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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
14 June 2007 (14.06.2007)

PCT

(10) International Publication Number
WO 2007/066321 A1

(51) International Patent Classification:
G06F 17/50 (2006.01)

(21) International Application Number:
PCT/IL2006/001350

(22) International Filing Date:
23 November 2006 (23.11.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/748,957 8 December 2005 (08.12.2005) US

(71) Applicant *(for all designated States except US)*: **MENTOR GRAPHICS CORPORATION** [US/US]; 8005 SW Boeckman Road, Wilsonville, Oregon 97070-9733 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **VELLER, Yossi** [IL/IL]; 9/a HaHagana Street, 46325 Herzliya (IL). **HANGA, Vasile** [IL/IL]; 3 Havradim Street, 42651 Netanya (IL). **ROZENMAN, Alexander** [IL/IL]; 13/10 Smilansky Street, 75258 Rishon Lezion (IL). **RACHAMIM, Rami** [IL/IL]; 12 Dultzin Street, 69630 Tel Aviv (IL).

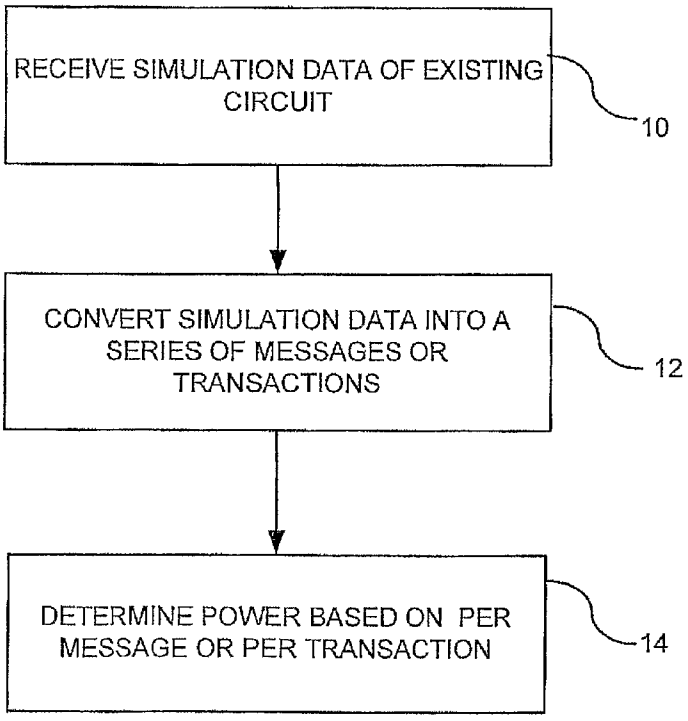
(74) Agents: **LUZZATTO, Kfir** et al.; Luzzatto & Luzzatto, Box 5352, 84152 Beersheva (IL).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM),

[Continued on next page]

(54) Title: TRANSACTION-BASED POWER MODEL IN CIRCUIT DESIGNS



(57) Abstract: A system and method is described for generating a power model of a circuit from a lower level description, such as Gate-level or RTL. In one aspect, simulation data is converted into a series of messages or transactions. Power is then determined on a per message or per transaction basis. In another aspect, an abstract power model is generated using a neural network. The neural network generates a system of weighted equations representing an accurate power model.

WO 2007/066321 A1

European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

TRANSACTION-BASED POWER MODEL IN CIRCUIT DESIGNS

Related application data

Priority is claimed to US provisional patent application 60/748,957, filed December 8, 2005, which is hereby incorporated by reference.

Field of the Invention

The present invention generally relates to simulation, and more particularly to modelling power of a simulated circuit.

Background of the Invention

The complexity of integrated circuits (ICs) being designed nowadays is continuously increasing and has resulted in complete system-on-chip (SoC) solutions. Even more, the complexity of such integrated systems is exploding thanks to advances in process fabrication. The limiting factor is now the ability to design, manage and verify such systems rather than the ability to fabricate them.

The typical design process begins with a software program that describes the behavior or functionality of a circuit. This software program is written in a hardware description language (HDL) that defines a behavior to be performed with limited implementation details. Logic synthesis tools convert the HDL program into a gate netlist description. The RTL description is used to verify functionality and ultimately generate a netlist that includes a list of components in the circuit and the interconnections between the components. This netlist is used to create the physical integrated circuit.

As SoC's are becoming larger, the only way to efficiently design such dense SoC's, both from the design complexity and time-to-market aspects, is by embedding Intellectual Property (IP) cores. Standards for

such cores are currently evolving. Ideally, they should be reusable, pre-characterized and pre-verified. But it is often desirable to change the design to create the next generation. For example, as fabrication technology changes, it is desirable to convert or migrate the design to the new process parameters. For example, an IP core may be designed and tested for 90 nm technology, but it is desirable to convert the IP core to a new process of 60 nm technology. Or it may be desirable to update the design and incorporate changes in order to create the next generation design.

In order to test and explore such changes in the design, simulation must be performed, which is very time consuming. A few seconds of real-time simulation can take weeks or even months. If the simulation results are not desirable, then the design process must start over again by changing the high-level code and re-simulating.

Because of such delays in simulation, designers are beginning to move the design process to a higher level of abstraction (meaning less focus on design details). At the higher level of abstraction, design exploration can be performed to evaluate which performance can be achieved, which parts to use, etc. The preferable higher level of abstraction is called Transaction Level Modeling (TLM), which refers to the evolving design and verification space called Electronic System Level (ESL) with methodologies that begin at a higher level of abstraction than the current mainstream Register Transfer Level (RTL). The main ESL design language SystemC, is driven from C/C++ rather than from hardware languages like Verilog and VHDL.

Power consumption is another important aspect of design exploration. There are three main power ingredients: static, dynamic, and internal.

There are a variety of techniques currently available in order to analyze power. One technique is to estimate power statistically based on gate count and technology parameters. Another technique is to estimate power based on average activity extracted from simulation. More accurate techniques for power determinations are based on netlist and place and route manipulations. In any event, there are two main power tool types on the market today: physical power solutions and ESL power solutions. Physical power solutions generally reduce power through netlist and physical manipulations. Such solutions focus on gate and interconnect levels and lack any functional or architecture-level perspective to support power management at the system level. Some ESL power solutions focus on software and algorithmic optimizations. There are some techniques that statistically characterize the power consumption of system tasks. However, the drawback is that they often assume statically scheduled systems and do not account for dynamic effects and are not accurate as data power is estimated. Other techniques are based on system simulation, with low-level power models of individual components, memories and system busses. Such techniques are computationally expensive making them unsuitable for system-level design space exploration.

Summary of the Invention

A system and method is disclosed for generating a high-level power model of a circuit from a lower level description, such as RTL or HDL.

In one aspect, simulation data is converted into a series of messages or transactions. Power is then determined on per message or per transaction basis.

In another aspect, an abstract power model is generated using a neural

network. The neural network generates a system of weighted equations. Input patterns are used to calculate a difference between the actual output values (using the equations) and desired values (using simulated data). The weights of the equations can then be modified to obtain an accurate power model.

These features and others of the described embodiments will be more readily apparent from the following detailed description, which proceeds with reference to the accompanying drawings.

Brief Description of the drawings

Figure 1 is a high-level flowchart of a method for determining power of a circuit based on a per transaction or per message basis;

Figure 2 is a more detailed flowchart of a method for generating a power model of a circuit;

Figure 3 is a hardware diagram of a system used to convert the description of the circuit into transactions;

Figure 4 is a detailed example showing simulated signal data of the circuit description on numerous hardware lines;

Figure 5 is a detailed example of a state machine for some of the available transactions;

Figure 6 is a detailed flowchart of a method for converting the simulated circuit description into transactions;

Figure 7 is a flowchart of a method for converting a series of transactions into a super-transaction representation;

Figure 8 shows a transaction-based view of the simulation data that may be displayed to the user;

Figure 9 is a flowchart of a method for performing power extraction;

Figure 10 is a flowchart of a method for simulating a netlist during the power extraction of Figure 9;

Figure 11 is a flowchart of a method performed by a neural network for generating a system of equations approximating the power of a circuit;

Figure 12 is a hardware diagram of a system for generating the power model;

Figure 13 shows a network that may be used to implement the system and method; and

Figure 14 is an exemplary flowchart of a method for implementation over the network of Figure 13.

Detailed Description of preferred embodiments

Figure 1 shows a high-level flowchart for generating a power model of a circuit. In process box 10, simulation data is received. The simulated data (e.g., in VCD format (Value Change Dump)) is a result of simulation of the circuit that is being modeled. Any desired simulator may be used, such as ModelSim®, available from Mentor Graphics Corporation. In process box 12, the simulation data is converted into a series of messages or transactions. This conversion process is explained in detail in Figures 3-8, but basically the system maps signal patterns into messages using pre-defined protocols (e.g., AMBA, PCI, etc.). Then the messages are converted to transactions (e.g., READ). Either the message or transaction level may be used in determining power. Finally, in process box 14, power of the circuit is determined on a per message or per transaction basis. As described further below, the power determined may be used to create a power model of the circuit. The power model is described further in Figures 9-12, but generally the power model is at a high level of abstraction, such as a transaction level model (TLM). The low-level description generally includes details at the

signal level, while the TLM uses high-level functions and equations to calculate power of messages or transactions based on inputs and is not concerned with the device-level description of the circuit.

Figure 2 provides more detail of the overall flow and is a useful to understand the organization of the other figures. In process box 20, the simulation data is received, similar to that already described in Figure 1. In process box 22, a series of messages or transactions are generated (Figures 3-8). In process box 24, power extraction is performed wherein particular messages are input and traced through gates in the circuit so that power can be determined on a per message or per transaction basis. The power extraction is further described in Figures 9 and 10. In process box 26, a power model is learned. The learning is described further in Figure 11, but generally "learning" is a standard term used in the industry, especially relating to neural networks. For example, an article entitled "Conditional Distribution Learning with Neural Networks", IEEE Signal Processing 1997, written by Tulay Hadah, Xiao Liu, and Kemal Sonmer describes some aspects of "learning" using neural networks. Finally, in process box 28, the power model is output at a higher level of abstraction.

Figure 3 shows a hardware diagram of a system 38 for converting a circuit description to a transaction level. A storage device 40 of any desired type has stored thereon the circuit design in HDL or any other desired language (e.g., RTL) that may be used to describe circuits. The low-level description generally includes details at the signal level.

A compiler 42 compiles the design and the protocol library 44. The compiler 42 may be any desired compiler and is usually included as part of a simulator package. The protocol library 44 includes messages and

transactions associated with a protocol used by the circuit. A sequence of messages form a transaction. Examples of messages include a request and an acknowledge of the bus, whereas a transaction is a complete operation, such as any of a variety of types of Read or Write transactions or control or setup transactions. A simulation kernel 46 simulates the compiled design in a well-known manner, as already described. The simulation kernel 46 outputs the simulation data 48 in any desired format. Box 48 can also represent a pre-simulated design data (VCD format).

A message recognition module 50 reads the simulation data 48 and analyzes the data to convert it to messages of the protocol stored in the protocol library 44. Figures 4-6 describe this conversion more thoroughly, but generally switching signals of the simulation are compared (during various time slices) to messages within the protocol library 44 to determine what message is being processed during a particular time slice. The messages associated with the switching signals during each time slice are then stored to convert the switching signals into messages.

A transaction recognition module 52 reads the messages determined by the message recognition module 50 and converts the messages into transactions using a comparison of a series of messages to predetermined messages within the protocol library 44. If a match is found, then the transaction recognition module stores the series of messages as a transaction. The result is that the messages are converted into a series of transactions.

A transaction sequence recognition module 54 converts multiple transactions into a single super-transaction sequence. For example,

several Writes can be converted into a single control operation. This conversion from multiple transactions to a super-transaction sequence is described further below in relation to Figure 7. If desired, the transaction sequence recognition module 54 may be bypassed or omitted, so that the transactions are output directly. Results 56 of the conversion are output onto a storage medium or a display.

In any event, the simulated circuit description is taken to a higher level of abstraction, as the simulation data is converted first to messages, then to transactions, and finally to transaction sequences. The transaction sequences are at a higher level of abstraction.

The compiler 42, simulator kernel 46, and modules 50, 52, 54, may all be run on the same computer. Alternatively, the circuit description may be compiled and simulated in a different location so that the resultant simulation data 48 is merely on a storage medium to be input into the message recognition module 50. In such a case, as shown at 58, it is desirable that the some of the protocol data from the protocol library 44 is incorporated into the simulation data in a pre-processing step.

Figure 4 shows a detailed example of part of the simulated signal data 48. Various signal data 70 on hardware lines are shown including a clock line 72, a read/write line 74, a bus request line 76, a ready line 78, address lines 80, and data lines 82. Simulation is also carried out on many more hardware lines, which are not shown for convenience. The signals being simulated follow a predetermined protocol 84. A protocol is a set of rules or standards designed to enable circuit elements to communicate together and exchange information with as little error as possible. The protocol 84 is made up of a plurality of transactions 85, such as shown at 86 (i.e., transaction A) and at 88 (i.e., transaction B). A

transaction is a discrete activity, such as a Read or Write operation that moves data from one place to another in the system. The transactions 86, 88 are in turn made up of a series of messages 90. For example, transaction 86 is shown as including three messages, 92, 94, and 96. A message is a smaller unit of information electronically transmitted from one circuit element to another to facilitate the transaction. Example messages include "request for bus", "acknowledge", "ready", etc. Those skilled in the art will readily recognize that these are only examples of transactions and messages and others may be used. Each message is associated with a time-slice 98, such as those shown at 100, 102, and 104. Normally, the time-slices are based on the clock signal 72. During each time-slice, the hardware lines 70 are analyzed to determine the message being sent in correspondence with the transactions of the protocol, as further described below. Transaction 88 is similar to transaction 86 and need not be further described.

Figure 5 shows an example part of a state machine 120 stored within the protocol library 44. Different states 122 are shown as numbered circles. Messages, such as those at 90, are shown in boxes, and cause the state machine to move from one state to another. Transactions may be defined by a path through the state machine 120 that starts at an idle state 124 (state 0) and that ends at the same idle state, although those skilled in the art will recognize that the state machine 120 may be constructed in a variety of different formats. For example, a read transaction 126 is made up of numbered states 0, 1, 2, 3, 4 and 5. The read transaction 126 is completed upon return to the idle state from state 5 to state 0, as shown by arrow 128. A write transaction 130 is made up of numbered states 0, 1, 2, 6, 7, 8, 9, and 10. The write transaction 130 is completed upon return to the idle state from state 7 to state 0, as shown by arrow 132.

Figure 6 shows a flowchart of a method preformed by the message recognition module 50 and the transaction recognition module 52 in order to convert the simulation data into a transaction-based description. At process box 150, the simulated input data (see box 48 in Figure 3) is received so that it may be used by the message recognition module 50. Such simulation data is normally within a database. In process box 152, the analysis starts by monitoring the signal data 70 on the various hardware lines upon which messages are received. Additionally, in process box 152, the protocol library 44 is read to access a state machine, such as state machine 120, associated with the protocol. In process box 154, in order to analyze a transaction, an assumption is made that the transaction starts from the idle state 124. In process box 156, a time-slice of data is read corresponding to the clock signal on hardware line 72. For example, in Figure 4, the data may be read starting with a time-slice 100. Thus, the switching signals on the various hardware lines are read in order to be analyzed. In process box 158, the data read is analyzed by comparing the switching signals to known patterns of messages stored in the protocol library 44. Returning briefly to Figure 5, from the idle state 124, a bus request message changes the state of the state machine to state 1. A bus request message has a particular pattern of signal data on the hardware lines, which is compared to a known pattern in the protocol library 44. Thus, once a match is found between the known pattern of messages and the message analyzed during the currently analyzed time-slice, the message has been determined and is stored in process box 160. In process box 162, the current state of the state machine is updated to reflect the change of state. Continuing with the example, the new state is state 1 after a bus request message is received. In decision box 164, a determination is made whether the state machine has returned to the idle state. If yes,

this indicates that a transaction is complete and the transaction is determined in process box 166 by comparing a sequence of the stored messages to a sequence of known messages in the protocol library 44. The sequence of stored messages are those received from the start of the idle state until the state machine returned to the idle state. Once a match is found between the sequence of stored messages and those in the protocol library, the transaction associated with those messages is easily obtained from the protocol library 44. The determined transaction is then stored as indicated in process box 166. In decision box 168, a check is made whether all of the input simulated signal data has been analyzed by reading whether the database including the signal data is at the end. If yes, the method ends as shown at 170. Otherwise, the method continues at process box 156 and the next time-slice is read (e.g., time-slice 102). Once the method ends, the database of signal data is converted into a series of transactions associated with the protocol found in the protocol database 44.

Figure 7 shows a method implemented by the transaction sequence recognition module 54 (see Figure 3). It may be desirable to group transactions together in order to display to a user the circuit at an even higher level of abstraction. For example, several write / read transactions can be shown as a single control transaction as opposed to individual transactions. In process box 200, a group of transactions is selected. For example, if there are many of the same type of transactions in sequence (e.g., Reads), such a sequence may be condensed. In process box 202, the selected group is compared to predetermined groups. In decision box 204, a determination is made whether there is a match between the selected group and the predetermined groups. If there is a match, then in process box 206, the sequence of transactions is stored as a single transaction in order to

convert the circuit description to an even higher level of abstraction. In decision box 208, a check is made whether all of the transactions have been read. If yes, then the method ends at 210. If not, then a new group of transactions is chosen at 212, and the process starts over at process box 202.

Figure 8 shows an example of a display showing the simulation data of Figure 3 at a higher level of abstraction. Particularly, instead of signals, the simulation data is shown as a series of transactions. Write transactions, such as at 240, are shown as dotted lines and read transactions, such as shown at 242, are shown as solid lines. Throughput is shown along the Y-axis and time is indicated along the X-axis. Thicker lines generally mean there is a grouping of many transactions so close in time that at the current zoom level they cannot be distinguished. Of course, a zoom option may be used to focus on particular transactions. As can readily be seen, the view of Figure 8 is much easier to read than that of Figure 4 and allows the designer to obtain a better overall system view of the flow of data.

Figure 9 shows further details of the power extraction 24. In process box 260, a technology library and model description are read. The technology library includes a set of gates and the physical characteristics of the gates (e.g., voltage, current, and power consumption). A variety of technology libraries may be used, but one possibility is the Synopsys® liberty® file. The model description is provided by the user and includes an interface of the circuit model describing the input/output ports, the inner state description of the circuit model that describes the internal states thereof, and the circuit HDL Netlist. Thus, the model description includes the model interface, the HDL netlist, and the lasting state description (including registers, buffers, etc.) In process box 262, the

model netlist is converted to a netlist with special monitors inserted. Such a conversion can be accomplished by regenerating the HDL netlist or by using PLI code. In any event, the monitors are used later during simulation to help track messages as they pass through the circuit. A monitor can be simply a function call that is activated whenever a gate switches (output changes) during which the message ID and power are assigned.

In process box 264, a netlist is simulated using a simulation sequence as a testbence. For example, a VCD (Value Change Dump) sequence can be used as a testbence. The VCD sequence is a sequence of messages that are run back through the netlist including monitors in order to perform the tracing of how much power a message uses. Figure 10 explains process box 264 in further detail.

In process box 266, a power database is output with power data on a per message or per transaction basis. This power database is used in the learning process as the power data is passed through a neural network.

Figure 10 shows the simulation 264 (Figure 9) in further detail. In process box 280, each message type in each port is assigned with all the associated input signals. The simulation sequences include messages and signals associated with the messages are assigned to the input ports in order to apply the signals to the circuit. In process box 282, the input message is assigned a unique identification so that power may be tracked for that particular message. In process box 284, the signals associated with the message are applied to the circuit in simulation and as the messages propagate through the gates of the circuit, any gate that switches is marked with the message ID that caused the gate to switch (process box 286). In process box 288, power associated with the

switched gate is assigned to the particular message ID that caused the gate to switch. In process box 290, instantaneous power can be calculated at any point as the accumulation of all power per message. In particular, power of all the gates having the same ID can be summed in order to determine power per message. Power per transaction can then easily be determined by adding the power of each message forming the transaction. This information is updated into the power database.

The power model is based on accumulated dynamic power and non-active (leakage) power. The formula that can be used in calculating power is: Power=message power (load+cell) + leakage power@parameters. The load is the load switching power. The cell is the cell internal switching power (short circuit). The leakage is the non-active power. The load per gate is estimated as load=(activity)*F*C*Vdd(2), where F is the clock frequency, C is the capacitance (internal gates and external wire and fan-out) and Vdd is the supply voltage.

Figure 11 is a flowchart of a method for performing "learning" 26 (Figure 2). In process box 300, the power database is used to create a system of weighted equations that represent the power behavior of the circuit. Thus, for example, the inputs are analyzed in conjunction with power information to generate the equations so that in any given state a determination of an equation for power can be made. Such a generation of equations is well known in the art using standard techniques of neural networks. In process box 302, input power patterns are applied to the generated system of equations to generate actual values produced by the equations. In process box 304, an error is calculated by using a difference between the actual values (process box 302) to the desired values (determined during simulation). In process box 306, based on

this difference, the weightings in the system of equations are modified in order to more closely match the desired values. In decision box 308, a check is made whether the actual values generated by the system of equations are within an acceptable limit. If so, the flowchart is exited at 310. If not, the flow returns to process box 302 in order to re-analyze the equations.

Figure 12 shows the overall system diagram for generating a power model at a higher level of abstraction so that power may be determined on a per message or per transaction basis. An HDL model file 320 is a description of the circuit. A synthesis engine 322 generates a gate netlist 324. Instrumentation for inserting monitors, shown at 325, inserts the appropriate monitors into the gate netlist to be used in simulation. A technology library 326 that includes the physical characteristics of the gates within the circuit is used by the synthesis engine in order to generate a gate netlist 324. The place and route engine 328 actually performs placing and routing of the circuit components and connections between components. The place and route determination increases accuracy because once the circuit is in silicon, extra delays and capacitances usually exist. As described further below, these extra delays and capacitances are back annotated into the model to increase accuracy. A simulator (not shown) generates simulation data 340 that is passed through a protocol extraction engine 342 to generate protocol data 344 (e.g., messages, transactions). A converter 346 converts the messages or transactions back to signals for simulation in simulation kernel 348. The simulation kernel 348 simulates the gate netlist and uses the special monitors to help track how the messages propagate through the simulated circuit. The output of the simulation kernel is passed to a power extraction engine 352, which uses the simulated data, a power model template 350, protocol data 344, and the

input from the technology library 326 to calculate and generate a database of power per message or transaction. The power model template includes the model description and receives the HDL file 320. The information within the database is passed to a neural network 354, which generates power functions based on a message or transaction basis and outputs a corresponding power model 355. Those skilled in the art will recognize that the neural network can be replaced with any other machine learning or statistical algorithm.

In a place and route incremental power engine 356, the power functions generated by the neural network 354 are corrected based on back-annotated place and route information from place and route engine 328. The result is that RTL or HDL files and gate-level designs are converted to TLM equivalent models with accurate power and performance behavior. Such models can be simulated as-is to run pure performance analysis of a system, or can be plugged into TLM functional models and used to provide timing and power behavior during fully functional simulation. The power model is also hierarchical so that it can support a top-down approach where users can start with a high-level power model and gradually refine it along with implementation and the data that is available at each step downstream. One possible application includes having a power model associated with each IP core in a system. Instantaneous power of the overall system may be computed quickly. Moreover, an IP core can be removed or replaced and the affect on power may be visualized quickly.

Figure 13 shows that portions of the system may be applied to a distributed network, such as the Internet. Of course, the system also may be implemented without a network (e.g., a single computer). A server computer 400 may have an associated database 402 (internal or

external to the server computer). The server computer is coupled to a network shown generally at 404. One or more client computers, such as those shown at 408 and 410, are coupled to the network to interface with the server computer using a network protocol.

Figure 14 shows a flow diagram using the network of FIGURE 13. In process box 450, the circuit description is sent from a client computer, such as 408, to the server computer 400. In process box 452, the power of the circuit is derived, as previously described. In process box 454, the extracted power is used in a learning process in order to generate an abstract power model of the circuit. In process box 456, the results are sent though the network to the client computer 408. Finally, in process box 458, the results are displayed to the user. It should be recognized that one or more of the process boxes may be performed on the client side rather than the server side, and vice versa.

The system described maps actual power behavior up to the message or transaction levels where it can be budgeted and managed. At these levels, optimization can be traded with performance, while maintaining a high level of accuracy and speed.

Having illustrated and described the principles of the illustrated embodiments, it will be apparent to those skilled in the art that the embodiments can be modified in arrangement and detail without departing from such principles.

For example, in some systems, a full conversion to transactions is unnecessary if only messages are to be used.

In view of the many possible embodiments, it will be recognized that the illustrated embodiments include only examples of the invention and should not be taken as a limitation on the scope of the invention. Rather, the invention is defined by the following claims. We therefore claim as the invention all such embodiments that come within the scope of these claims.

Claims:

1. A method for analyzing power of a circuit design, comprising:

receiving simulation data of the circuit design;

converting the simulation data into a series of messages or transactions; and

determining power in the circuit design on a per message or per transaction basis.

2. The method of any of claim 1, further including learning the power behavior using the determined power and generating a power model of the circuit including equations representing the power used by the circuit on a per message or per transaction basis.

3. The method of any of claims 1-2, wherein converting simulation data includes analyzing two or more switching signals on simulated hardware lines;

comparing the switching signals to known patterns of messages in a protocol; and

determining a match between the switching signals and a message in the protocol.

4. The method of any of claims 1-3, wherein determining transactions includes:

comparing a sequence of messages to known transactions, which include sequences of messages of the protocol; and

determining a match between the sequence of messages obtained from the simulation data and a transaction.

5. The method of any of claims 1-4, wherein determining power includes:

extracting power by tracing how a message affects gates of the circuit; and

for each gate affected, associating the message with the power used in that gate.

6. The method of any of claims 1-5, further including calculating power in a message by adding the power of each gate affected by the message.

7. The method of any of claims 1-6, further including calculating power in a transaction by adding the power of each message comprised in the transaction.

8. The method of any of claims 1-7, further including generating an abstract model including a system of equations that represent the power on a per message or per transaction basis.

9. The method of claim 8, wherein generating an abstract model includes learning the circuit using neural networks or any other machine learning or statistical algorithm.

10. The method of claim 9, wherein learning includes:

generating a system of weighted equations representing the power of the circuit on a per message or per transaction basis;

applying input patterns to the system of equations to generate actual output values;

calculating an error by using a difference between the actual values and desired values;

modifying the weightings in the system of equations based on the calculated error.

11. The method of any of the preceding claims 8-10, wherein the circuit description is in a register transfer level and the abstract model is a transaction-level model having a different level of abstraction than the register transfer level.

12. The method of any of the preceding claims 1-11, wherein at least part of the method is performed on a client computer coupled to a network and part of the method is performed on a server computer coupled to the network.

13. A computer-readable medium including instructions stored thereon for performing the method of claim 1.

14. An apparatus to model power in a circuit, comprising:

a power extraction engine to determine power used on a per message or per transaction basis using simulation data; and

a neural network coupled to the power extraction engine and using the power determined to generate an abstract power model of the circuit.

15. The apparatus of claim 14, further including a technology library coupled to the power extraction engine, the technology library including information regarding power consumption of gates within the circuit.

16. The apparatus of any of claims 14-15, wherein part of the apparatus is located on a client computer and another part is located on a server computer.

17. The apparatus of any of claims 14-16, wherein the abstract model is in an electronic-system-level model.

18. The apparatus of any of claims 14-17, further including a place and route incremental power engine coupled to the neural network.

19. An apparatus to model power in a circuit, comprising:

means for receiving simulation data of the circuit design;

means for converting the simulation data into a series of messages or transactions; and

means for generating a power model that determines power in the circuit design on a per message or per transaction basis.

20. The apparatus of claim 19, further including means for back annotating power based on place and route information.

21. The apparatus of any of claims 19 or 20, further including simulation means coupled to the means for converting.

FIGURE 1

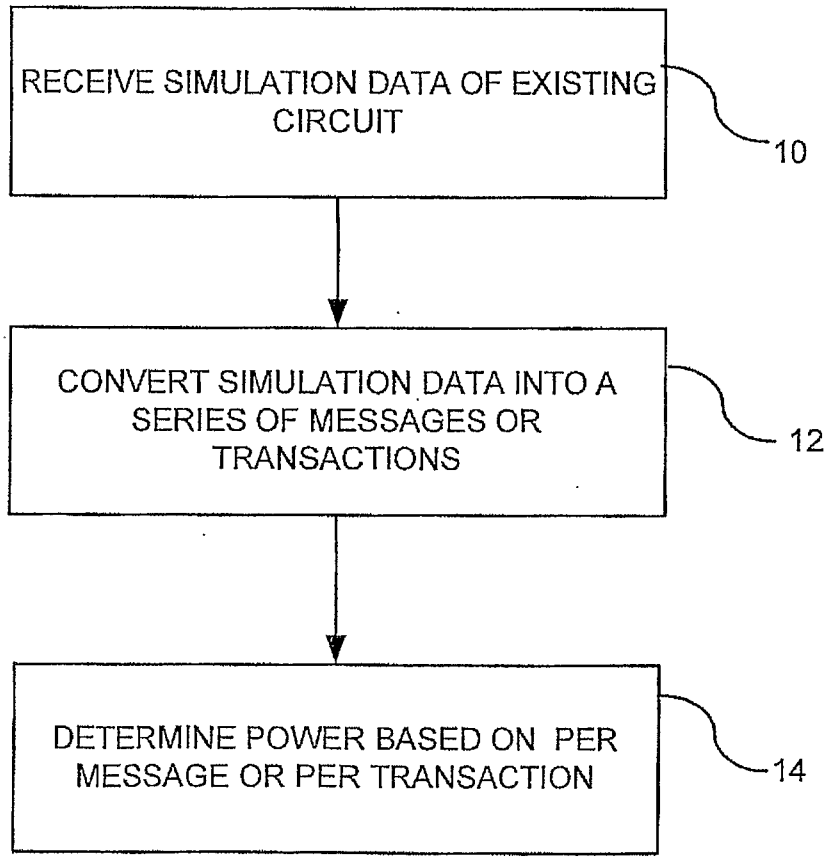

RECEIVE SIMULATION DATA OF EXISTING CIRCUIT
10
CONVERT SIMULATION DATA INTO A SERIES OF MESSAGES OR TRANSACTIONS
12
DETERMINE POWER BASED ON PER MESSAGE OR PER TRANSACTION
14

FIGURE 2

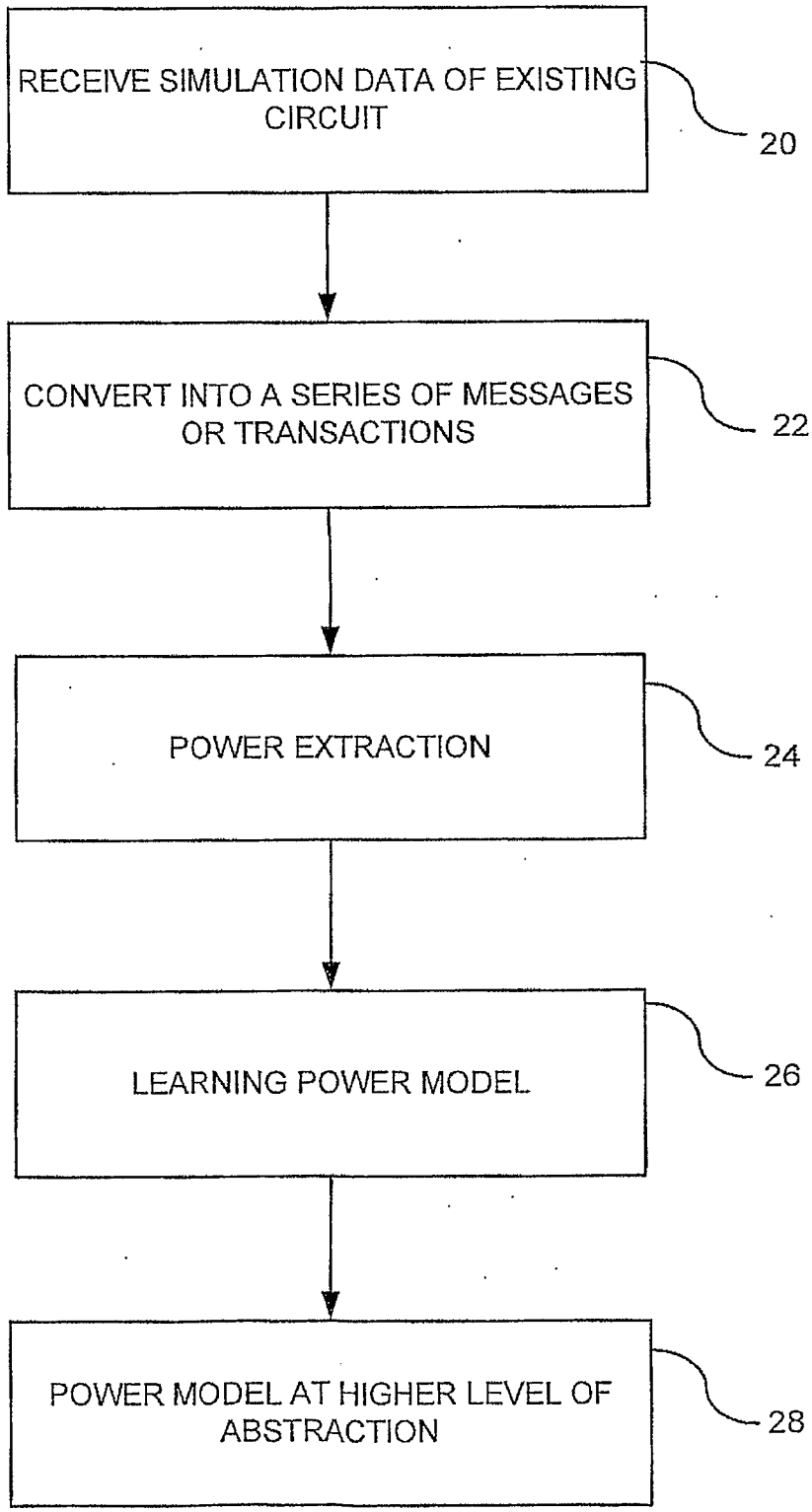
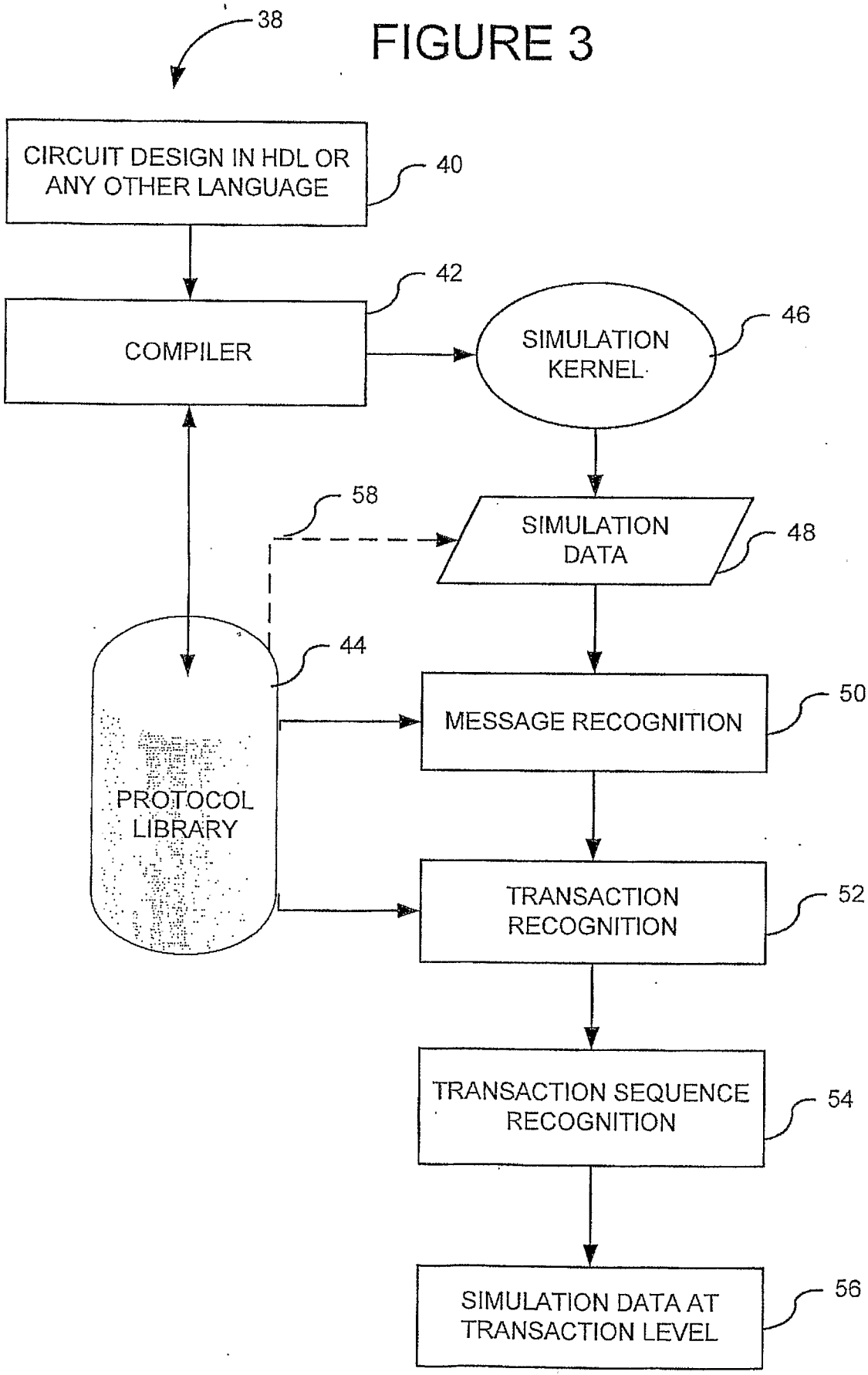

RECEIVE SIMULATION DATA OF EXISTING CIRCUIT
20
CONVERT INTO A SERIES OF MESSAGES OR TRANSACTIONS
22
POWER EXTRACTION
24
LEARNING POWER MODEL
26
POWER MODEL AT HIGHER LEVEL OF ABSTRACTION
28

FIGURE 3


38
CIRCUIT DESIGN IN HDL OR ANY OTHER LANGUAGE
40
COMPILER
42
SIMULATION KERNEL
46
SIMULATION DATA
48
58
PROTOCOL LIBRARY
44
MESSAGE RECOGNITION
50
TRANSACTION RECOGNITION
52
TRANSACTION SEQUENCE RECOGNITION
54
SIMULATION DATA AT TRANSACTION LEVEL
56

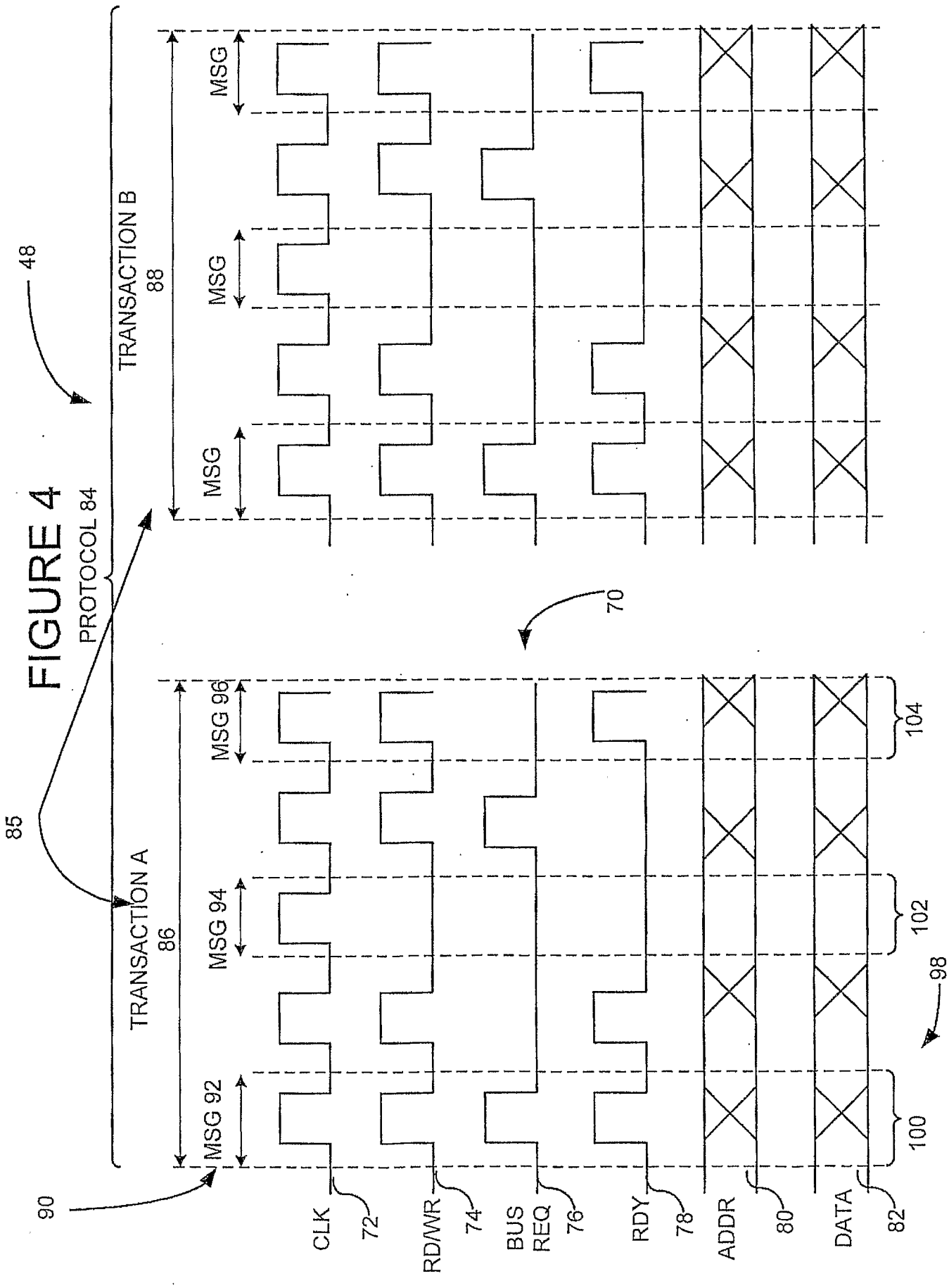


FIGURE 4
48
PROTOCOL 84
85
TRANSACTION A
86
TRANSACTION B
88
MSG 92
MSG 94
MSG 96
MSG
MSG
MSG
70
90
98
100
102
104
CLK
72
RD/WR
74
BUS REQ
76
RDY
78
ADDR
80
DATA
82

FIGURE 5

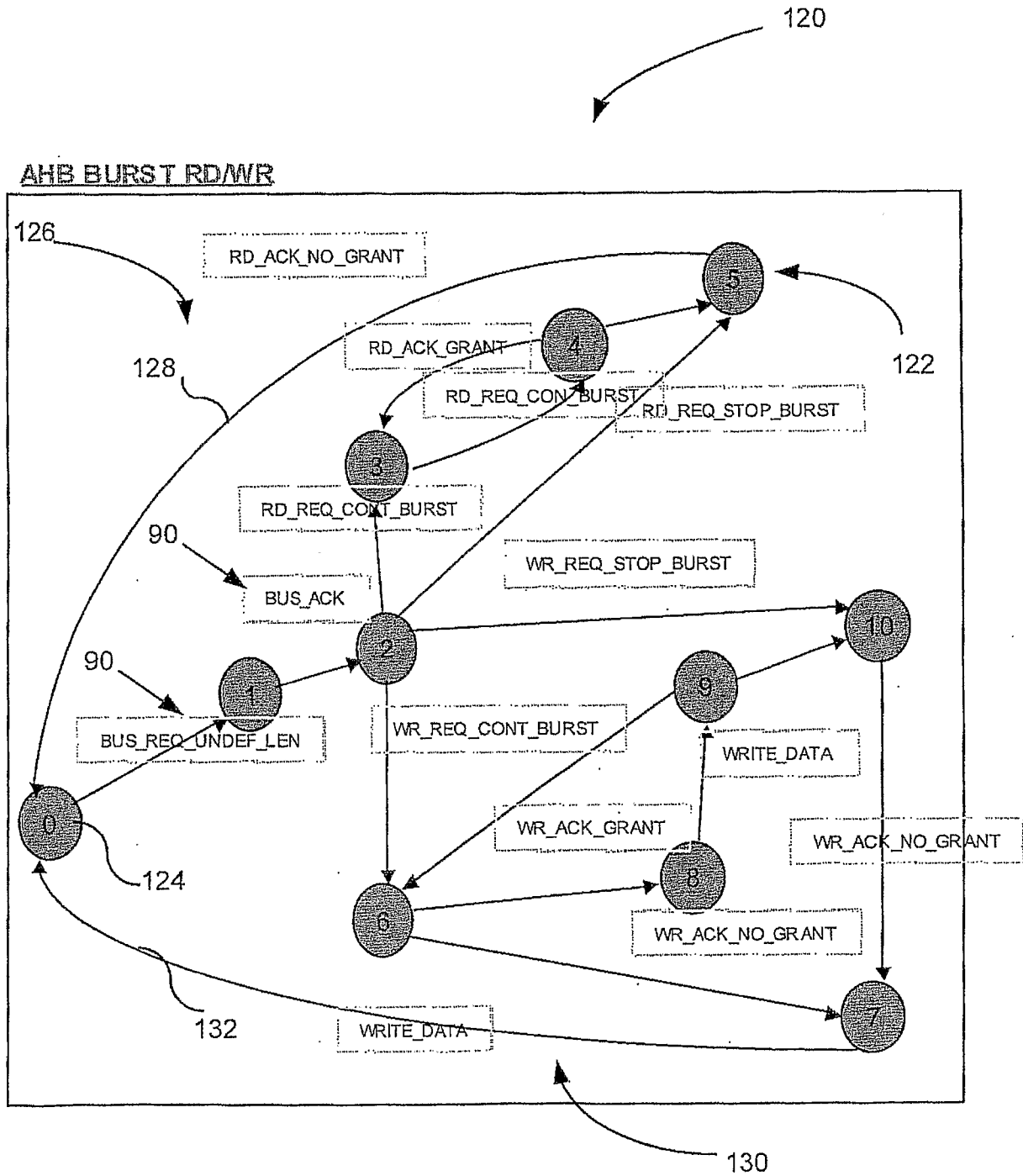

120
AHB BURST RD/WR
126
RD_ACK_NO_GRANT
5
RD_ACK_GRANT
4
128
122
RD_REQ_CONT_BURST
RD_REQ_STOP_BURST
8
RD_REQ_CONT_BURST
90
WR_REQ_STOP_BURST
BUS_ACK
10
2
90
1
9
WR_REQ_CONT_BURST
BUS_REQ_UNDEF_LEN
WRITE_DATA
0
WR_ACK_GRANT
WR_ACK_NO_GRANT
124
6
WR_ACK_NO_GRANT
7
132
WRITE_DATA
130

FIGURE 6

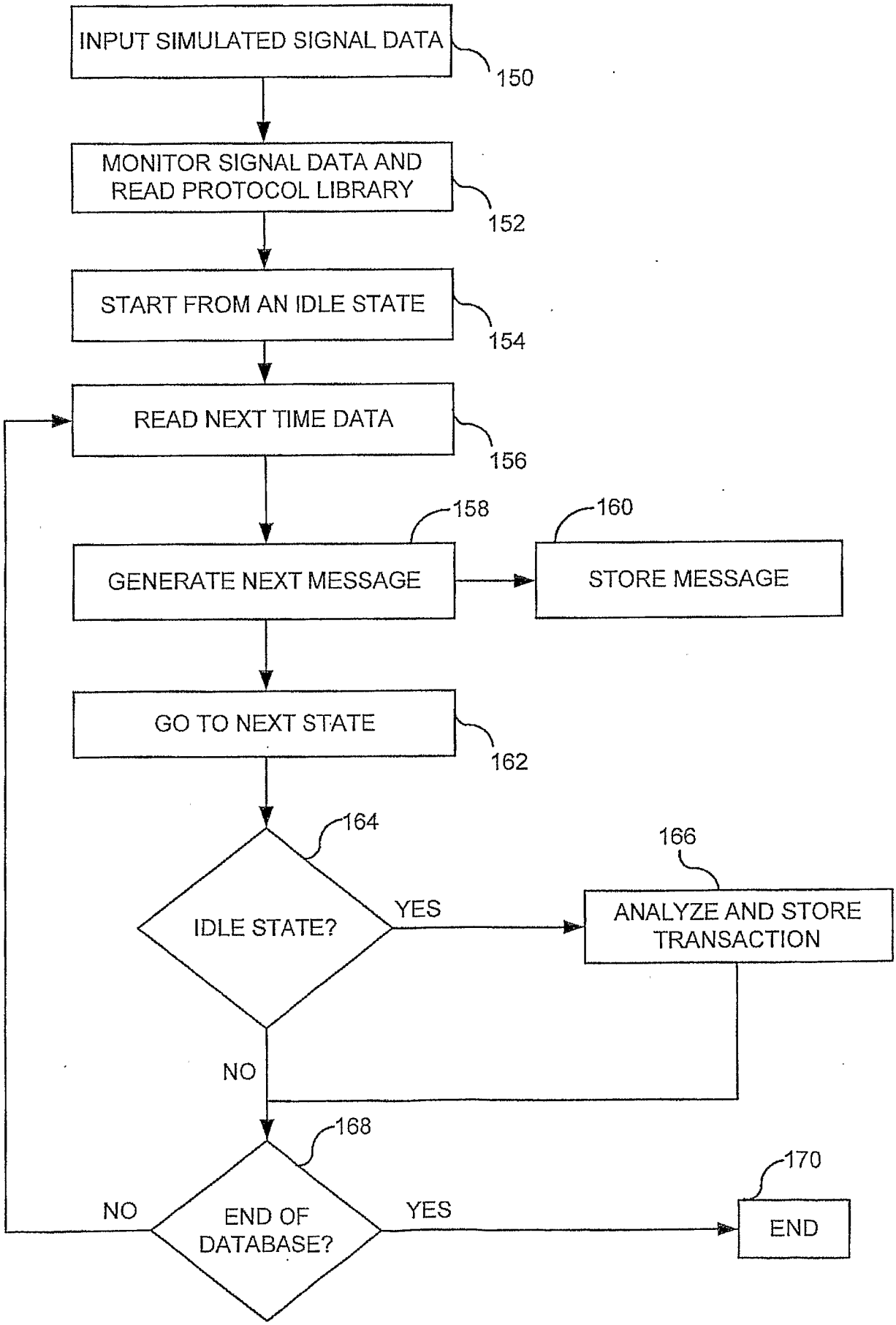

INPUT SIMULATED SIGNAL DATA
150
MONITOR SIGNAL DATA AND READ PROTOCOL LIBRARY
152
START FROM AN IDLE STATE
154
READ NEXT TIME DATA
156
158
GENERATE NEXT MESSAGE
160
STORE MESSAGE
GO TO NEXT STATE
162
164
IDLE STATE?
YES
166
ANALYZE AND STORE TRANSACTION
NO
168
END OF DATABASE?
NO
YES
170
END

FIGURE 7

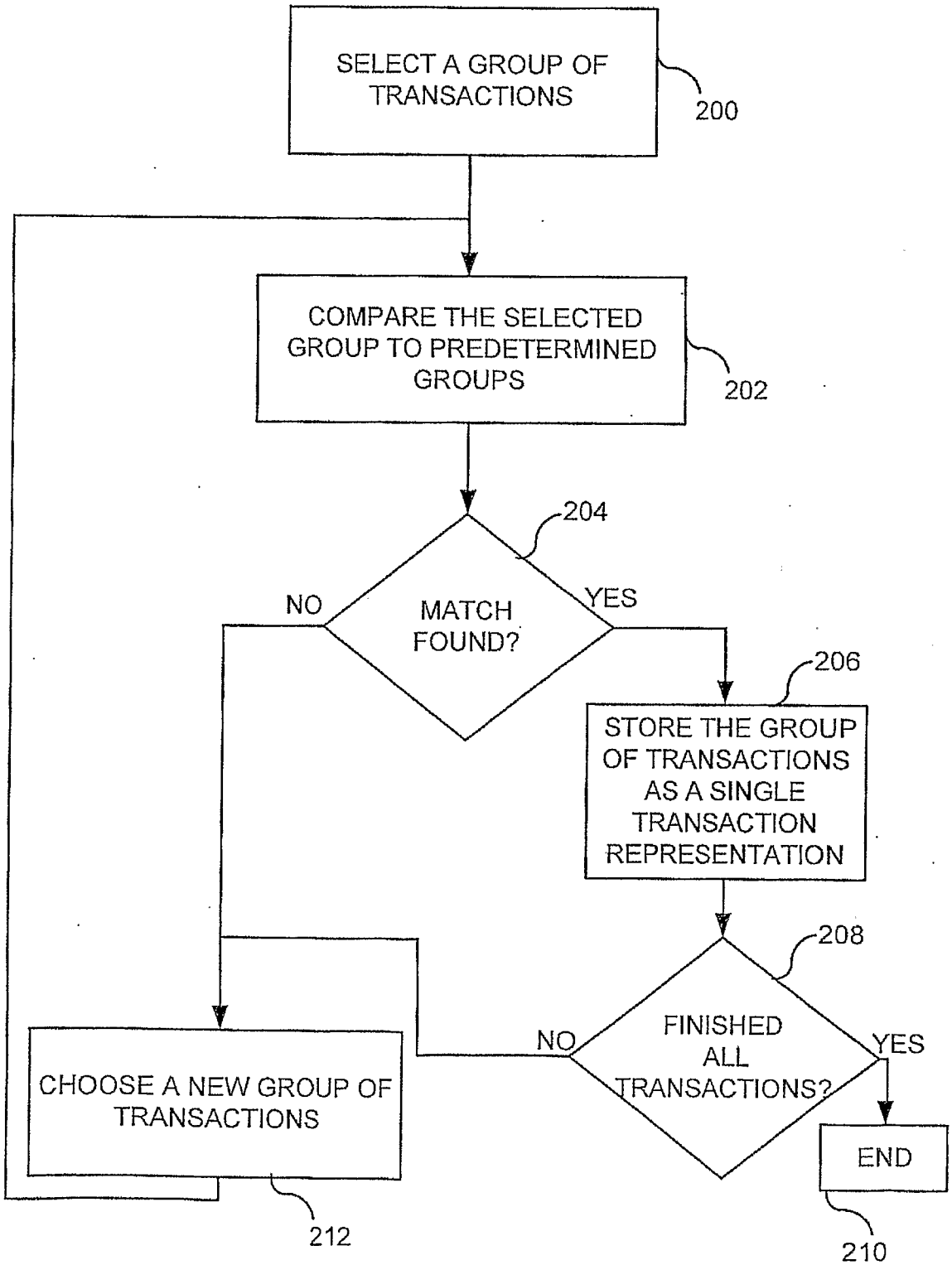

SELECT A GROUP OF TRANSACTIONS
200
COMPARE THE SELECTED GROUP TO PREDETERMINED GROUPS
202
204
MATCH FOUND?
NO
YES
206
STORE THE GROUP OF TRANSACTIONS AS A SINGLE TRANSACTION REPRESENTATION
208
FINISHED ALL TRANSACTIONS?
NO
YES
CHOOSE A NEW GROUP OF TRANSACTIONS
212
END
210

FIGURE 8

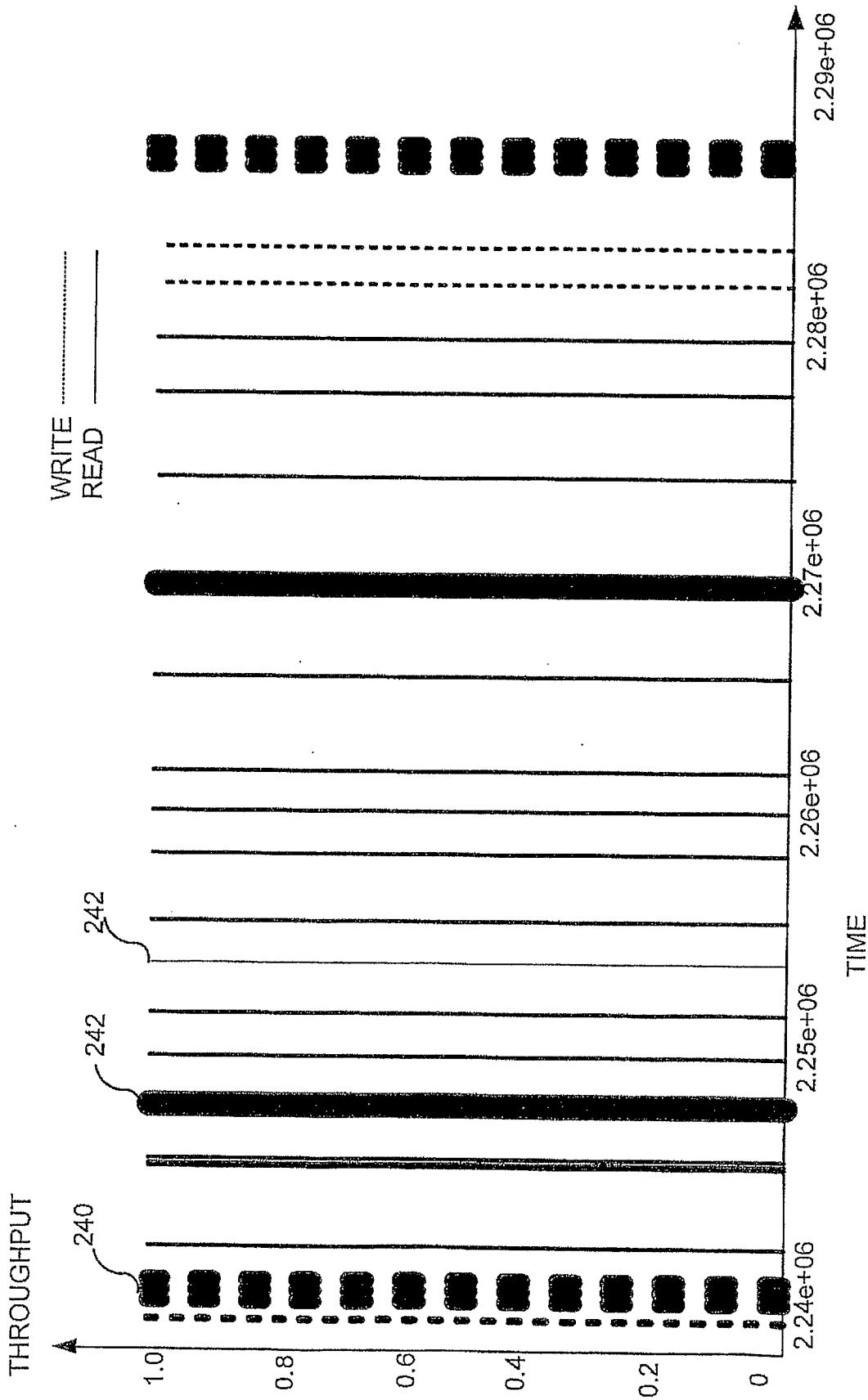

THROUGHPUT
1.0
0.8
0.6
0.4
0.2
0
240
242
242
WRITE
READ
2.24e+06
2.25e+06
2.26e+06
2.27e+06
2.28e+06
2.29e+06
TIME

FIGURE 9

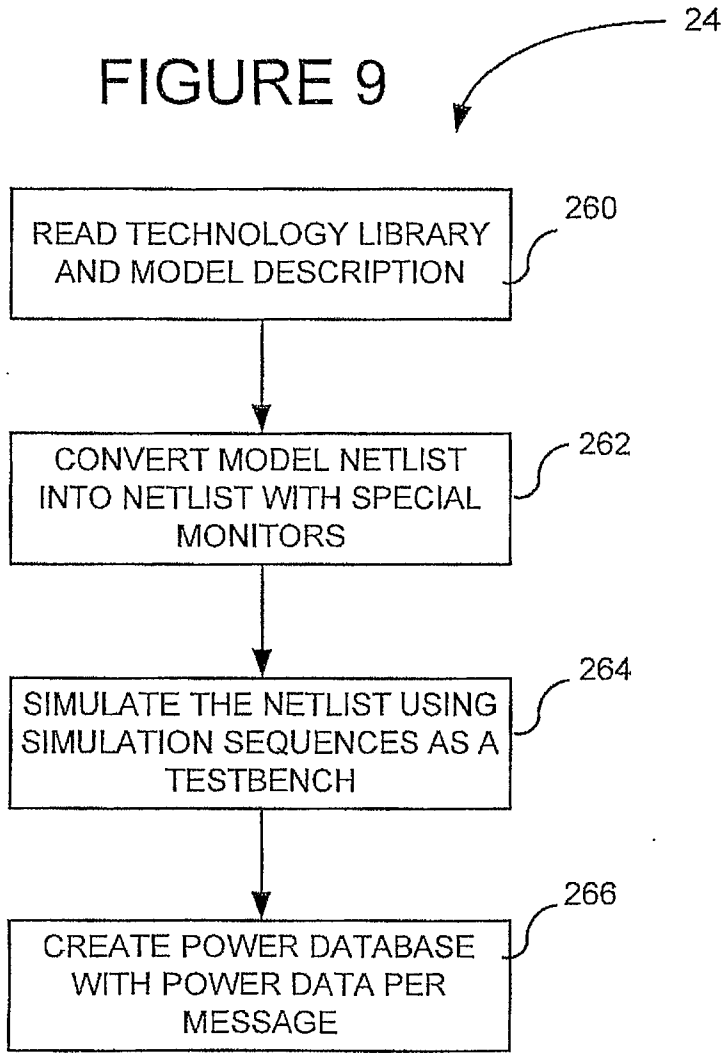

24
READ TECHNOLOGY LIBRARY AND MODEL DESCRIPTION
260
CONVERT MODEL NETLIST INTO NETLIST WITH SPECIAL MONITORS
262
SIMULATE THE NETLIST USING SIMULATION SEQUENCES AS A TESTBENCH
264
CREATE POWER DATABASE WITH POWER DATA PER MESSAGE
266

FIGURE 10

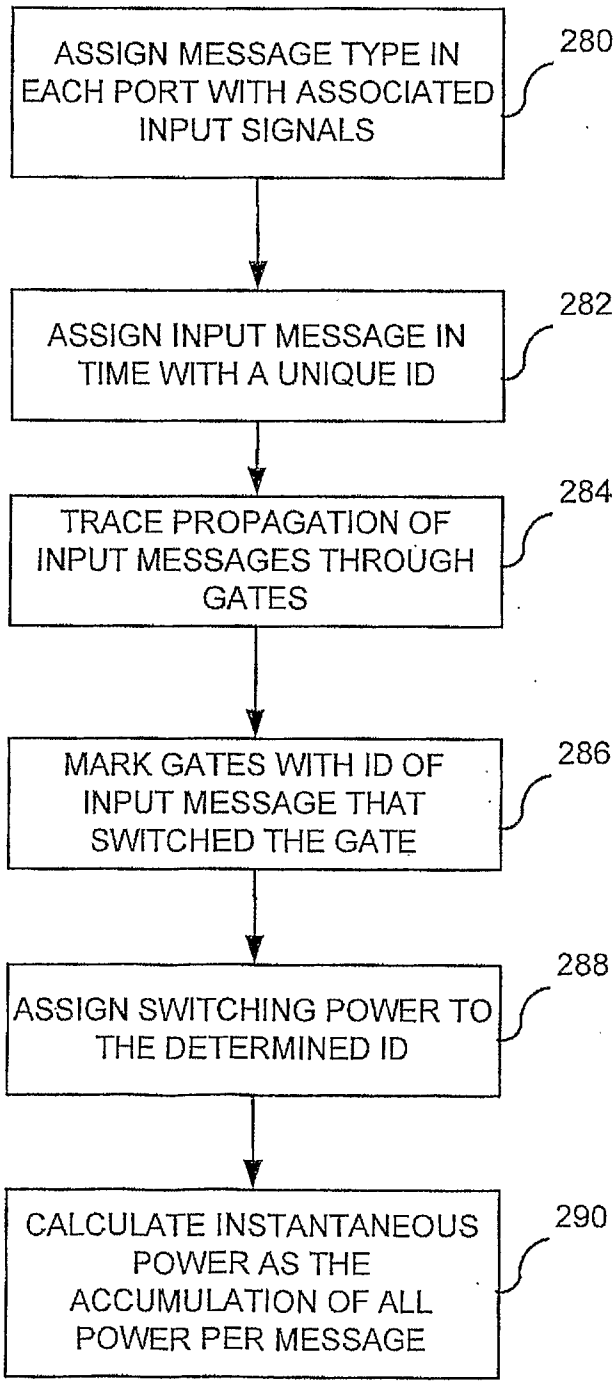

ASSIGN MESSAGE TYPE IN EACH PORT WITH ASSOCIATED INPUT SIGNALS
280
ASSIGN INPUT MESSAGE IN TIME WITH A UNIQUE ID
282
TRACE PROPAGATION OF INPUT MESSAGES THROUGH GATES
284
MARK GATES WITH ID OF INPUT MESSAGE THAT SWITCHED THE GATE
286
ASSIGN SWITCHING POWER TO THE DETERMINED ID
288
CALCULATE INSTANTANEOUS POWER AS THE ACCUMULATION OF ALL POWER PER MESSAGE
290

FIGURE 11

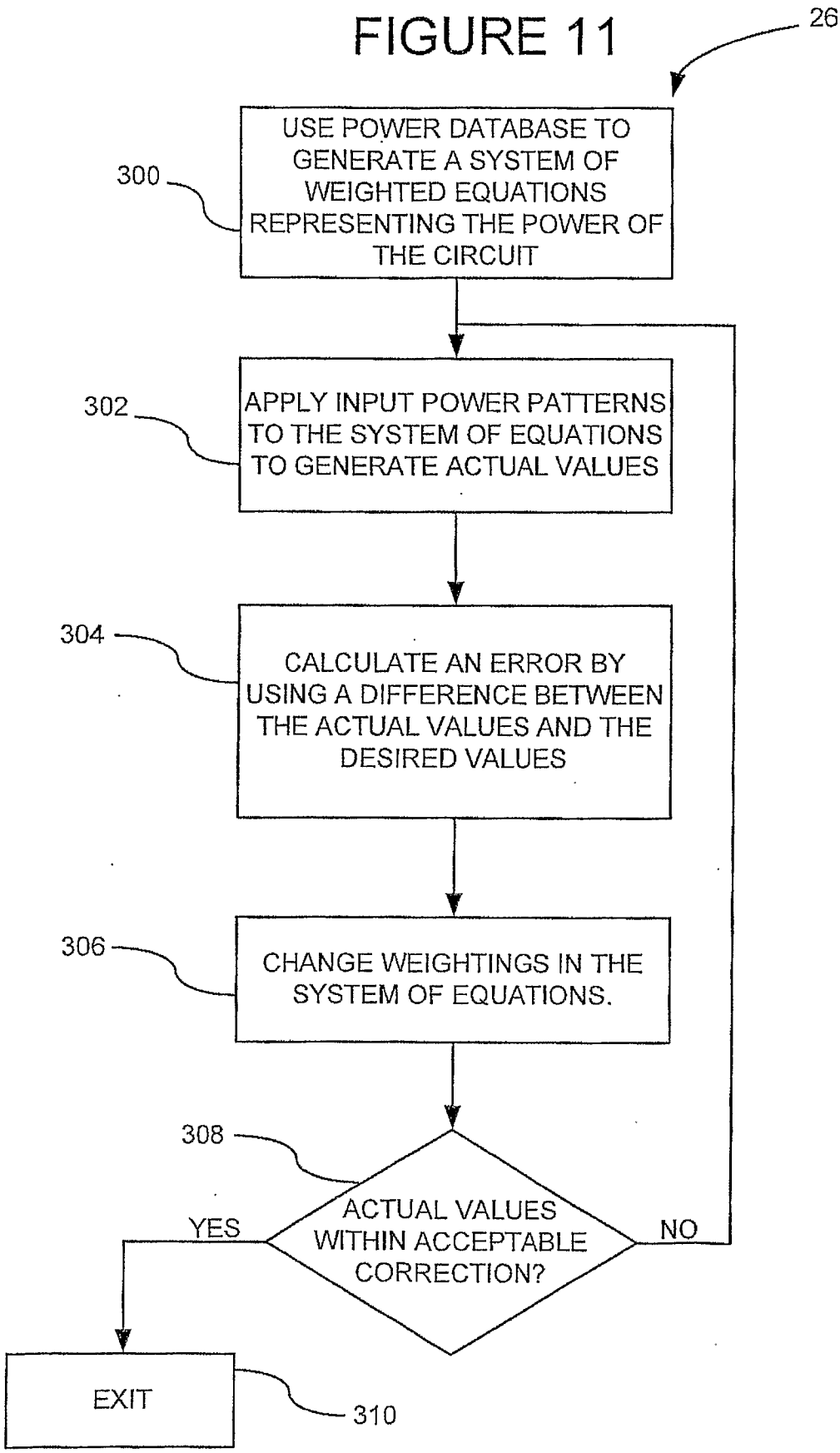

26
USE POWER DATABASE TO GENERATE A SYSTEM OF WEIGHTED EQUATIONS REPRESENTING THE POWER OF THE CIRCUIT
300
APPLY INPUT POWER PATTERNS TO THE SYSTEM OF EQUATIONS TO GENERATE ACTUAL VALUES
302
CALCULATE AN ERROR BY USING A DIFFERENCE BETWEEN THE ACTUAL VALUES AND THE DESIRED VALUES
304
CHANGE WEIGHTINGS IN THE SYSTEM OF EQUATIONS.
306
ACTUAL VALUES WITHIN ACCEPTABLE CORRECTION?
308
YES
NO
EXIT
310

FIGURE 12

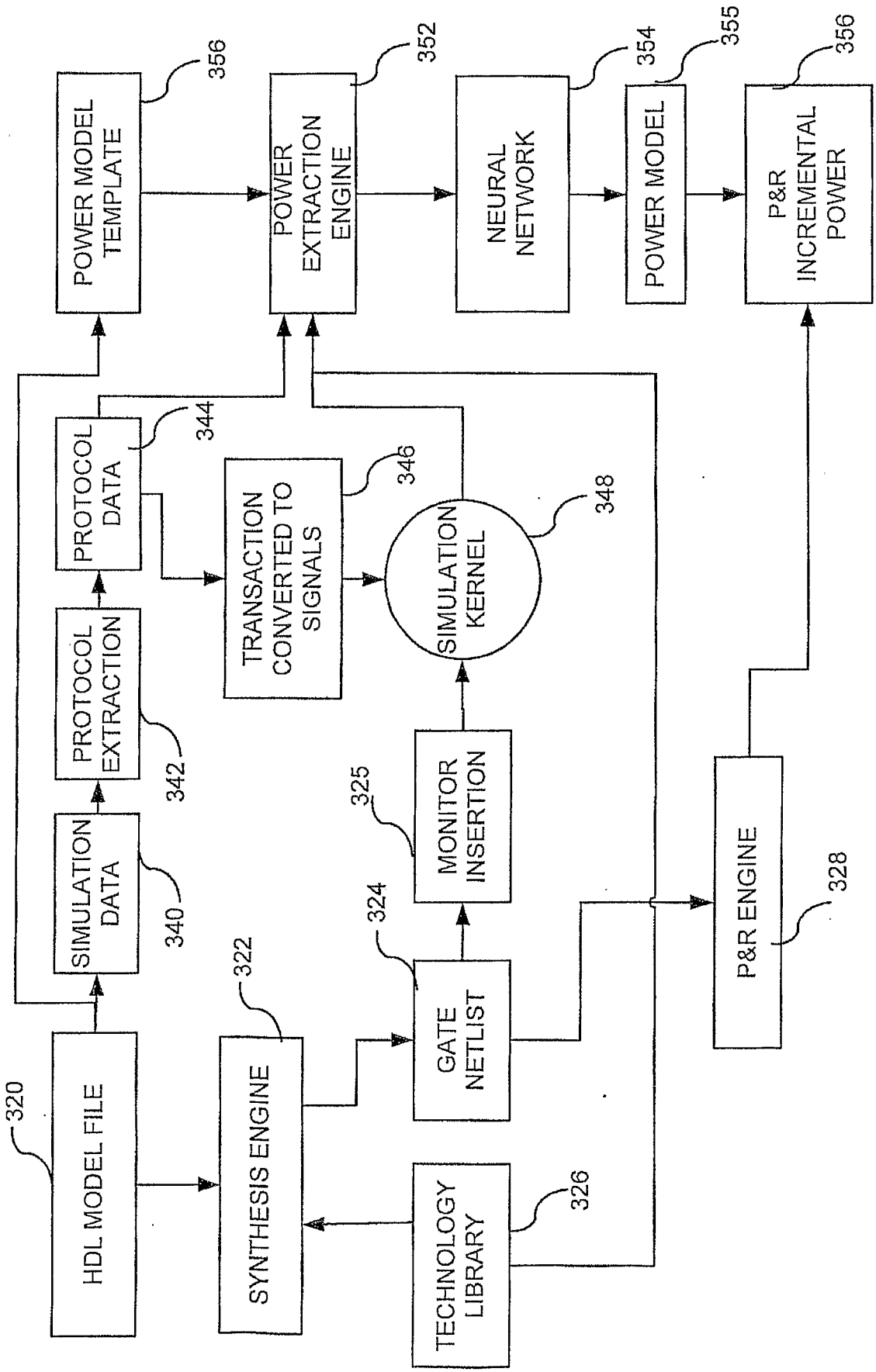

HDL MODEL FILE
320
SYNTHESIS ENGINE
322
TECHNOLOGY LIBRARY
326
GATE NETLIST
324
SIMULATION DATA
340
PROTOCOL EXTRACTION
342
PROTOCOL DATA
344
TRANSACTION CONVERTED TO SIGNALS
346
SIMULATION KERNEL
348
MONITOR INSERTION
325
P&R ENGINE
328
POWER MODEL TEMPLATE
356
POWER EXTRACTION ENGINE
352
NEURAL NETWORK
354
POWER MODEL
355
P&R INCREMENTAL POWER
356

FIGURE 13

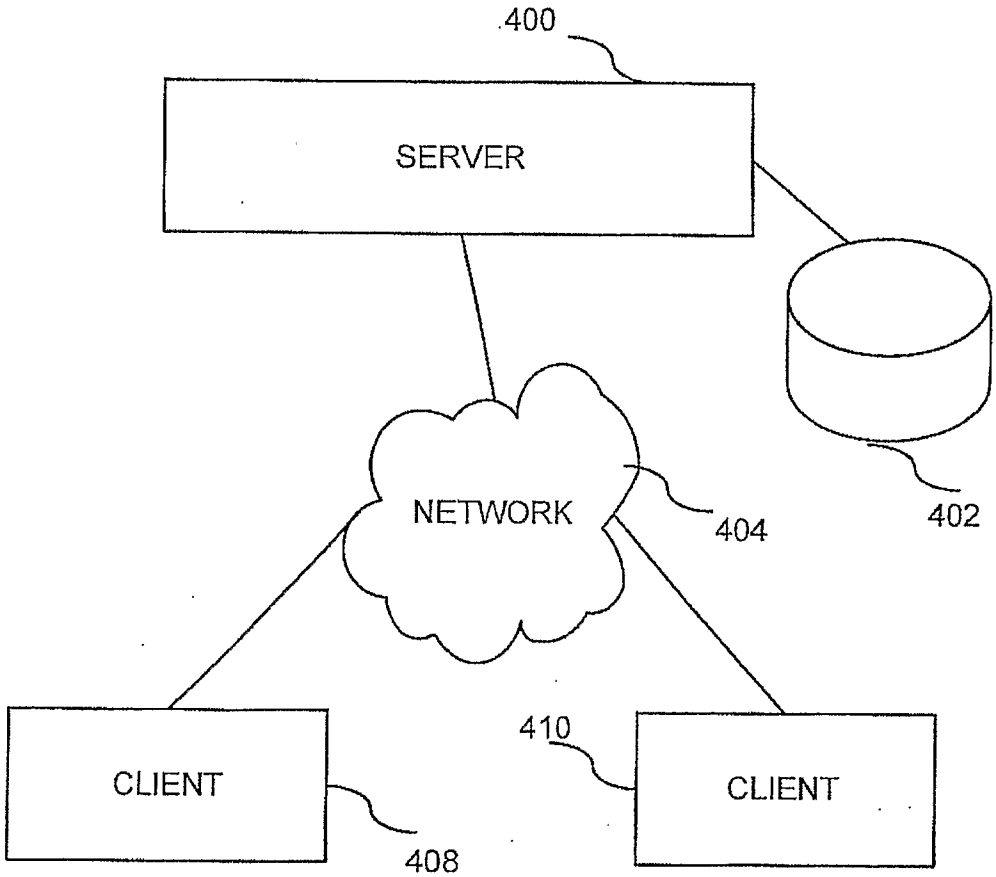
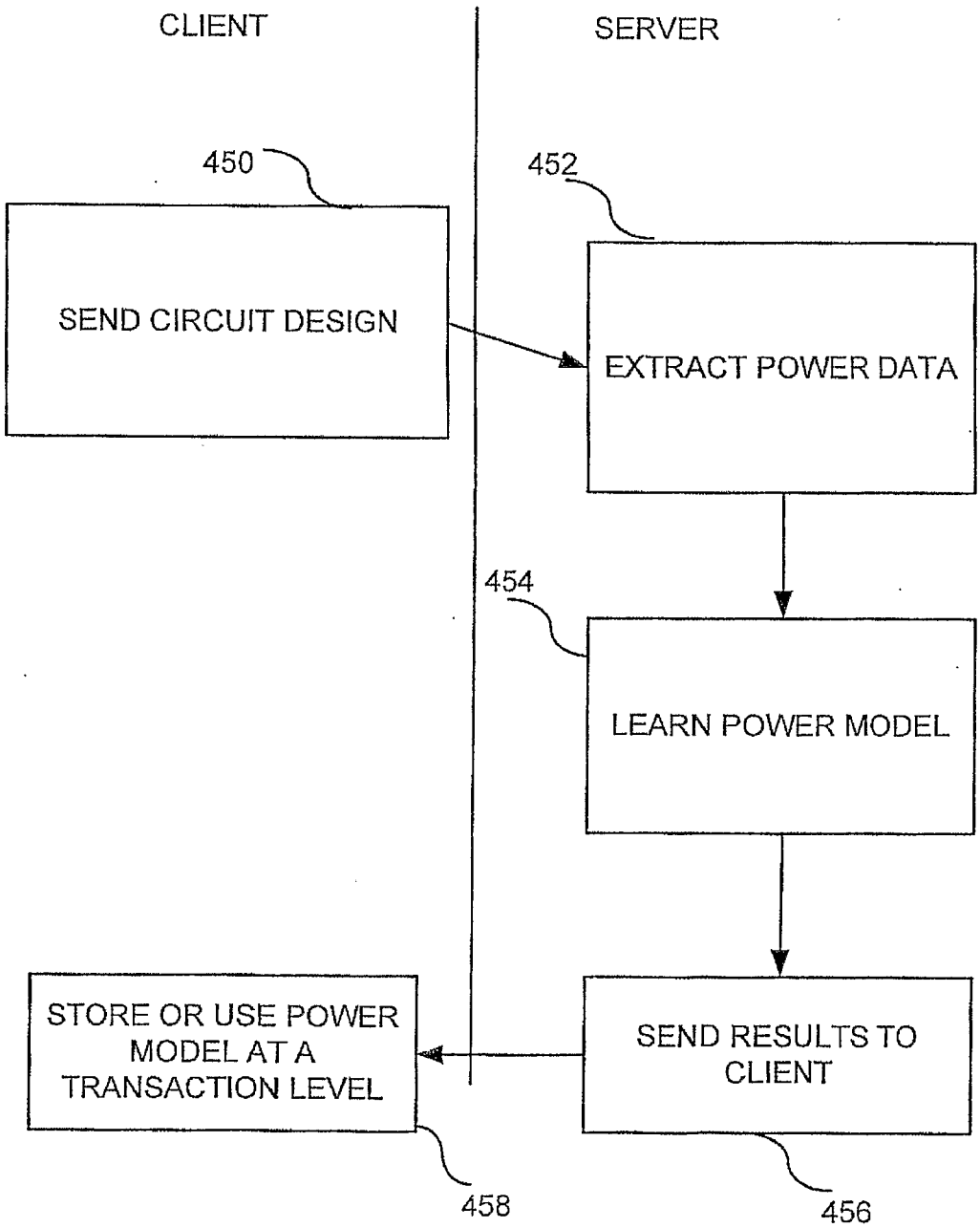


FIGURE 14


CLIENT
SERVER
450
452
SEND CIRCUIT DESIGN
EXTRACT POWER DATA
454
LEARN POWER MODEL
STORE OR USE POWER MODEL AT A TRANSACTION LEVEL
SEND RESULTS TO CLIENT
458
456

INTERNATIONAL SEARCH REPORT

International application No
PCT/IL2006/001350

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F17/50

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

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

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2002/004927 A1 (TAKAHASHI MIWAKA [JP] ET AL) 10 January 2002 (2002-01-10) abstract figures 2-4,12,14,20 paragraph [0084] - paragraph [0096] paragraph [0120] - paragraph [0131]	1-21
Y	US 2005/131666 A1 (TSAI JIEN-SHEN [TW] ET AL) 16 June 2005 (2005-06-16) abstract figures 2,4,5 paragraph [0008] - paragraph [0013] paragraph [0016] - paragraph [0019] paragraph [0030] - paragraph [0040] paragraph [0045] paragraph [0072] - paragraph [0074]	1-21

-/--

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
30 March 2007	17/04/2007

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	ALONSO NOGUEIRO, L

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/IL2006/001350

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	NAM SUNG KIM ET AL: "Microarchitectural Power Modeling Techniques for Deep Sub-Micron Microprocessors" PROCEEDINGS OF THE 2004 INTERNATIONAL SYMPOSIUM ON LOW POWER ELECTRONICS AND DESIGN. ISLPED'04. NEWPORT BEACH, CA, AUG. 9 - 11, 2004, INTERNATIONAL SYMPOSIUM ON LOW POWER ELCTRONICS AND DESIGN, NEW YORK, NY : ACM, US, 9 August 2004 (2004-08-09), pages 212-217, XP010764367 ISBN: 1-58113-929-2 page 211 - page 217 -----	1-21
A	US 6 263 301 B1 (COX STEVEN G [US] ET AL) 17 July 2001 (2001-07-17) abstract figures 3,4,8 column 2, line 20 - column 3, line 60 column 5, line 25 - line 45 column 8, line 65 - line 67 -----	1-21

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/IL2006/001350

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2002004927 A1	10-01-2002	JP 2001338010 A	07-12-2001
US 2005131666 A1	16-06-2005	NONE	
US 6263301 B1	17-07-2001	NONE	