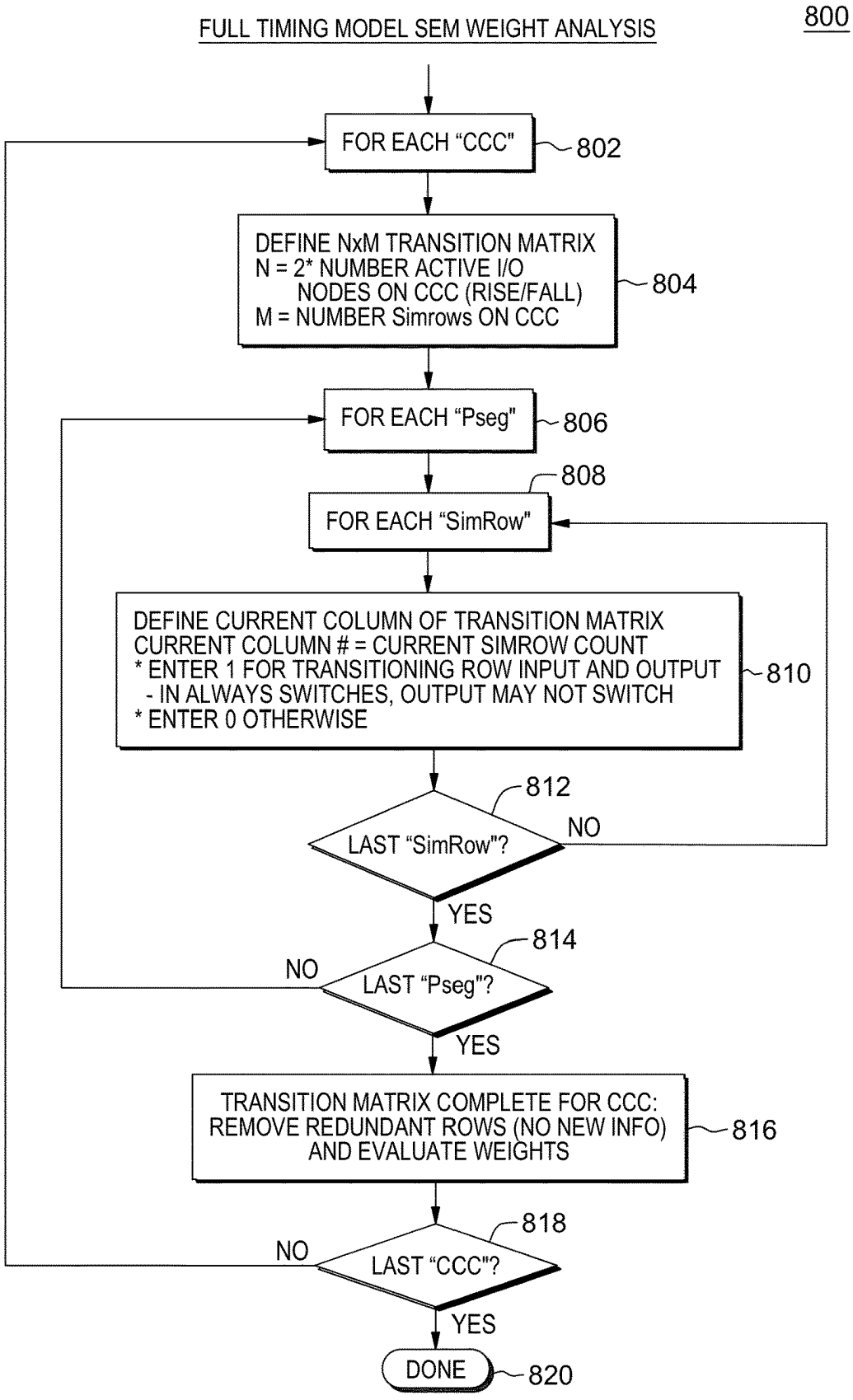

FULL TIMING MODEL SEM WEIGHT ANALYSIS
800
FOR EACH "CCC"
802
DEFINE NxM TRANSITION MATRIX
N = 2* NUMBER ACTIVE I/O
NODES ON CCC (RISE/FALL)
M = NUMBER Simrows ON CCC
804
FOR EACH "Pseg"
806
808
FOR EACH "SimRow"
DEFINE CURRENT COLUMN OF TRANSITION MATRIX
CURRENT COLUMN # = CURRENT SIMROW COUNT
* ENTER 1 FOR TRANSITIONING ROW INPUT AND OUTPUT
- IN ALWAYS SWITCHES, OUTPUT MAY NOT SWITCH
* ENTER 0 OTHERWISE
810
812
LAST "SimRow"?
NO
YES
814
NO
LAST "Pseg"?
YES
TRANSITION MATRIX COMPLETE FOR CCC:
REMOVE REDUNDANT ROWS (NO NEW INFO)
AND EVALUATE WEIGHTS
816
818
NO
LAST "CCC"?
YES
DONE
820

FIG. 8

perOutput SUBMATRIX Y1: 4X4

	EARLY	EARLY	LATE	LATE
A:R	1	0	1	0
A:F	0	1	0	1
Y1:R	0	1	0	1
Y1:F	1	0	1	0
	SIM #1	SIM #2	SIM #3	SIM #4

FIG. 9A

perOutput SUBMATRIX Y2: 4X4

	EARLY	EARLY	LATE	LATE
A:R	1	0	1	0
A:F	0	1	0	1
Y2:R	0	1	0	1
Y2:F	1	0	1	0
	SIM #5	SIM #6	SIM #7	SIM #8

FIG. 9B

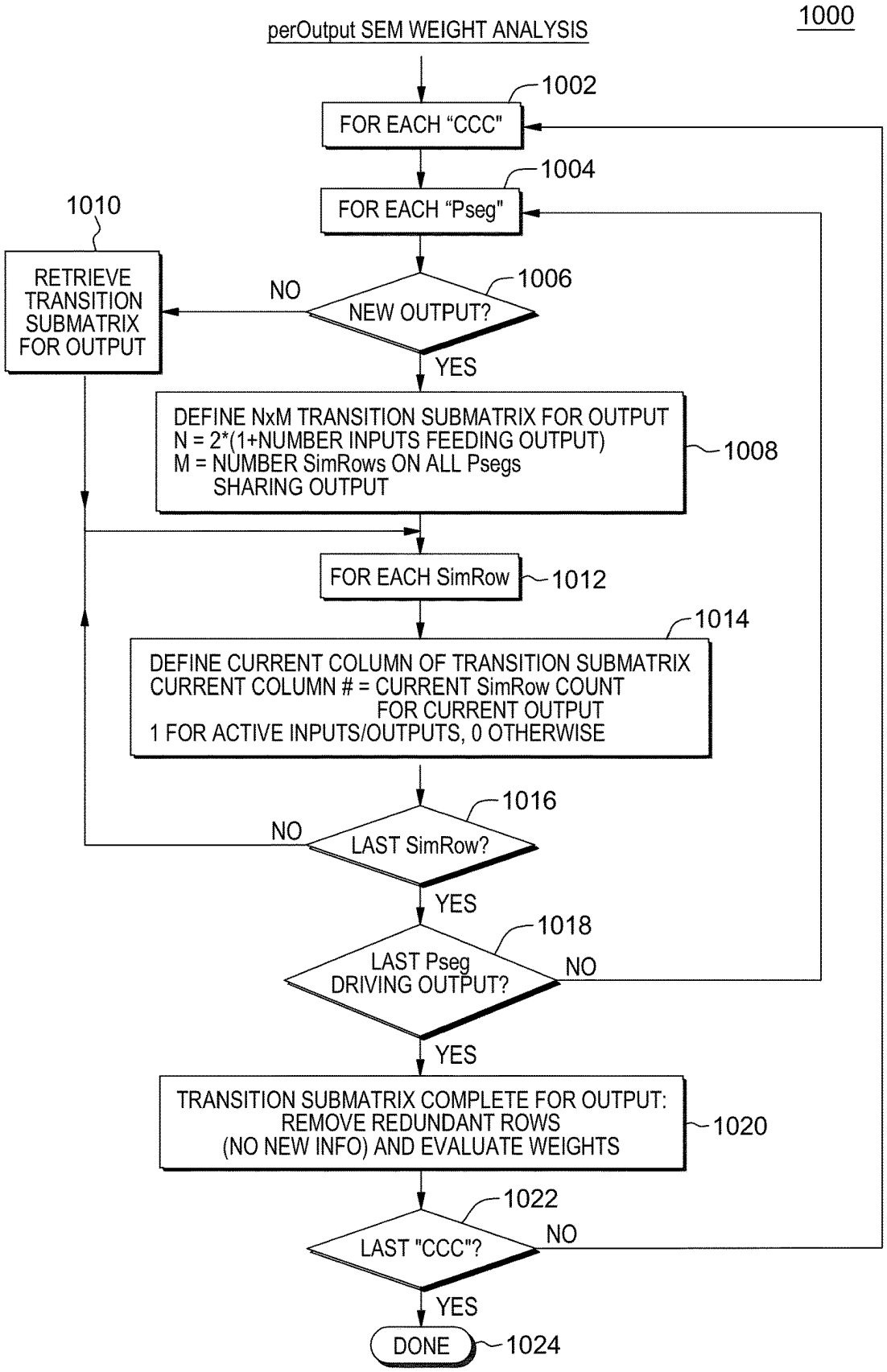

1000
perOutput SEM WEIGHT ANALYSIS
1002
FOR EACH "CCC"
1004
FOR EACH "Pseg"
1010
RETRIEVE TRANSITION SUBMATRIX FOR OUTPUT
NO
1006
NEW OUTPUT?
YES
DEFINE NxM TRANSITION SUBMATRIX FOR OUTPUT
N = 2*(1+NUMBER INPUTS FEEDING OUTPUT)
M = NUMBER SimRows ON ALL Psegs SHARING OUTPUT
1008
FOR EACH SimRow
1012
1014
DEFINE CURRENT COLUMN OF TRANSITION SUBMATRIX
CURRENT COLUMN # = CURRENT SimRow COUNT FOR CURRENT OUTPUT
1 FOR ACTIVE INPUTS/OUTPUTS, 0 OTHERWISE
1016
NO
LAST SimRow?
YES
1018
LAST Pseg DRIVING OUTPUT?
NO
YES
TRANSITION SUBMATRIX COMPLETE FOR OUTPUT: REMOVE REDUNDANT ROWS (NO NEW INFO) AND EVALUATE WEIGHTS
1020
1022
LAST "CCC"?
NO
YES
DONE
1024

FIG. 10

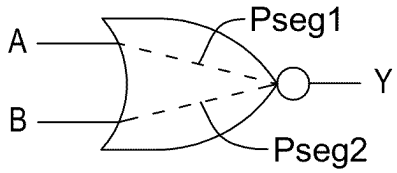


FIG. 11A

	SIM1	SIM2	SIM3	SIM4		WEIGHT		SF
A:R	1	0	1	0	*	W1	=	SfA
A:F	0	1	0	1		W2		SfA
Y:R	0	1	0	0		W3		0.5* SfY
Y:F	1	0	0	0		W4		0.5* SfY

Pseg1

FIG. 11B

	SIM5	SIM6	SIM7	SIM8		WEIGHT		SF
B:R	1	0	1	0	*	W5	=	SfB
B:F	0	1	0	1		W6		SfB
Y:R	0	1	0	0		W7		0.5* SfY
Y:F	1	0	0	0		W8		0.5* SfY

Pseg2

FIG. 11C

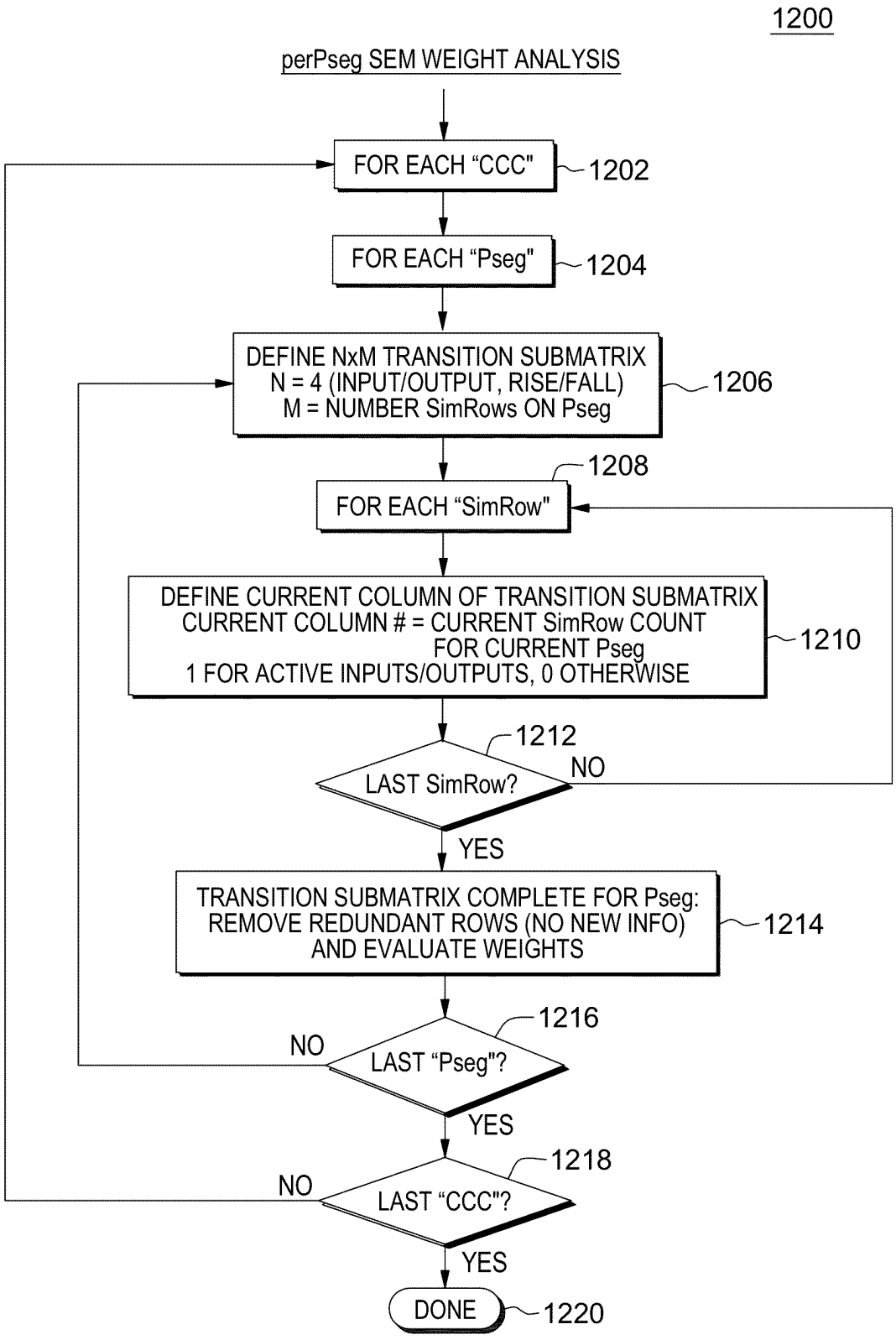

1200
perPseg SEM WEIGHT ANALYSIS
FOR EACH "CCC"
1202
FOR EACH "Pseg"
1204
DEFINE NxM TRANSITION SUBMATRIX
N = 4 (INPUT/OUTPUT, RISE/FALL)
M = NUMBER SimRows ON Pseg
1206
1208
FOR EACH "SimRow"
DEFINE CURRENT COLUMN OF TRANSITION SUBMATRIX
CURRENT COLUMN # = CURRENT SimRow COUNT
FOR CURRENT Pseg
1 FOR ACTIVE INPUTS/OUTPUTS, 0 OTHERWISE
1210
1212
LAST SimRow?
NO
YES
TRANSITION SUBMATRIX COMPLETE FOR Pseg:
REMOVE REDUNDANT ROWS (NO NEW INFO)
AND EVALUATE WEIGHTS
1214
1216
NO
LAST "Pseg"?
YES
1218
NO
LAST "CCC"?
YES
DONE
1220

FIG. 12

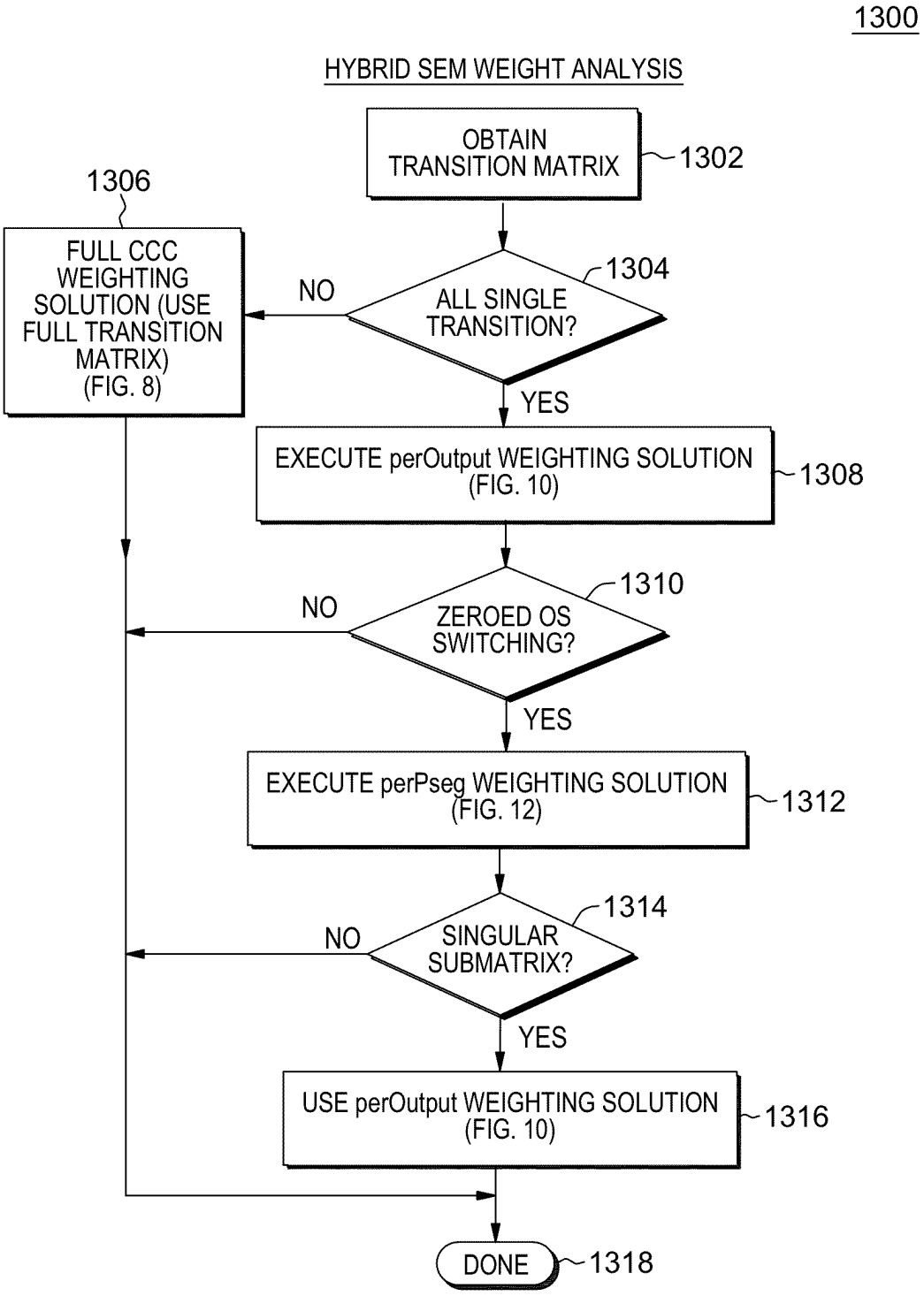

1300
HYBRID SEM WEIGHT ANALYSIS
OBTAIN TRANSITION MATRIX
1302
1306
FULL CCC WEIGHTING SOLUTION (USE FULL TRANSITION MATRIX) (FIG. 8)
NO
ALL SINGLE TRANSITION?
1304
YES
EXECUTE perOutput WEIGHTING SOLUTION (FIG. 10)
1308
NO
ZEROED OS SWITCHING?
1310
YES
EXECUTE perPseg WEIGHTING SOLUTION (FIG. 12)
1312
NO
SINGULAR SUBMATRIX?
1314
YES
USE perOutput WEIGHTING SOLUTION (FIG. 10)
1316
DONE
1318

FIG. 13

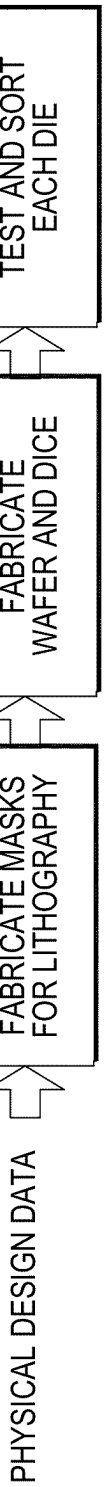

PHYSICAL DESIGN DATA
1410
FABRICATE MASKS FOR LITHOGRAPHY
1420
FABRICATE WAFER AND DICE
1430
TEST AND SORT EACH DIE

FIG. 14

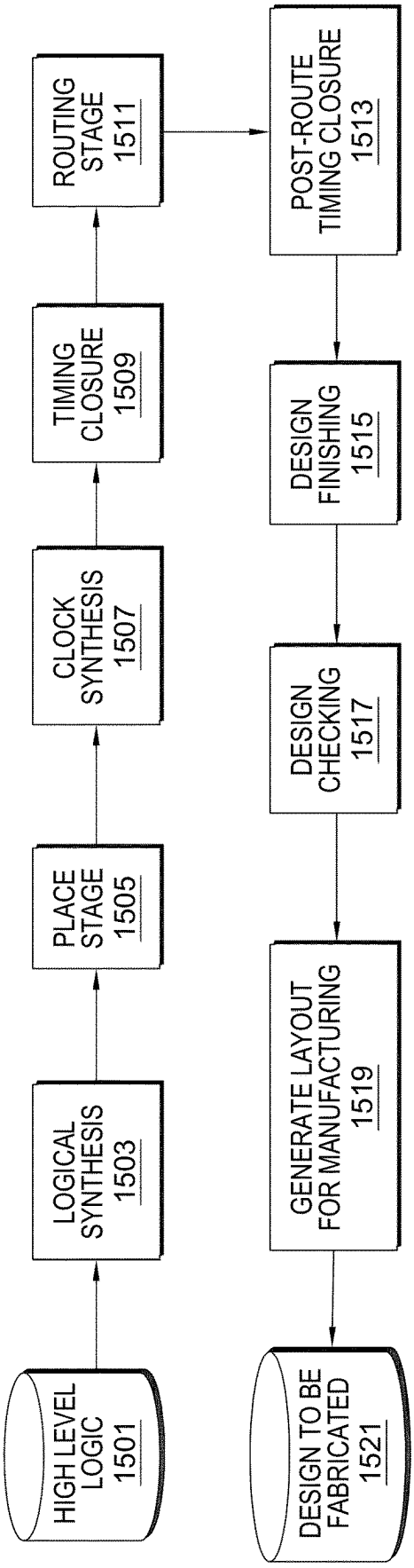

HIGH LEVEL LOGIC 1501
LOGICAL SYNTHESIS 1503
PLACE STAGE 1505
CLOCK SYNTHESIS 1507
TIMING CLOSURE 1509
ROUTING STAGE 1511
POST-ROUTE TIMING CLOSURE 1513
DESIGN FINISHING 1515
DESIGN CHECKING 1517
GENERATE LAYOUT FOR MANUFACTURING 1519
DESIGN TO BE FABRICATED 1521

FIG. 15

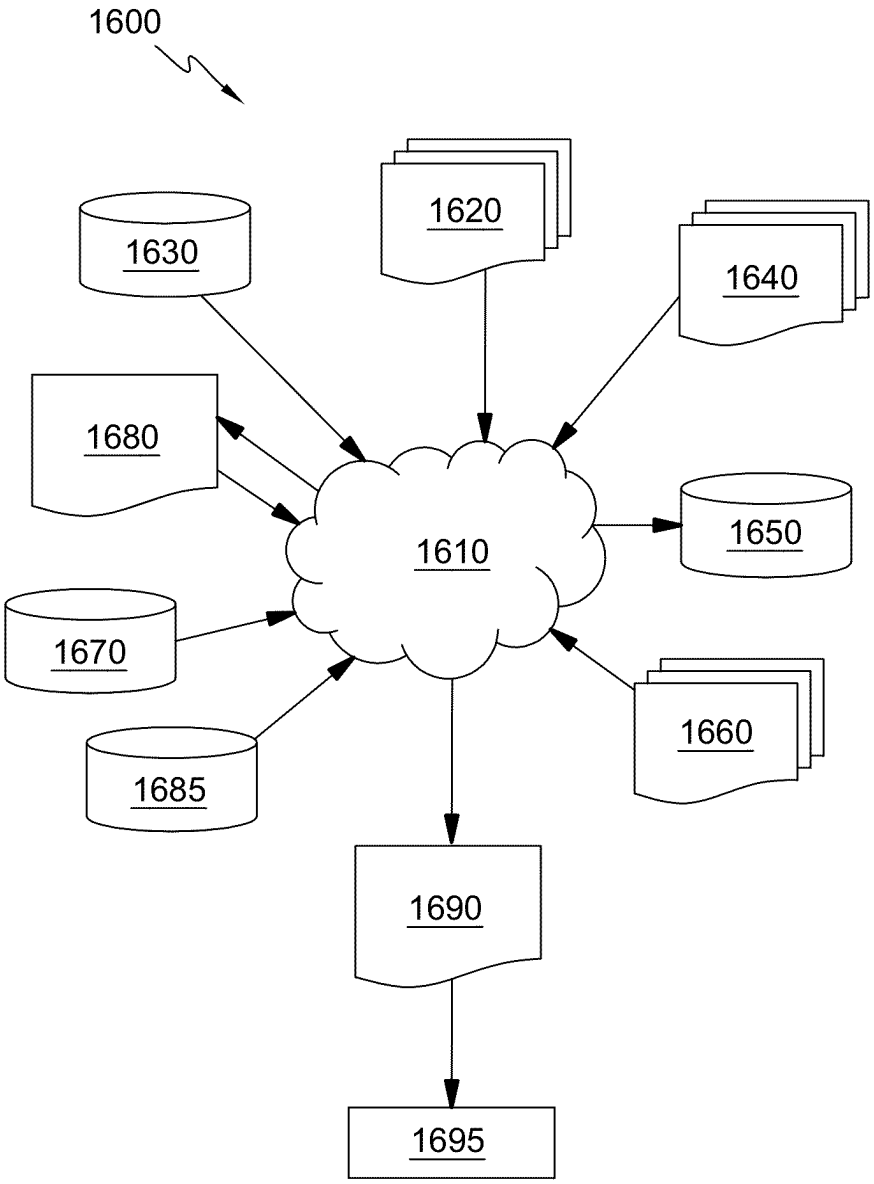


FIG. 16

AUTOMATED SIGNAL ELECTROMIGRATION ANALYSIS FOR LOGIC CIRCUIT DESIGN

BACKGROUND

[0001] The present invention relates to the design and manufacture of integrated circuits (ICs), and specifically, to very large-scale integration (VLSI) designs and devices based on the analysis and optimization of such circuits.

[0002] Signal electromigration (SEM) analysis is one type of analysis of a logic circuit design to facilitate improving or optimizing the circuit design. Electromigration is a type of potential fault mechanism in integrated circuits, caused by the movement of atoms based on the flow of current through material. Electromigration can result in both vacancies and deposits within, for instance, a wire or gate of the integrated circuit. As part of the design and fabrication of integrated circuits, simulation of electromigration is performed in modern technology computer aided design (TCAD) tools.

SUMMARY

[0003] Certain shortcomings of the prior art are overcome, and additional advantages are provided herein through the provision of a computer-implemented method, which includes: performing, by at least one processor set, signal electromigration analysis on a logic circuit design for an integrated circuit. The performing includes producing a state-analysis-based transition matrix using state analysis data for the logic circuit design, and defining transition submatrices by partitioning the state-analysis-based transition matrix for the logic circuit design. In addition, the performing includes determining per simulation weighting factors for the signal electromigration analysis using the transition submatrices and switching factor data, and applying the determined per simulation weighting factors to weight, for signal electromigration analysis, electrical current data collected from a plurality of timing simulations of a transient simulator for the logic circuit design. Further, performing the signal electromigration analysis includes using the weighted electrical current data in assessing signal electromigration impact on circuit performance of the logic circuit design. In addition, the computer-implemented method includes, based on the assessing of the impact on circuit performance, facilitating generating a modified logic circuit design through modifying of the logic circuit design.

[0004] Computer program products and computer systems relating to one or more aspects are also claimed herein. Further, services relating to one or more aspects are also described and may be claimed herein.

[0005] Additional features and advantages are realized through the techniques described herein. Other embodiments and aspects are described in detail herein and are considered part of the claimed aspects.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] One or more aspects are particularly pointed out and distinctly claimed as examples in the claims at the conclusion of the specification. The foregoing and objects, features, and advantages of one or more aspects are apparent from the following detailed description taken in conjunction with the accompanying drawings in which:

[0007] FIG. **1** depicts one example of a computing environment to include and/or use one or more of the aspects of the present disclosure;

[0008] FIG. **2** depicts one embodiment of a computer program product with a signal electromigration analysis and circuit design module, in one or more aspects of the present disclosure;

[0009] FIG. **3** depicts one embodiment of a signal electromigration analysis and circuit design process, in accordance with one or more aspects of the present disclosure;

[0010] FIG. **4** illustrates an exemplary logic gate and related timing graph model;

[0011] FIG. **5** illustrates another exemplary logic circuit and an associated channel connected components (CCC) timing graph model built using transistor-level static timing;

[0012] FIG. **6**A depicts another embodiment of a logic circuit design and associated channel connected components (CCC) timing graph model;

[0013] FIG. **6**B is an exemplary state-analysis-based transition matrix containing transitions within a channel connected components expression of the logic circuit design of FIG. **6**A, in accordance with one or more aspects of the present disclosure;

[0014] FIG. **7** depicts one embodiment of signal electromigration weight factor determination, in accordance with one or more aspects of the present disclosure;

[0015] FIG. **8** depicts one embodiment of a full timing model signal electromigration weight analysis workflow, in accordance with one or more aspects of the present disclosure;

[0016] FIG. **9**A-**9**B depict exemplary transition submatrices per output of a logic circuit design obtained by partitioning the full state-analysis-based transition matrix, in accordance with one or more aspects of the present disclosure;

[0017] FIG. **10** depicts one embodiment of a per output signal electromigration weight factor workflow, in accordance with one or more aspects of the present disclosure;

[0018] FIG. **11**A is another exemplary logic gate of a logic circuit design;

[0019] FIGS. **11**B-**11**C depict exemplary per timing arc transition submatrices (or per propagate segment (pseg)) obtained by partitioning the state-analysis-based transition matrix for the logic circuit design of FIG. **11**A per timing arc, in accordance with one or more aspects of the present disclosure;

[0020] FIG. **12** depicts one embodiment of a per pseg signal electromigration weighting factor workflow, in accordance with one or more aspects of the present disclosure;

[0021] FIG. **13** depicts one embodiment of a hybrid signal electromigration workflow, in accordance with one or more aspects of the present disclosure;

[0022] FIG. **14** is a flow diagram of a design process used in semiconductor design, manufacture, and test, in accordance with one or more aspects of the present disclosure;

[0023] FIG. **15** depicts a further example of integrated circuit fabrication, including generating physical design data, in accordance with one or more aspects of the present disclosure; and

[0024] FIG. **16** shows an exemplary high-level Electronic Design Automation (EDA) tool flow, in which aspects of present disclosure can be employed.

DETAILED DESCRIPTION

[0025] Aspects of the present disclosure and certain features, advantages, and details thereof, are explained more fully below with reference to the non-limiting example(s) illustrated in the accompanying drawings. Descriptions of well-known systems, devices, processing techniques, etc., are omitted so as not to unnecessarily obscure the disclosure in detail. It should be understood, however, that the detailed description and the specific example(s), while indicating aspects of the disclosure, are given by way of illustration only, and are not by way of limitation. Various substitutions, modifications, additions, and/or arrangements, within the spirit and/or scope of the underlying inventive concepts will be apparent to those skilled in the art for this disclosure. Note further that reference is made below to the drawings, where the same or similar reference numbers used throughout different figures designate the same or similar components. Also, note that numerous inventive aspects and features are disclosed herein, and unless otherwise inconsistent, each disclosed aspect or feature is combinable with any other disclosed aspect or feature as desired for a particular application of the concepts disclosed.

[0026] Note also that illustrative embodiments are described below using specific code, designs, architectures, protocols, layouts, schematics, systems, or tools only as examples, and not by way of limitation. Furthermore, the illustrative embodiments are described in certain instances using particular software, hardware, tools, and/or data processing environments only as example for clarity of description. The illustrative embodiments can be used in conjunction with other comparable or similarly purposed structures, systems, applications, architectures, etc. One or more aspects of an illustrative control embodiment can be implemented in software, hardware, or a combination thereof.

[0027] As understood by one skilled in the art, program code, as referred to in this application, can include software and/or hardware. For example, program code in certain embodiments of the present disclosure can utilize a software-based implementation of the functions described, while other embodiments can include fixed function hardware. Certain embodiments combine both types of program code. Examples of program code, also referred to as one or more programs, are depicted in FIG. **1**, including operating system **122** and signal electromigration analysis and circuit design module **200**, which are stored in persistent storage **113**.

[0028] One or more aspects of the present disclosure are incorporated in, performed and/or used by a computing environment. As examples, the computing environment can be of various architectures and of various types, including, but not limited to: personal computing, client-server, distributed, virtual, emulated, partitioned, non-partitioned, cloud-based, quantum, grid, time-sharing, clustered, peer-to-peer, mobile, having one node or multiple nodes, having one or more processor sets, each with one processor or multiple processors, and/or any other type of environment and/or configuration, etc., that is capable of executing a process (or multiple processes) that, e.g., perform capacitance extraction-related processing, such as disclosed herein. Aspects of the present disclosure are not limited to a particular architecture or environment.

[0029] Prior to further describing detailed embodiments of the present disclosure, an example of a computing environment to include and/or use one or more aspects of the present disclosure is discussed below with reference to FIG. **1**.

[0030] Various aspects of the present disclosure are described by narrative text, flowcharts, block diagrams of computer systems and/or block diagrams of the machine logic included in computer program product (CPP) embodiments. With respect to any flowcharts, depending upon the technology involved, the operations can be performed in a different order than what is shown in a given flowchart. For example, again depending upon the technology involved, two operations shown in successive flowchart blocks may be performed in reverse order, as a single integrated step, concurrently, or in a manner at least partially overlapping in time.

[0031] A computer program product embodiment ("CPP embodiment" or "CPP") is a term used in the present disclosure to describe any set of one, or more, storage media (also called "mediums") collectively included in a set of one, or more, storage devices that collectively include machine readable code corresponding to instructions and/or data for performing computer operations specified in a given CPP claim. A "storage device" is any tangible device that can retain and store instructions for use by a computer processor. Without limitation, the computer readable storage medium may be an electronic storage medium, a magnetic storage medium, an optical storage medium, an electromagnetic storage medium, a semiconductor storage medium, a mechanical storage medium, or any suitable combination of the foregoing. Some known types of storage devices that include these mediums include: diskette, hard disk, random access memory (RAM), read-only memory (ROM), erasable programmable read-only memory (EPROM or Flash memory), static random access memory (SRAM), compact disc read-only memory (CD-ROM), digital versatile disk (DVD), memory stick, floppy disk, mechanically encoded device (such as punch cards or pits/lands formed in a major surface of a disc) or any suitable combination of the foregoing. A computer readable storage medium, as that term is used in the present disclosure, is not to be construed as storage in the form of transitory signals per se, such as radio waves or other freely propagating electromagnetic waves, electromagnetic waves propagating through a waveguide, light pulses passing through a fiber optic cable, electrical signals communicated through a wire, and/or other transmission media. As will be understood by those of skill in the art, data is typically moved at some occasional points in time during normal operations of a storage device, such as during access, de-fragmentation or garbage collection, but this does not render the storage device as transitory because the data is not transitory while it is stored.

[0032] Computing environment **100** contains an example of an environment for the execution of at least some of the computer code involved in performing the inventive methods, such as signal electromigration analysis and circuit design module block **200**. In addition to block **200**, computing environment **100** includes, for example, computer **101**, wide area network (WAN) **102**, end user device (EUD) **103**, remote server **104**, public cloud **105**, and private cloud **106**. In this embodiment, computer **101** includes processor set **110** (including processing circuitry **120** and cache **121**), communication fabric **111**, volatile memory **112**, persistent storage **113** (including operating system **122** and block **200**, as identified above), peripheral device set **114** (including user interface (UI) device set **123**, storage **124**, and Internet

of Things (IoT) sensor set **125**), and network module **115**. Remote server **104** includes remote database **130**. Public cloud **105** includes gateway **140**, cloud orchestration module **141**, host physical machine set **142**, virtual machine set **143**, and container set **144**.

[0033] Computer **101** may take the form of a desktop computer, laptop computer, tablet computer, smart phone, smart watch or other wearable computer, mainframe computer, quantum computer or any other form of computer or mobile device now known or to be developed in the future that is capable of running a program, accessing a network or querying a database, such as remote database **130**. As is well understood in the art of computer technology, and depending upon the technology, performance of a computer-implemented method may be distributed among multiple computers and/or between multiple locations. On the other hand, in this presentation of computing environment **100**, detailed discussion is focused on a single computer, specifically computer **101**, to keep the presentation as simple as possible. Computer **101** may be located in a cloud, even though it is not shown in a cloud in FIG. **1**. On the other hand, computer **101** is not required to be in a cloud except to any extent as may be affirmatively indicated.

[0034] Processor set **110** includes one, or more, computer processors of any type now known or to be developed in the future. Processing circuitry **120** may be distributed over multiple packages, for example, multiple, coordinated integrated circuit chips. Processing circuitry **120** may implement multiple processor threads and/or multiple processor cores. Cache **121** is memory that is located in the processor chip package(s) and is typically used for data or code that should be available for rapid access by the threads or cores running on processor set **110**. Cache memories are typically organized into multiple levels depending upon relative proximity to the processing circuitry. Alternatively, some, or all, of the cache for the processor set may be located "off chip." In some computing environments, processor set **110** may be designed for working with qubits and performing quantum computing.

[0035] Computer readable program instructions are typically loaded onto computer **101** to cause a series of operational steps to be performed by processor set **110** of computer **101** and thereby effect a computer-implemented method, such that the instructions thus executed will instantiate the methods specified in flowcharts and/or narrative descriptions of computer-implemented methods included in this document (collectively referred to as "the inventive methods"). These computer readable program instructions are stored in various types of computer readable storage media, such as cache **121** and the other storage media discussed below. The program instructions, and associated data, are accessed by processor set **110** to control and direct performance of the inventive methods. In computing environment **100**, at least some of the instructions for performing the inventive methods may be stored in block **200** in persistent storage **113**.

[0036] Communication fabric **111** is the signal conduction paths that allow the various components of computer **101** to communicate with each other. Typically, this fabric is made of switches and electrically conductive paths, such as the switches and electrically conductive paths that make up busses, bridges, physical input/output ports and the like. Other types of signal communication paths may be used, such as fiber optic communication paths and/or wireless communication paths.

[0037] Volatile memory **112** is any type of volatile memory now known or to be developed in the future. Examples include dynamic type random access memory (RAM) or static type RAM. Typically, the volatile memory is characterized by random access, but this is not required unless affirmatively indicated. In computer **101**, the volatile memory **112** is located in a single package and is internal to computer **101**, but, alternatively or additionally, the volatile memory may be distributed over multiple packages and/or located externally with respect to computer **101**.

[0038] Persistent storage **113** is any form of non-volatile storage for computers that is now known or to be developed in the future. The non-volatility of this storage means that the stored data is maintained regardless of whether power is being supplied to computer **101** and/or directly to persistent storage **113**. Persistent storage **113** may be a read only memory (ROM), but typically at least a portion of the persistent storage allows writing of data, deletion of data and re-writing of data. Some familiar forms of persistent storage include magnetic disks and solid state storage devices. Operating system **122** may take several forms, such as various known proprietary operating systems or open source Portable Operating System Interface type operating systems that employ a kernel. The code included in block **200** typically includes at least some of the computer code involved in performing the inventive methods.

[0039] Peripheral device set **114** includes the set of peripheral devices of computer **101**. Data communication connections between the peripheral devices and the other components of computer **101** may be implemented in various ways, such as Bluetooth connections, Near-Field Communication (NFC) connections, connections made by cables (such as universal serial bus (USB) type cables), insertion type connections (for example, secure digital (SD) card), connections made though local area communication networks and even connections made through wide area networks such as the internet. In various embodiments, UI device set **123** may include components such as a display screen, speaker, microphone, wearable devices (such as goggles and smart watches), keyboard, mouse, printer, touchpad, game controllers, and haptic devices. Storage **124** is external storage, such as an external hard drive, or insertable storage, such as an SD card. Storage **124** may be persistent and/or volatile. In some embodiments, storage **124** may take the form of a quantum computing storage device for storing data in the form of qubits. In embodiments where computer **101** is required to have a large amount of storage (for example, where computer **101** locally stores and manages a large database) then this storage may be provided by peripheral storage devices designed for storing very large amounts of data, such as a storage area network (SAN) that is shared by multiple, geographically distributed computers. IoT sensor set **125** is made up of sensors that can be used in Internet of Things applications. For example, one sensor may be a thermometer and another sensor may be a motion detector.

[0040] Network module **115** is the collection of computer software, hardware, and firmware that allows computer **101** to communicate with other computers through WAN **102**. Network module **115** may include hardware, such as modems or Wi-Fi signal transceivers, software for packetizing and/or de-packetizing data for communication network

transmission, and/or web browser software for communicating data over the internet. In some embodiments, network control functions and network forwarding functions of network module **115** are performed on the same physical hardware device. In other embodiments (for example, embodiments that utilize software-defined networking (SDN)), the control functions and the forwarding functions of network module **115** are performed on physically separate devices, such that the control functions manage several different network hardware devices. Computer readable program instructions for performing the inventive methods can typically be downloaded to computer **101** from an external computer or external storage device through a network adapter card or network interface included in network module **115**.

[0041] WAN **102** is any wide area network (for example, the internet) capable of communicating computer data over non-local distances by any technology for communicating computer data, now known or to be developed in the future. In some embodiments, the WAN may be replaced and/or supplemented by local area networks (LANs) designed to communicate data between devices located in a local area, such as a Wi-Fi network. The WAN and/or LANs typically include computer hardware such as copper transmission cables, optical transmission fibers, wireless transmission, routers, firewalls, switches, gateway computers and edge servers.

[0042] End User Device (EUD) **103** is any computer system that is used and controlled by an end user (for example, a customer of an enterprise that operates computer **101**) and may take any of the forms discussed above in connection with computer **101**. EUD **103** typically receives helpful and useful data from the operations of computer **101**. For example, in a hypothetical case where computer **101** is designed to provide a recommendation to an end user, this recommendation would typically be communicated from network module **115** of computer **101** through WAN **102** to EUD **103**. In this way, EUD **103** can display, or otherwise present, the recommendation to an end user. In some embodiments, EUD **103** may be a client device, such as thin client, heavy client, mainframe computer, desktop computer and so on.

[0043] Remote server **104** is any computer system that serves at least some data and/or functionality to computer **101**. Remote server **104** may be controlled and used by the same entity that operates computer **101**. Remote server **104** represents the machine(s) that collect and store helpful and useful data for use by other computers, such as computer **101**. For example, in a hypothetical case where computer **101** is designed and programmed to provide a recommendation based on historical data, then this historical data may be provided to computer **101** from remote database **130** of remote server **104**.

[0044] Public cloud **105** is any computer system available for use by multiple entities that provides on-demand availability of computer system resources and/or other computer capabilities, especially data storage (cloud storage) and computing power, without direct active management by the user. Cloud computing typically leverages sharing of resources to achieve coherence and economies of scale. The direct and active management of the computing resources of public cloud **105** is performed by the computer hardware and/or software of cloud orchestration module **141**. The computing resources provided by public cloud **105** are typically implemented by virtual computing environments that run on various computers making up the computers of host physical machine set **142**, which is the universe of physical computers in and/or available to public cloud **105**. The virtual computing environments (VCEs) typically take the form of virtual machines from virtual machine set **143** and/or containers from container set **144**. It is understood that these VCEs may be stored as images and may be transferred among and between the various physical machine hosts, either as images or after instantiation of the VCE. Cloud orchestration module **141** manages the transfer and storage of images, deploys new instantiations of VCEs and manages active instantiations of VCE deployments. Gateway **140** is the collection of computer software, hardware, and firmware that allows public cloud **105** to communicate through WAN **102**.

[0045] Some further explanation of virtualized computing environments (VCEs) will now be provided. VCEs can be stored as "images." A new active instance of the VCE can be instantiated from the image. Two familiar types of VCEs are virtual machines and containers. A container is a VCE that uses operating-system-level virtualization. This refers to an operating system feature in which the kernel allows the existence of multiple isolated user-space instances, called containers. These isolated user-space instances typically behave as real computers from the point of view of programs running in them. A computer program running on an ordinary operating system can utilize all resources of that computer, such as connected devices, files and folders, network shares, CPU power, and quantifiable hardware capabilities. However, programs running inside a container can only use the contents of the container and devices assigned to the container, a feature which is known as containerization.

[0046] Private cloud **106** is similar to public cloud **105**, except that the computing resources are only available for use by a single enterprise. While private cloud **106** is depicted as being in communication with WAN **102**, in other embodiments a private cloud may be disconnected from the internet entirely and only accessible through a local/private network. A hybrid cloud is a composition of multiple clouds of different types (for example, private, community or public cloud types), often respectively implemented by different vendors. Each of the multiple clouds remains a separate and discrete entity, but the larger hybrid cloud architecture is bound together by standardized or proprietary technology that enables orchestration, management, and/or data/application portability between the multiple constituent clouds. In this embodiment, public cloud **105** and private cloud **106** are both part of a larger hybrid cloud.

[0047] The computing environment described above is only one example of a computing environment to incorporate, perform and/or use one or more aspects of the present disclosure. Other examples are possible. Further, in one or more embodiments, one or more of the components/modules of FIG. **1** need not be included in the computing environment and/or are not used for one or more aspects of the present disclosure. Further, in one or more embodiments, additional and/or other components/modules can be used. Other variations are possible.

[0048] By way of example, one or more embodiments of a signal electromigration analysis and circuit design module and process are described initially with reference to FIGS. **2A-3**. FIGS. **2A-2B** depict one embodiment of a signal

electromigration analysis and circuit design module **200** that includes code or instructions to perform signal electromigration analysis and circuit design modification processing, in accordance with one or more aspects of the present disclosure, and FIG. **3** depicts one embodiment of a signal electromigration analysis and circuit design process, in accordance with one or more aspects of the present disclosure.

[0049] Referring to FIGS. **1**-**2**B, signal electromigration analysis and circuit design module **200** includes, in one example, various sub-modules used to perform processing, in accordance with one or more aspects of the present disclosure. The sub-modules are, e.g., computer-readable program code (e.g., instructions) in computer-readable media (e.g., persistent storage (e.g., persistent storage **113**, such as a disk) and/or a cache (e.g., cache **121**), as examples). The computer-readable media can be part of a computer program product and can be executed by and/or using one or more computers, such as computer(s) **101**; one or more processor sets **110** (FIG. **1**); processors, such as one or more processors of processor set **110**; and/or processing circuitry, such as processing circuitry of processor set **110**, etc.

[0050] As noted, FIGS. **2**A-**2**B depict one embodiment of signal electromigration analysis and circuit design module **200** which, in one or more implementations, includes, or facilitates, signal electromigration analysis and circuit design processing in accordance with one or more aspects of the present disclosure. In the embodiment of FIG. **2**A, example sub-modules of signal electromigration analysis and circuit design module **200** include a perform signal electromigration analysis sub-module **202** to facilitate performing signal electromigration analysis on a logic circuit design for an integrated circuit. Perform signal electromigration analysis sub-module **202** includes, in or one or more embodiments, a produce transition matrix sub-module **204** which produces a state-analysis-based transition matrix using state analysis data from the logic circuit design, and a define transition submatrices sub-module **206** which defines transition submatrices by partitioning the state-analysis-based transition matrix for the logic circuit design. In one or more embodiments, the transition submatrices cover the entire transition matrix. In addition, the perform signal electromigration analysis sub-module **202** includes, in one or more embodiments, a determine per simulation weighting factors sub-module **208** which facilitates determining per simulation weighting factors for the signal electromigration analysis using the transition submatrices and switching factor data. In one or more embodiments, perform signal electromigration analysis sub-module **202** further includes a weight collected electrical current data sub-module **210**, which applies the determined per simulation weighting factors to weight, for signal electromigration analysis, electrical current data collected from a plurality of timing situations of a transient simulator for the logic circuit design. In addition, perform signal electromigration analysis sub-module **202** includes an assess signal electromigration impact sub-module **212**, which uses the weighted electrical current data in assessing signal electromigration impact on circuit performance of the logic circuit design. Un In one or more embodiments, signal electromigration analysis and circuit design module **202** further includes a facilitate generation of modified logic circuit design sub-module **214** which, based on the assessing of the impact on circuit performance, results in generating a modified logic circuit design through modifying of the logic circuit design, for instance, to remediate one or more identified signal electromigration issues with the logic circuit design.

[0051] Note that although various sub-modules are described herein, signal electromigration analysis and circuit design processing, such as disclosed, can use, or include, additional, fewer, and/or different sub-modules. A particular sub-module can include additional code, including code of other sub-modules, or less code. Further, additional and/or fewer sub-modules can be used. Many variations are possible.

[0052] As illustrated in FIG. **2**B, in one or more embodiments, define transition submatrices sub-module **206** can include a define one transition submatrices type sub-module **216** to facilitate defining one type of transition submatrices from one partitioning of the state-analysis-based transition matrix, and a define another transition submatrices type sub-module **218** to facilitate defining another type of transition submatrices from another partitioning of the state-analysis-based transition matrix. In one or more embodiments, the per simulation weighting factors are determined using at least one submatrices type of the one transition submatrices type and the other transmission submatrices type. In one or more embodiments, the one transition submatrices type includes transition submatrices per output of the logic circuit design, and the other transition submatrices type includes transition submatrices per timing arc (or propagate segment (pseg)) of the logic circuit design.

[0053] In one or more implementations, determining the per simulation weighting factors for the electromigration analysis further includes ascertaining one set of per simulation weighting factors using the transition submatrices per output of the logic circuit design. Based on there being no zeroing of output switching factors associated with use of the transition submatrices per output, then the ascertained one set of weighting factors is used as the determined per simulation weighting factors for the electromigration analysis.

[0054] In one or more implementations, determining the per simulation weighting factors for the electromigration analysis further includes using both types of per simulation weighting factors. For instance, based on there being a zeroing of one or more output switching factors employing the transition submatrices per output, then determining of the per simulation weighting factors for electromigration analysis further includes ascertaining another set of per simulation weighting factors using the transition submatrices per timing arc of the logic circuit design. Based on there being no singularity associated with use of the transition submatrices per timing arc of the logic circuit design, then the ascertained other per simulation weight factors are used as the determined per simulation factors for the signal electromigration analysis. In one or more embodiments, based on there being one or more singularities associated with the transition submatrices per timing arc of the logic circuit design, then the ascertained one set of per simulation weighting factors using the transition submatrices per output can be used as the determined per simulation weighting factors for the signal electromigration analysis.

[0055] In one or more embodiments, where the transition matrix is determined to include one or more multi-state transition simulations, then the per simulation weighting factors for the signal electromigration analysis can be deter-

mined using the full timing model SEM weight analysis, that is, using the state-analysis-based transition matrix rather than the transition submatrices.

[0056] In one or more implementations, using the weighted electrical current data in assessing signal electromigration impact occurs within a transistor-level static timing context.

[0057] Advantageously, one or more aspects disclosed herein improve processing within a computing environment by facilitating design and manufacture of integrated circuits, specifically, by facilitating very large-scale integration (VLSI) designs and devices based on analysis and optimization of circuit designs to address one or more identified signal electromigration issues. Disclosed herein, in one or more embodiments, is an enhanced design tool, or enhanced signal electromigration analysis and circuit design tool, to facilitate performing automated electromigration analysis and modification of a logic circuit design of an integrated circuit. The tool is particularly advantageous in analyzing high performance custom logic circuit designs. In one or more embodiments, the signal electromigration analysis and circuit design module and process disclosed minimize singular solutions, and insure adequate weight magnitudes for assessing signal electromigration impact on circuit performance of a logic circuit design. Further, in one or more embodiments, the signal electromigration analysis and circuit design module and process disclosed minimize negative weights.

[0058] In one or more embodiments, the signal electromigration analysis and circuit design module is used, in accordance with one or more aspects of the present disclosure, to perform signal electromigration analysis and circuit design processing. FIG. **3** depicts one example of a signal electromigration analysis and circuit design process **300**, such as disclosed herein. The process is executed, in one or more embodiments, by a computer (e.g., computer **101** (FIG. **1**)), and/or one or more processor sets, such as a processor or processing circuitry (e.g., of processor set **110** of FIG. **1**). In one example, code or instructions implementing the process, are part of a module, such as signal electromigration analysis and circuit design module **200**. In other examples, the code can be included in one or more other modules and/or one or more other sub-modules of one or more other modules. Various options are available.

[0059] As illustrated in FIG. **3**, in one example, signal electromigration analysis and circuit design process **300** executing on one or more computers (e.g., computer **101** of FIG. **1**), one or more processor sets (e.g., processor set **110** of FIG. **1**, such as a processor or processing circuitry of the processor set) performs signal electromigration analysis on a logical circuit design for an integrated circuit **302**. Performing signal electromigration analysis on a logical circuit design includes, in one or more embodiments, producing a state-analysis-based transition matrix using state analysis data for the logic circuit design **304**, and defining transition submatrices by partitioning the state-analysis-based transition matrix for the logic circuit design **306**. Further, performing signal electromigration analysis on the logic circuit design includes, in one or more embodiments, determining per simulation weighting factors for the signal electromigration analysis using the transition submatrices and switching factor data **308**, and applying the determined per simulation weighting factors to weight, for signal electromigration analysis, electrical current data collected from a plurality of timing situations of a transient simulator for the logic circuit design **310**. In addition, performing signal electromigration analysis on the logic circuit design **302** includes, in one embodiment, using the weighted electrical current data in assessing signal electromigration impact on circuit performance on the logic circuit design **312**. In one or more embodiments, signal electromigration analysis and circuit design process **300** further includes facilitating generating a modified logic circuit design based on the assessing of the signal electromigration impact on circuit performance **314**. For instance, the facilitating generating can include, by way of example, generating a modified logic circuit design to address one or more identified signal electromigration issues or impacts on circuit performance based on the signal electromigration analysis of the logic circuit design. A variety of modifications to a logic circuit design can be, for instance, automatically implemented to address an identified signal electromigration issue. In one or more implementations, a logic circuit design can be modified automatically based on the signal electromigration analysis to, for instance, prevent an electromigration issue by, for example: increasing a metal width to reduce current density; reducing frequency; lowering a supplied voltage; shortening a wire length; reducing buffer size and clock lines; etc.

[0060] Conventionally, in VLSI (very large-scale integration) digital design, fabricated devices can include millions of transistors implementing hundreds of storage devices, functional logic circuits, and the like. The designs are often segmented or partitioned into sub-blocks (such as cores, units, macros, subhierarchies, and the like) to make the design process more manageable. For example, the design, placement, and routing of circuits may be conducted at a high level and a sub-block level, where the high level considers the complete device including all sub-blocks (known as in-context design) and the sub-block level considers the design of a single sub-block (known as out-of-context design). While a sub-block level design can be used in multiple instances within a device, conventionally, only a single version of the design of the sub-block is produced.

[0061] Timing analysis and timing considerations for a sub-block conventionally include constraints, such as the specification of the arrival time (AT) for each input signal at the entry of the sub-block and the specification of the required arrival time (RAT) for generating each output signal of the sub-block. The required arrival time must consider the propagation delay through the circuit, including the slew rate of the output signal. The propagation delays, slew rates, and a like are influenced by a variety of factors to be analyzed in designing the integrated circuit. Signal electromigration (SEM) analysis, is one type of analysis of a logic circuit design to facilitate optimizing the circuit design.

[0062] Electromigration is a type of potential fault mechanism in integrated circuits caused by the movement of atoms based on the flow of current through the material. Electromigration can result in both vacancies and deposits within, for instance, a wire, a gate, or other component of the integrated circuit. As part of the design and fabrication of integrated circuits, simulation of electromigration is performed in modern technology computer aided design (TCAD) tools. Results of the TCAD analysis can be used to modify, for instance, one or more design rules for a logic circuit design, and thereby modify the logic circuit design

itself to further resist signal electromigration during operation of the fabricated integrated circuit. In accordance with one or more aspects, automated signal electromigration analysis is provided herein, along with modification of one or more logic circuit designs to, for instance, eliminate an out of specification electromigration issue or otherwise reduce or minimize signal electromigration of the fabricated integrated circuit.

[0063] In one or more embodiments disclosed herein, certain novel static-transistor-level timing processes are employed to facilitate signal electromigration analysis. The processes disclosed allow for a highly automated set-up, with high performance relative to conventional dynamic signal electromigration processes, and provide excellent coverage since all possible charge states are considered.

[0064] In one or more embodiments, the signal electromigration analysis disclosed herein determines metrics to assess impact of signal electromigration on the resultant integrated circuit during the design phase. The metrics include, for instance, equivalent average DC current (I_{dc}), root mean square current (I_{rms}) and maximum current during a clock period (I_{peak}). The determined metrics are compared to allowable limits to identify any anticipated signal electromigration violations. In one or more embodiments, weighting factors are determined to apply to the ascertained electrical current data, which as described herein, is collected from a plurality of timing simulations of a transient simulator (or timing simulator) for the logic circuit design (in one embodiment).

[0065] By way of example, FIG. **4** illustrates one embodiment of a logic gate and a related timing graph model for the logic gate. In this example, the logic gate is a NAND2 gate with inputs A, B and output Y. Static timing builds a timing graph containing timing points (gate inputs/outputs) and timing arcs (delay segments through the gates/wires, also referred to herein as propagate segments or psegs). In one or more embodiments, timing data is propagated through the timing graph, which can be used to determine if data signals arrive too early or too late relative to clocks, to satisfy constraints, that is, the appropriate data is to be latched in the appropriate clock cycle. The initial arrival times (ATs) are inserted at the primary inputs, and timing arc delays are determined. The determined arc delays include separate early and late, clock and data signals, side inputs, rise/fall, etc. Results are propagated to timing points, and cumulative effects of delays through the design are obtained. The clock signal AT is compared to the data signal AT at the latches to ensure proper capture of data by each latch. Note that although FIG. **4** illustrates one single gate, in practice, the logic circuits can be made up of hundreds to millions of logic gates.

[0066] Delays and/or slews are evaluated on the timing arcs, or propagate segments (psegs). The psegs each represent a delay path, that is, a path a signal takes which is not instantaneous. One goal of static timing analysis is to estimate the delay and/or slew along a timing arc for circuit timing purposes. The delays and/or slews are a function of many factors including, for instance, differing side input states, early/late mode, rise/fall transitions, signals with differing clock cycles or differing start times, etc. Multiple calculations are typically required for a given pseg over varying conditions to determine all required delay/slew data. In the timing graph model of FIG. **4**, one timing arc (or pseg) is determined to have a delay of four, and the other a delay of 3, with the minimum solution and maximum solution arrival times at the output noted. As explained, this is a single gate example only with single delay per arc shown. In practice, there may be tens of delays per arc, hundreds to millions of such gates in a design where timing propagates gate-to-gate.

[0067] Transistor-level static timing analysis builds many such timing graphs. Each represents a set of propagate segments with some input/output commonality. In one or more embodiments, channel connected components (CCCs) analysis can be used in an evaluation state. In one embodiment, the inputs can be defined as the gates, and the outputs the drains (driving other gates or nets) and the delay arcs (that is, timing arcs or psegs) are defined inbetween the inputs and outputs. Note there can be an arbitrary number of inputs and outputs and/or an arbitrary number of psegs in a given channel connected component. In one or more embodiments, all timing and signal electromigration analysis can occur per channel connected component, which is roughly the equivalent of a gate-based analysis with delays and/or slews calculated and propagated through the channel connected components of the logic circuit design.

[0068] FIG. **5** illustrates one exemplary logic circuit or schematic model with two channel connected components CCC #**1 500**, CCC #**2 501**, which are shown in block format in the CCC timing graph model of FIG. **5**. Inputs and outputs for each CCC block are illustrated, with the blocks being handled, in one or more embodiments, similar to the gates described above in generating the timing graph model. For each CCC, state-analysis occurs to determine (in one embodiment) all the delay evaluations required. State-analysis determines possible conditions a CCC may experience, and determines what are the feasible transitions for a given timing arc (or pseg) within the CCC.

[0069] Transistor level static timing analysis leverages transient simulation to evaluate timing. In particular, this is the method by which all delay calculations for a given timing arc are typically determined, that is, on all active timing arcs. State analysis determines the complete state for a given delay calculation, such as specific inputs/outputs of interest, early or late mode, rising or falling transition, specific clock cycle, specific side inputs, etc. For each timing arc (or pseg) there are many such entries, and each is a simulation row (or 'simrow'). Each row defines the design state for a single simulation design state (transitions, side inputs, etc.). In one or more embodiments, there can be tens of these for a single timing arc, and thousands for all timing arcs of a CCC. The sim rows can each produce a different delay result on the timing arc. One simulation per delay and/or slew evaluation is obtained to cover all possible states. Standard static timing evaluates timing (delay/slew) per timing arc, then propagates the determined timings. A transition matrix can be defined from the state analysis data, and describes a subset of the total state data. Specifically, the transition matrix defines which pseg input is active and transitioning during a given delay/slew analysis, and it further defines the pseg output as active and transitioning or not, depending on the state of the circuit. In accordance with one or more aspects of disclosed herein, the transition matrix or state-analysis-based transition matrix, described is defined by such state analysis data.

[0070] In one or more embodiments disclosed herein, delays/slews can be modeled from voltage waveforms, and include current waveforms that are used herein to facilitate

the signal electromigration analysis disclosed. Thus, transistor-level static timing analysis data is repurposed herein for signal electromigration analysis. In particular, certain data available for timing purposes can be reused or repurposed for signal electromigration analysis. For instance, the timing data can include or generate current waveforms and state analysis data such as described herein. In particular, note that the state analysis data is used herein for a completely different purpose than the standard timing applications.

[0071] Note also that many simulations occur per channel connected component. In accordance with one or more aspects, the per simulation charge data is to be merged. For instance, for each net, the charge data is summed for all simulations which impact that net. To do so, the weight for each simulation needs to be obtained. Note that the sum of all weights being applied must equal the net's switching factor. For instance, if the net switches at ten percent of the clock cycles, then the sum of the switching factor for that net is 0.1. If the net switches with each clock cycle, then the switching factor for that net is 1.0.

[0072] In accordance with one or more aspects of the present disclosure, a transition matrix is defined per channel connected component (e.g., per logic circuit design (e.g., per sub-circuit design)). The transition matrix indicates inputs and outputs of interest for a given simulation. In one embodiment, rows include input/output nets (and rise:fall (R:F) transitions), and columns indicate individual transient simulations (e.g., early/late mode, varying input transitions, varying side input states, varying timing arcs measured). A binary 1 equates to a transition of interest during simulation, otherwise the value is 0. Note in this regard that 0 does not mean no transition, but rather the transition is not of interest (e.g., not a measured pseg). In one or more embodiments, many more simulations than pins are ascertained, resulting in a highly under-determined matrix.

[0073] By way of example, FIG. **6**A-**6**B illustrate one embodiment of a logic circuit design and associated timing graph model (FIG. **6**A), and an exemplary state-analysis-based transition matrix (FIG. **6**B). Note that the logic circuit design of FIG. **6**A is one example only of a channel connected component for which signal electromigration analysis can be performed as described herein. As illustrated in the associated timing graph model of FIG. **6**A, the channel connected component includes a single input A, and two outputs Y**1**, Y**2**, with two timing arcs or propagate segments (pseg**1**, pseg**2**) between the inputs and outputs.

[0074] FIG. **6**B illustrates one embodiment of state analysis data (for a state-analysis-based transition matrix) containing transitions within, for instance, a channel connected component (CCC) expression of a logic circuit design, in accordance with one or more aspects of the present disclosure. Note that in the table, the 1 and 0 refer to changes or no changes of interest, and not states. Further, note that the table of FIG. **6**B does not specify what physically switches, just what is of interest in a delay evaluation. In static timing, the calculations are performed per timing arc. For instance, in simulation #**1** (sim #**1**), A pin rises and Y**1** falls, which defines the delay calculation of interest (pseg**1**). Y**2** also falls in simulation #**1**, but pseg**2** is not in the delay arc of interest, so it is not marked in the table (or resultant transition matrix). In this manner, the state-analysis data, which defines every possible transition a channel connected component can make, can be leveraged to produce a state-analysis-based transition matrix for use in determining per-simulation signal electromigration weights, such as described herein. Note that the embodiment of FIG. **6**B depicts one example only of state analysis data to form a transition matrix such as disclosed. In practice, there may be many more simulations for a given block or CCC.

[0075] As noted, in one or more embodiments of performing signal electromigration analysis of a logic circuit design such as described herein, per simulation weighting factors are to be determined. In one or more implementations, switching factors are assigned to each unique input/output net (e.g., logic circuit design). The switching factors represent a percentage of clock cycles that the net transitions. There are a variety of approaches to estimating fractions of a total switching factor value that can be used in determining the switching factors. In one or more embodiments, bounding switching factor values can be assigned (e.g., 0.1 static signals, 1.0 dynamic signals).

[0076] With the switching factors determined, per simulation weighting factors can then be ascertained. In one embodiment, weights can be determined using linear algebra on a transition matrix and switching factor vector, with the weight 'W' in the equivalent average DC current (I_{dc}) and root mean square current (I_{rms}) equations being the simulation weights. Note that the weighting factors are a function of the switching factors, and in the transition matrix. The weighting factors define contribution of a simulation's charge data to the cumulative I_{dc}, and I_{rms}. In particular, the sums of the weights in the non-zero columns must equal the switching factor, and the problem is under-determined, with more unknowns than equations.

[0077] FIG. **7** illustrates one embodiment representative of a signal electromigration weight evaluation solution. In this embodiment, linear algebra is used to solve for the weights or weighting factors: AX=B for X, where X is the unknown vector of weights (W**1**-W**8**). The weighting factor solution uses the transition matrix (A) and the switching factors vector (B), where the switching factor is a percent of clock cycles that the timing clock actually switches. The problem is an over constrained linear algebra problem, and many solution techniques are available, with one approach being the quadratic programing (or least squares) approach. In one embodiment, the approach includes:

[0078] Defining a target vector C (size=#columns of A, the same as X vector)

[0079] Function of switching factors, non-zero transition matrix data

[0080] Seeking solution for vector X which minimizes error between itself and target, vector C

[0081] Defining QP: min $(X-C)^T(X-C)$ such that AX=B (where T is transpose, and B is the switching factors)

[0082] This constrains X to one possible solution of AX=B

[0083] Representative solution: $X=A^T(AA^T)^{-1}(B-AC)+C$

[0084] FIG. **8** depicts one embodiment of a full timing model signal electromigration (SEM) weight analysis workflow **800**, in accordance with one or more aspects of present invention. In this SEM weight analysis process, a transition matrix is defined for each channel connected component (CCC) **802**. In particular, for each CCC, an N×M transition matrix is defined **804** where (in one example only) N=2*number active I/O nodes on CCC (rise/fall) and

M=number of simrows on the CCC. Note that in one embodiment, the data can be obtained in a prior loop over all CCC timing arcs (or psegs), counting nodes and simrows for that CCC, but they otherwise can be known from the state analysis, or other set-up or preprocessing. For each pseg (or timing arc) **806**, and each simrow (or simulation row) **808**, the process includes defining the current column of the transition matrix **810** by setting the current column number equal to the current simrow column, and entering a 1 for a transitioning row input and output (where input always switches, and output may not switch), or otherwise entering 0. Processing determines whether the last simrow has been processed **812**, and if "No", continues until each simrow for that pseg has been processed. The process continues for each pseg **814**, and once the last pseg has been processed, the state-analysis-based transition matrix is complete for the CCC by, for instance, removing redundant rows (no new information), and then using the matrix in evaluating the weights **816** (in one or more embodiments). Processing continues for each channel connected component (CCC) **818**, and completes once the last channel connected component (CCC) has been processed **820**. Note that FIG. **8** depicts one embodiment only of a full timing model SEM weight analysis approach, such as described herein. In one or more other embodiments, it would be possible to, for instance, cache all state matrix information, and evaluate weights as a post process. The significant detail is that the state-analysis-based transition matrix is completely defined at that point.

[0085] Note that in certain embodiments, the full timing model SEM weight analysis approach may not be optimal. For instance, matrices can in certain cases be singular, where no inverse exists, and no solution is possible. Further, the weighting data can in certain cases be poorly behaved. For instance, linear algebra can place constraints on the switching factors, which can be violated. When violated, negative weights can result, and these negative weights cannot be optimized. Further, overly small weights can require manual intervention to debug or scale up. Disclosed hereinbelow with reference to FIGS. **9A-13** are certain enhanced, automated signal electromigration analysis processes that mitigate these issues, and make the transistor level timing approach more practical for signal electromigration analysis and design modification.

[0086] One goal of the automated signal electromigration analysis and design modification approaches disclosed herein is to minimize singularities at all stages of the signal electromigration process. As noted a singular matrix indicates no solution is found for a channel connected component (CCC), and all weights for that channel connected component (CCC) are typically set to 0. A singularity can be driven by an imbalance in the switching factors out (SF_{out}) versus the switching factors in (SF_{in}) that violate matrix restraints.

[0087] An attribute of some transition matrices is that it can define switching inputs (e.g., for a NOR2 logic gate) where outputs may not switch. That is, not all input switching results in output switching (OS). When the output of a timing arc does not switch with the input, this condition is known as Non Output Switching (NOS). This is expected behavior and is not an issue, it is just an attribute of the matrix. What is an issue is when OS simulations have negative weights calculated and must be zeroed out. Pursuant to this disclosure, most/all negative weights are to be evaluated on NOS timing arcs. Advantageously, this allows the invalidative weights to be put on the arcs that contribute little charge.

[0088] In one or more implementations, the sum of the non-zero transient weights equals the nodal switching factor. In the under-constrained problem, an infinite number of solutions are possible. Some solutions require negative weights to satisfy the summation equation. Negative weights are undefined in the signal electromigration process and are artifacts of the matrix solution. One functional heuristic solution method is to zero out negative weights and resolve using the reduced matrix. This can produce a valid solution, but can also result in a singular matrix. Further, an issue can arise involving some positive weights being overly small, requiring further post processing to debug for validity, and/or scale-up for impact. This can be a time-consuming process and can result in an approximation. In one or more embodiments, negative weights can be excluded from the signal electromigration analysis, and small weights can under-count the impact. This can result in an optimistic solution, and addition manual intervention especially when zeroing "output switching" (OS) sensitizations, which can have significant charge associated. Further, the goal is to minimize zeroing of OS data by steering any zeroing to non-output switching (NOS) data.

[0089] As described herein, signal electromigration analysis processing is enhanced by defining transition submatrices through a partitioning of the state-analysis-based transition matrix for the logic circuit design. In one or more embodiments, the state-analysis-based data can be partitioned into multiple different types of transition submatrices including, for instance, perOutput and per timing arc (perPseg), as described herein.

[0090] FIGS. **9A-9B** depict exemplary transition submatrices perOutput of a logic circuit design obtained by partitioning a full state-analysis-based transition matrix. In this example, the tables of FIGS. **9A** and **9B** are obtained by partitioning the full state-analysis data table of FIG. **6B**, in one example only. In this process, the weight analysis defines and solves constraints using output-specific submatrices. In one embodiment, the defining of the transition submatrices is perOutput such that the submatrices span sensitizations for a single output, in which case the number of submatrices equals the number of outputs in the logic circuit design. The defined transition submatrices together span the full simulation space. In this case, the analysis is limited to only those inputs driving the particular output of the defined transition submatrix (including NOS). This results in few simulations, and larger weights to satisfy the switching factor. As noted, the approach in FIG. **8** can result in small weights, and the perOutput approach simplifies on the fly switching factor correction compared with using the full timing mode SEM weight analysis approach. Further, defining transition submatrices perOutput of the logic circuit design simplifies the solution and there are fewer differing switching factors in a given matrix. In this approach, each of the transition submatrices is solved independently for weights. Advantageously, using the per timing arc approach, singularities are substantially eliminated using default bounding switching factors.

[0091] FIG. **10** illustrates one embodiment of a perOutput signal electromigration (SEM) weight analysis process **1000** which includes for each channel connected component (CCC) **1002**, and for each pseg (or timing arc) **1004**,

determining whether this is a new output to be considered **1006**. If the output is not new, then the transition submatrix for that output can be retrieved, for instance, from memory **1010**. If the output is a new output to be considered, then a transition submatrix is defined for that output as an N×M transition submatrix, where N=2*(1+Number inputs feeding the output) and M=number of simrows on all psegs sharing the output.

[0092] For each simrow **1012** a current column of the transition submatrix is defined **1014**, where the current column number equals the current simrow count for the current output, and where 1 is inserted for active inputs/outputs, and 0 otherwise. Processing determines whether the last simrow has been considered **1016**, and if so, whether the last pseg driving the output has been considered **1018**. If either is no, processing loops back until the last simrow has been processed, and the last pseg driving the output as been considered.

[0093] As illustrated in FIG. **10**, the transition submatrix can be completed for that output **1020** by removing any redundant rows (that is, no new information), and then used to evaluate the weights **1020**. Processing determines whether the last channel connected component (CCC) has been processed **1022**, and repeats the process until complete **1024**.

[0094] By way of further example, FIGS. **11**A-**11**C illustrate another exemplary logic gate of a logic circuit design (FIG. **11**A) and exemplary transition submatrices perPseg of the logic gate (FIGS. **11**B-**11**C) obtained by partitioning the full state analysis transition matrix. In this approach, simulation weights are solved using pseg-specific submatrices. PerPseg means multiple submatrices are being defined, where each submatrix only spans sensitizations for a single input/output pair. In particular, the number of submatrices equals the number of psegs. The analysis is limited to one input and one output per submatrix. This advantageously eliminates extraneous and/or repeated contributions driving singularities, and negative OS weights. Also, as noted, the smaller submatrices are easier to manage when manual intervention is required. The perPseg approach is similar to the above described perOutput approach of FIGS. **9**A-**10**, but with finer granularity in many cases. There are fewer sensitizations per matrix, which results in more robust weights to satisfy the switching factors than in the perOutput approach. Advantageously, using the perPseg approach, singularities are substantially eliminated using default bounding switching factors. Further, negative OS weights are further reduced, or eliminated, even relative to the perOutput approach. The exception is with templated CCCs, which can have multiple switching inputs/outputs, internal nets, or other items requiring the full timing model SEM weight analysis of FIG. **8**.

[0095] In the example of FIGS. **11**A-**11**C, the Pseg**1** (FIG. **11**B) and Pseg**2** (FIG. **11**C) depicted tables or transition submatrices can be obtained by partitioning the full state-analysis-based transition matrix for the logic circuit design of FIG. **11**A. Note that in this regard that certain shared output switching factors (SF) need to be pro-rated, as illustrated in FIGS. **11**B-**11**C. Because there are two psegs in this example, the switching factor is divided by 50%. That is, in the per Pseg approach the shared output switching factor can require scaling. With fan-in, output sensitizations are repeated over multiple inputs. A given pseg only considers some subset of the non-zero entries. But the sum of all weights over all non-zero entries needs to equal the total output switching factor. The solution is to scale the switching factor output based on its contributions to the total output weighting. For instance, the scaled switching factor output for a pseg is equal to the summation of the output switching psegs in a given submatrix (OS_{psegs}) divided by the summation of the output switching psegs in the entire CCC (OS_{ccc}).

[0096] FIG. **12** illustrates one embodiment of a perPseg signal electromigration (SEM) weight analysis process **1200** which includes for each channel connected component (CCC) **1202**, and for each pseg (or timing arc) **1204**, defining an N×M transition submatrix, where N=4 (input/output, rise/fall) and M=number of simrows on the pseg **1206**. For each simrow **1208**, a current column of the transition submatrix is defined **1210**, where the current column number equals the current simrow count for the current pseg, and where the 1 is inserted for active inputs/outputs, and 0 otherwise. Processing determines whether the last simrow has been considered **1212**, and if not, processing loops back until the last simrow has been processed. As illustrated, the transition submatrix is completed for that pseg by removing any redundant rows (that is, no new information), and then evaluating the weights **1214**. Processing determines whether the last pseg has been processed **1216**, and if so, whether the last CCC has been considered **1218**. If either is "No", processing loops back until the last pseg has been processed for each CCC, which completes the process **1220**.

[0097] As noted herein, both the perOutput SEM weight analysis and perPseg SEM weight analysis processes have advantages and one or more issues associated therewith in comparison with the full timing model SEM weight analysis approach using the full transition matrix produced, as described herein, using state analysis data for a logic circuit design. For instance, the perPseg approach can drive singular submatrices in certain cases (e.g., less than 1% of all simulations). In cases of a fan-in, where the switching factor output (SF_{out}) is less than or equal to the summation of the switching factor input (SF_{in}), the perPseg approach can result in singularity. One solution is to use the perOutput approach under these conditions. Further, the perPseg approach can be more pessimistic than the perOutput approach, and another solution can be to use the perOutput approach when a result does not zero out OS data, and to fall back on the perPseg approach when the perOutput approach zeros OS data. In certain simulations, the full timing model (SEM) weight analysis approach may still be desirable. For instance, the perPseg approach requires a single input switching, and the perOutput approach requires single/shared input switching, and both approaches require single/no output switching. Rarely, certain specialized circuits violate these conditions. The logic circuit designs that violate the conditions are typically smaller circuits, not frequently used. In these cases, the full timing model (SEM) weight analysis can further be leveraged, such as described below with reference to the hybrid SEM weight analysis approach of FIG. **13**.

[0098] As noted, in one or more embodiments, a hybrid SEM weight analysis approach can be used in determining per simulation weighting factors for signal electromigration analysis, such as described herein. Note that the approach of FIG. **13** represents one embodiment only of such a hybrid SEM weight analysis approach. The hybrid SEM weight

analysis approach of FIG. **13** begins with obtaining (e.g., producing, retrieving, etc.) a state-analysis-based transition matrix for the logic circuit design **1302**. The process determines whether the matrix contains all single transitions **1304**. In one or more embodiments, single transitions include a 1 on an input node, and a 1 on an output node of the logic circuit design. As noted, for certain templates, there are sometimes multiple transitions occurring, and in those cases, the full channel connected component (CCC) weighting solution of FIG. **8** can be employed **1306** to determine the per simulation weighting factors for the signal electromigration analysis. Assuming that the transition matrix includes all single transitions, then processing can execute, in the embodiment of FIG. **13**, the perOutput weighting solution of FIG. **10 1308**. Processing then determines if there is any zeroed output switching data using the defined transition submatrices perOutput approach **1310**, and if not, the process is complete **1318**. If there is zeroed output switching executing the perOutput weighting solution, then (in one embodiment) processing executes the perPseg weighting solution of FIG. **12 1312**. Processing determines whether this results in a singular submatrix **1314**, and if "No" processing is complete **1318**. Otherwise, if a singular submatrix is obtained using the perPseg weighting solution, then the hybrid SEM weight analysis approach of FIG. **13** reverts to using the perOutput weighting solution approach **1316** for determining the per simulation weighting factors for signal electromigration analysis.

[0099] Advantageously, using a hybrid SEM weight analysis approach, such as the hybrid SEM weight analysis approach of FIG. **13**, singularities can be significantly reduced or eliminated, for instance, with either the perPseg or perOutput method. Any remaining singularities would typically be driven by templates, which can use the full timing model SEM weight analysis to resolve. In addition, output switching negative weights are significantly reduced by the perOutput approach, and largely eliminated by the perPseg approach.

[0100] Advantageously, computer-implemented methods, computer program products and computer systems are disclosed herein for facilitating signal electromigration analysis and circuit design by performing a plurality of signal transition current/charge density evaluations, and solving for required per transition weights using linear algebra processing of at least one of a plurality of submatrix solution approaches, such that the sum of the weights for active transitions on a given net equals the set switching factor (SF) for that net, and applying the weights in a summation of charge across all transitions influencing the current through a given net. In one or more embodiments, the solution executes using submatrices defined perOutput of a channel connected component (CCC), which minimizes singular solutions, and insures adequate weight magnitudes. In another embodiment, the solution executes using submatrices defined perPseg of the channel connected component (CCC), which advantageously minimizes singularities and negative weights, and ensures adequate weights. Note that with the perPseg approach, outputs shared between matrices are pro-rated such that the sum of weights across all matrices equal the global switching factor. Performing the weighting solution using a plurality of submatrix solution approaches can be optimal in many cases, for instance, to minimize pessimism in the result, and manage any perPseg singularities. In one or more embodiments, the signal electromigration analysis disclosed herein is a static transistor-level timing-based signal electromigration analysis. In general, the signal electromigration analysis disclosed can be based on a plurality of charge data, with each representing a single transition.

[0101] The computer-implemented methods, computer program products and computer systems disclosed herein provide a due means for performing signal electromigration analysis leveraging transistor level static timing, and allow for efficient and accurate SEM analysis with minimal operator interaction. The processes disclosed herein utilize timing-based simulations on a design circuitry, providing extremely accurate sign-off quality SEM results, while avoiding high levels of human interaction required by existing industry standard SEM analysis approaches.

[0102] In one or more embodiments, a further step includes fabricating a physical integrated circuit in accordance with the VLSI circuit design. One non-limiting specific example that accomplishes this is described herein in connection with FIGS. **14-16**. For example, a design structure, based on the VLSI design, is provided to fabrication equipment to facilitate fabrication of a physical integrated circuit in accordance with the design structure.

[0103] In one or more embodiments, a layout is prepared based on the analysis. In one or more embodiments, the layout is instantiated as a design structure. In one or more embodiments, a physical integrated circuit is fabricated in accordance with the design structure.

[0104] As noted, in one or more embodiments, the layout is instantiated as a design structure. A physical integrated circuit is then fabricated in accordance with the design structure. Refer also to discussion for FIGS. **14-16**. FIG. **14** is a flow diagram of a design process used in semiconductor design, manufacture, and/or test. Once the physical design data is obtained, based, in part, on the design processes described herein, an integrated circuit designed in accordance therewith can be fabricated according to known processes that are generally described with reference to FIG. **14**. Generally, a wafer with multiple copies of the final design is fabricated and cut (i.e., diced) such that each die is one copy of the integrated circuit. At block **1410**, the processes include fabricating masks for lithography based on the finalized physical layout. At block **1420**, fabricating the wafer includes using the masks to perform photolithography and etching. Once the wafer is diced, testing and sorting each die is performed at **1430** to filter out any faulty die. Furthermore, referring to FIGS. **14-16**, in one or more embodiments, the at least one processor is operative to generate a design structure for the integrated circuit design in accordance with the VLSI design, and in at least some embodiments, the at least one processor is further operative to control integrated circuit manufacturing equipment to fabricate a physical integrated circuit in accordance with the design structure. Thus, the layout can be instantiated as a design structure, and the design structure can be provided to fabrication equipment to facilitate fabrication of a physical integrated circuit in accordance with the design structure. The physical integrated circuit will be improved (for example, because of proper signal electromigration mitigation) compared to circuits designed using prior art techniques, at least under conditions where there is the same CPU time budget for the design process. To achieve similar improvements with prior-art techniques, even if possible,

would require expenditure of more CPU time as compared to embodiments of the invention.

[0105] FIG. **15** depicts an example high-level Electronic Design Automation (EDA) tool flow, which is responsible for creating an optimized microprocessor (or other IC) design to be manufactured. A designer can start with a high-level logic description **1501** of the circuit (e.g. VHDL or Verilog). The logic synthesis tool **1503** compiles the logic and optimizes it without any sense of its physical representation, and with estimated timing information. Placement tool **1505** takes the logical description and places each component, looking to minimize congestion in each area of the design. The clock synthesis tool **1507** optimizes the clock tree network by cloning/balancing/buffering the latches or registers. The timing closure step **1509** performs a number of optimizations on the design, including buffering, wire tuning, and circuit repowering; its goal is to produce a design which is routable, without timing violations, and without excess power consumption. Routing stage **1511** takes the placed/optimized design and determines how to create wires to connect the components, without causing manufacturing violations. Post-route timing closure **1513** performs another set of optimizations to resolve any violations that are remaining after the routing. Design finishing **1515** then adds extra metal shapes to the netlist, to conform with manufacturing requirements. Checking steps **1517** analyze whether the design is violating any requirements such as manufacturing, timing, power, electromigration or noise. When the design is clean, the final step **1519** is to generate a layout for the design, representing all the shapes to be fabricated in the design to be fabricated **1521**.

[0106] One or more embodiments integrate the timing and signal electromigration analysis techniques herein with semiconductor integrated circuit design simulation, test, layout, and/or manufacture. In this regard, FIG. **16** shows a block diagram of an exemplary design flow **1600** used for example, in semiconductor IC logic design, simulation, test, layout, and manufacture. Design flow **1600** includes processes, machines and/or mechanisms for processing design structures or devices to generate logically or otherwise functionally equivalent representations of design structures and/or devices, such as those that can be analyzed using timing analysis or the like. The design structures processed and/or generated by design flow **1600** may be encoded on machine-readable storage media to include data and/or instructions that when executed or otherwise processed on a data processing system generate a logically, structurally, mechanically, or otherwise function-ally equivalent representation of hardware components, circuits, devices, or systems. Machines include, but are not limited to, any machine used in an IC design process, such as designing, manufacturing, or simulating a circuit, component, device, or system. For example, machines may include: lithography machines, machines and/or equipment for generating masks (e.g., E-V writers), computers, or equipment for simulating design structures, any apparatus used in the manufacturing or test process, or any machines for programming functionality equivalent representations of the design structures into any medium (e.g., a machine for programming a programmable gate array).

[0107] Design flow **1600** may vary depending on the type of representation being designed. For example, a design flow **1600** for building an application specific IC (ASIC) may differ from a design flow **1600** for designing a standard component or from a design flow **1600** for instantiating the design into a programmable array, for example a programmable gate array (PGA) or a field programmable gate array (FPGA).

[0108] FIG. **16** illustrates multiple such design structures **1620** that is preferably processed by a design process **1610**. Design structure **1620** may be a logical simulation design structure generated and processed by design process **1610** to product a logically equivalent functional representation of a hardware device. Design structure **1620** may also or alternatively comprise data and/or program instructions that when processed by design process **1610**, generate a functional representation of the physical structure of a hardware device. Whether representing functional and/or structural design features, design structure **1620** may be generated using electronic computer-aided design (ECAD), such as implemented by a core developer/designer. When encoded on a gate array or storage medium, design structure **1620** may be accessed and processed by one or more hardware and/or software modules within design process **1610** to simulate or otherwise functionally represent an electronic component, circuit, electronic or logic module, apparatus, device, or system. As such, design structure **1620** may comprise files or other data structures including human and/or machine-readable source code, compiled structures, and computer executable code structure that when processed by a design or simulation data processing system, functionally simulate or otherwise represent circuits or other levels of hardware logic design. Such data structures may include hardware-description language (HDL) design entities or other data structures conforming to and/or compatible with lower-level HDL design languages, such as Verilog and VHDL, and/or higher level design languages, such as C or C++.

[0109] Design process **1610** preferably employs and incorporates hardware and/or software modules for synthesizing, translating, or otherwise processing a design/simulation functional equivalent of components, circuits, devices, or logic structures to generate a Netlist **1680**, which may contain design structures such as design structure **1620**. Netlist **1680** may comprise, for example, compiled or otherwise processed data structures representing a list of wires, discrete components, logic gates, control circuits, I/O devices, modules, etc., that describes the connections to other elements and circuits in an integrated circuit design. Netlist **1680** may be recorded on a machine-readable data storage medium or programmed into a programmable gate array, a compact flash, or other flash memory. Additionally, or in the alternative, the medium may be a system or cache memory, buffer space, or other suitable memory.

[0110] Design process **1610** may include hardware and software modules for processing a variety of input data structure system, including Netlist **1680**. Such data structure types may reside, for example, within library elements **1630** and include a set of commonly used elements, circuits, and devices, including models, layouts, and symbolic representations, for a given manufacturing technology (e.g., different technology nodes, 32 nm, 45 nm, 90 nm, etc.). The data structure types may further include design specifications **1640**, characterization data **1650**, verification data **1660**, design rules **1670**, and test data files **1685**, which may include input test patterns, output test results, and other testing information. Design process **1610** may further include, for example, standard mechanical design processes

such as stress analysis, thermal analysis, mechanical event simulation, process simulation for operations such as casting, molding, and die press forming, etc. One of ordinary skill in the art of mechanical design can appreciate the extent of possible mechanical design tools and applications used in design process **1610** without deviating from the scope and spirit of the invention. Design process **1610** may also include modules for performing standard circuit design processes such as timing analysis, verification, design rule checking, place and route operations, etc. Improved placement can be performed as described herein.

[0111] Design process **1610** employs and incorporates logic and physical design tools such as HDL compilers and simulation model build tools to process design structure **1620** together with some or all of the depicted supporting data structures along with any additional mechanical design or data (if applicable), to generate a second design structure **1690**. Design structure **1690** resides on a storage medium or programmable gate array in a data format used for the exchange of data of mechanical devices and structures (e.g. information stored in an IGES, DXF, Parasolid XT, JT, DRG, or any other suitable format for storing or rendering such mechanical design structures). Similar to design structure **1620**, design structure **1690** preferably comprises one or 10 more files, data structures, or other computer-encoded data or instructions that reside on data storage media and that when processed by an ECAD system generate a logically or otherwise functionally equivalent form of one or more IC designs or the like. In one embodiment, design structure **1690** may comprise a compiled, executable HDL simulation model that functionally simulates the devices to be analyzed.

[0112] Design structure **1690** may also employ a data format used for the exchange of layout data of integrated circuits. and/or symbolic data format (e.g. information stored in a GDSII (GDS2), GL1, OASIS, map files, or any other suitable format for storing such design data structures). Design structure **1690** may comprise information such as, for example, symbolic data, map files, test data files, design content files, manufacturing data, layout parameters, wires, levels of metal, vias, shapes, data for routing through the manufacturing line, and any other data required by a manufacturer or other designer/developer to produce a device or structure as described herein (e.g., .lib files). Design structure **1690** may then proceed to a stage **1695** where, for example, design structure **1690**: proceeds to tape-out, is released to manufacturing, is released to a mask house, is sent to another design house, is sent back to the customer, etc.

[0113] The terminology used herein is for the purpose of describing particular embodiments only and is not intended to be limiting of the disclosure. As used herein, the singular forms "a", "and" and "the" are intended to include the plural forms as well, unless the context clearly indicates otherwise. It will be further understood that the terms "comprise" (and any form of comprise, such as "comprises" and "comprising"), "have" (and any form of have, such as "has" and "having"), "include" (and any form of include, such as "includes" and "including"), and "contain" (and any form contain, such as "contains" and "containing") are open-ended linking verbs. As a result, a method or device that "comprises", "has", "includes" or "contains" one or more steps or elements possesses those one or more steps or elements, but is not limited to possessing only those one or more steps or elements. Likewise, a step of a method or an element of a device that "comprises", "has", "includes" or "contains" one or more features possesses those one or more features, but is not limited to possessing only those one or more features. Furthermore, a device or structure that is configured in a certain way is configured in at least that way, but may also be configured in ways that are not listed.

[0114] The corresponding structures, materials, acts, and equivalents of all means or step plus function elements in the claims below, if any, are intended to include any structure, material, or act for performing the function in combination with other claimed elements as specifically claimed. The description of one or more embodiments has been presented for purposes of illustration and description but is not intended to be exhaustive or limited to in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art. The embodiment was chosen and described in order to best explain various aspects and the practical application, and to enable others of ordinary skill in the art to understand various embodiments with various modifications as are suited to the particular use contemplated.

What is claimed is:

1. A computer-implemented method comprising:

performing, by at least one processor set, signal electromigration analysis on a logic circuit design for an integrated circuit, the performing comprising:

producing a state-analysis-based transition matrix using state analysis data for the logic circuit design;

defining transition submatrices by partitioning the state-analysis-based transition matrix for the logic circuit design;

determining per simulation weighting factors for the signal electromigration analysis using the transition submatrices and switching factor data;

applying the determined per simulation weighting factors to weight, for signal electromigration analysis, electrical current data collected from a plurality of timing simulations of a transient simulator for the logic circuit design;

using the weighted electrical current data in assessing signal electromigration impact on circuit performance of the logic circuit design; and

based on the assessing of the impact on circuit performance, facilitating generating a modified logic circuit design through modifying of the logic circuit design.

2. The computer-implemented method of claim **1**, wherein defining the transition submatrices comprises defining the transition submatrices per output of the logic circuit design.

3. The computer-implemented method of claim **1**, wherein defining the transition submatrices comprises defining the transition submatrices per timing arc of the logic circuit design.

4. The computer-implemented method of claim **1**, wherein defining the transition submatrices comprises:

defining one transition submatrices type from one partitioning of the transition matrix; and

defining another transition submatrices type from another partitioning of the transition matrix, wherein the per simulation weighting factors are determined using at least one submatrices type of the one transition submatrices type and the other transition submatrices type.

5. The computer-implemented method of claim **4**, wherein the one transition submatrices type comprises tran-

sition submatrices per output of the logic circuit design, and the other transition submatrices type comprises transition submatrices per timing arc of the logic circuit design.

6. The computer-implemented method of claim **5**, wherein determining the per simulation weighting factors for the electromigration analysis further comprises:

ascertaining one set of per simulation weighting factors using the transition submatrices per output of the logic circuit design; and

based on there being no zeroing of output switching factors associated with use of the transition submatrices per output, using the ascertained one set of per simulation weighting factors as the determined per simulation weighting factors for the signal electromigration analysis.

7. The computer-implemented method of claim **6**, wherein, based on there being a zeroing of one or more output switching factors employing the transition submatrices per output, the determining of the per simulation weighting factors for the electromigration analysis further comprises:

ascertaining another set of per simulation weighting factors using the transition submatrices per timing arc of the logic circuit design; and

based on there being no singularity associated with the transition submatrices per timing arc of the logic circuit design, using the ascertained other set of per simulation weighting factors as the determined per simulation weighting factors for the signal electromigration analysis.

8. The computer-implemented method of claim **7**, wherein based on there being one or more singularities associated with the transition submatrices per timing arc of the logic circuit design, using the ascertained one set of per simulation weighting factors as the determined per simulation weighting factors for the signal electromigration analysis.

9. The computer-implemented method of claim **1**, further comprising determining, based on the state-analysis-based transition matrix comprising one or more multi-state transition simulations, the per simulation weighting factors for the signal electromigration analysis using the full state-analysis-based transition matrix rather than the transition submatrices.

10. The computer-implemented method of claim **1**, wherein using the weighted electrical current data in assessing signal electromigration impact occurs within a transistor-level static timing context.

11. A computer program product comprising:

a set of one or more computer-readable storage media; and

program instructions, collectively stored in the set of one or more storage media, for causing at least one processor set to perform computer operations comprising:

performing, by at least one processor set, signal electromigration analysis on a logic circuit design for an integrated circuit, the performing comprising:

producing state-analysis-based transition matrix using state analysis data for the logic circuit design;

defining transition submatrices by partitioning the state-analysis-based transition matrix for the logic circuit design;

determining per simulation weighting factors for the signal electromigration analysis using the transition submatrices and switching factor data;

applying the determined per simulation weighting factors to weight, for signal electromigration analysis, electrical current data collected from a plurality of timing simulations of a transient simulator for the logic circuit design;

using the weighted electrical current data in assessing signal electromigration impact on circuit performance of the logic circuit design; and

based on the assessing of the impact on circuit performance, facilitating generating a modified logic circuit design through modifying of the logic circuit design.

12. The computer program product of claim **11**, wherein defining the transition submatrices comprises defining the transition submatrices per output of the logic circuit design.

13. The computer program product of claim **11**, wherein defining the transition submatrices comprises defining the transition submatrices per timing arc of the logic circuit design.

14. The computer program product of claim **11**, wherein defining the transition submatrices comprises:

defining one transition submatrices type from one partitioning of the transition matrix; and

defining another transition submatrices type from another partitioning of the transition matrix, wherein the per simulation weighting factors are determined using at least one submatrices type of the one transition submatrices type and the other transition submatrices type.

15. The computer program product of claim **14**, wherein the one transition submatrices type comprises transition submatrices per output of the logic circuit design, and the other transition submatrices type comprises transition submatrices per timing arc of the logic circuit design.

16. The computer program product of claim **15**, wherein determining the per simulation weighting factors for the electromigration analysis further comprises:

ascertaining one set of per simulation weighting factors using the transition submatrices per output of the logic circuit design; and

based on there being no zeroing of output switching factors associated with use of the transition submatrices per output, using the ascertained one set of per simulation weighting factors as the determined per simulation weighting factors for the signal electromigration analysis.

17. The computer program product of claim **16**, wherein, based on there being a zeroing of one or more output switching factors employing the transition submatrices per output, the determining of the per simulation weighting factors for the electromigration analysis further comprises:

ascertaining another set of per simulation weighting factors using the transition submatrices per timing arc of the logic circuit design; and

based on there being no singularity associated with use of the transition submatrices per timing arc of the logic circuit design, using the ascertained other set of per simulation weighting factors as the determined per simulation weighting factors for the signal electromigration analysis.

18. A computer system comprising:

at least one processor set;

a set of one or more computer-readable storage media; and

programing instructions, collectively stored in the set of one or more storage media, for causing the at least one processor set to perform computer operations comprising:

performing, by at least one processor set, signal electromigration analysis on a logic circuit design for an integrated circuit, the performing comprising:

producing a state-analysis-based transition matrix using state analysis data for the logic circuit design;

defining transition submatrices by partitioning the state-analysis-based transition matrix for the logic circuit design;

determining per simulation weighting factors for the signal electromigration analysis using the transition submatrices and switching factor data;

applying the determined per simulation weighting factors to weight, for signal electromigration analysis, electrical current data collected from a plurality of timing simulations of a transient simulator for the logic circuit design;

using the weighted electrical current data in assessing signal electromigration impact on circuit performance of the logic circuit design; and

based on the assessing of the impact on circuit performance, facilitating generating a modified logic circuit design through modifying of the logic circuit design.

19. The computer program product of claim **18**, wherein defining the transition submatrices comprises defining the transition submatrices per output of the logic circuit design.

20. The computer program product of claim **18**, wherein defining the transition submatrices comprises defining the transition submatrices per timing arc of the logic circuit design.

* * * * *