**PCT** WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

**(51) International Patent Classification 4 :**
**G06F 15/31, 15/32, 15/324**
**G06F 15/347, 15/60**

**A1**

**(11) International Publication Number: WO 88/ 01412**

**(43) International Publication Date:** 25 February 1988 (25.02.88)

**(21) International Application Number:** PCT/US87/02079

**(22) International Filing Date:** 20 August 1987 (20.08.87)

**(31) Priority Application Number:** 898,476

**(32) Priority Date:** 20 August 1986 (20.08.86)

**(33) Priority Country:** US

**(71) Applicant:** DIGITAL EQUIPMENT CORPORATION [US/US]; 146 Main Street, Maynard, MA 01754 (US).

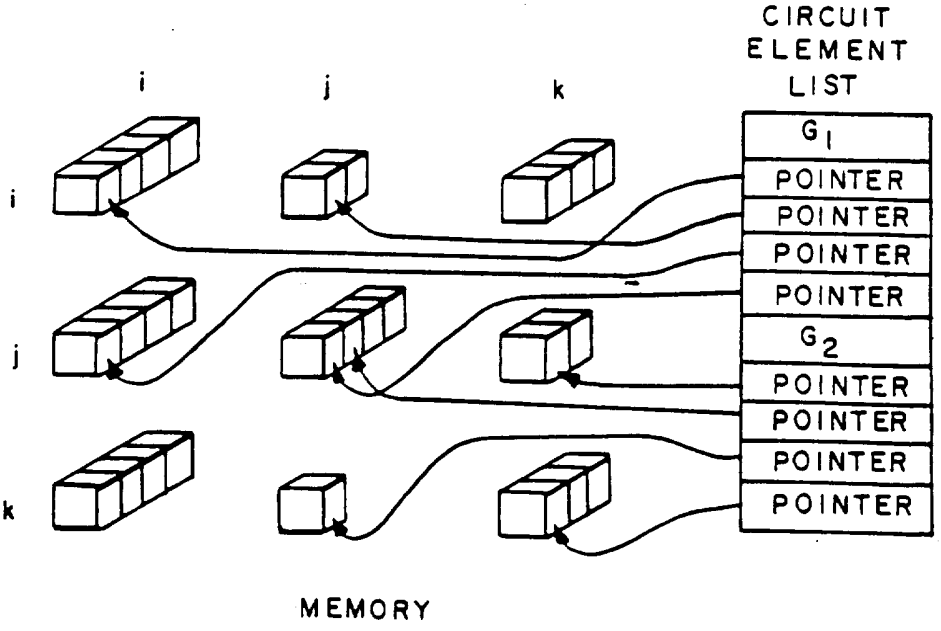
**(72) Inventors:** BISCHOFF, Gabriel ; 2 Royal Crest Drive, #10, Marlboro, MA 01752 (US). GREENBERG, Steven ; 67 Laurel Drive, Bolton, MA 01740 (US).

**(74) Agents:** CESARI, Robert, A. et al.; Cesari and McKenna, Union Wharf East, Boston, MA 02109 (US).

**(81) Designated States:** AT (European patent), BE (European patent), CH (European patent), DE (European patent), FR (European patent), GB (European patent), IT (European patent), JP, LU (European patent), NL (European patent), SE (European patent).

**Published**
*With international search report.*
*Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.*

**(54) Title:** METHOD AND APPARATUS FOR CIRCUIT SIMULATION USING PARALLEL PROCESSORS INCLUDING MEMORY ARRANGEMENT AND MATRIX DECOMPOSITION SYNCHRONIZATION




**(57) Abstract**

A system for simulating LSI and VLSI circuits generates a matrix for solution of simultaneous equations necessary for the circuit (G1, G2) simulation. It synchronizes parallel processors during matrix decomposition by assigning a single processor to given row (i, J, K) within the matrix and sets a flag when the values in a row extending from a diagonal matrix element are ready for use.

*FOR THE PURPOSES OF INFORMATION ONLY*

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

| | | | | | |
|---|---|---|---|---|---|
| **AT** | Austria | **FR** | France | **ML** | Mali |
| **AU** | Australia | **GA** | Gabon | **MR** | Mauritania |
| **BB** | Barbados | **GB** | United Kingdom | **MW** | Malawi |
| **BE** | Belgium | **HU** | Hungary | **NL** | Netherlands |
| **BG** | Bulgaria | **IT** | Italy | **NO** | Norway |
| **BJ** | Benin | **JP** | Japan | **RO** | Romania |
| **BR** | Brazil | **KP** | Democratic People's Republic of Korea | **SD** | Sudan |
| **CF** | Central African Republic | | | **SE** | Sweden |
| **CG** | Congo | **KR** | Republic of Korea | **SN** | Senegal |
| **CH** | Switzerland | **LI** | Liechtenstein | **SU** | Soviet Union |
| **CM** | Cameroon | **LK** | Sri Lanka | **TD** | Chad |
| **DE** | Germany, Federal Republic of | **LU** | Luxembourg | **TG** | Togo |
| **DK** | Denmark | **MC** | Monaco | **US** | United States of America |
| **FI** | Finland | **MG** | Madagascar | | |

# METHOD AND APPARATUS FOR CIRCUIT SIMULATION USING PARALLEL PROCESSORS INCLUDING MEMORY ARRANGEMENT AND MATRIX DECOMPOSITION SYNCHRONIZATION

## FIELD OF THE INVENTION

The field of the present invention relates to computer assisted circuit simulation employing parallel processing.

## BACKGROUND AND SUMMARY OF THE INVENTION

In the field of LSI and VLSI circuits, increased circuit complexity and increased density of the devices on a single chip have resulted in an ever increasing need for computer aided circuit simulation, which is capable of carrying out the necessary circuit simulation in a reasonable period of time. Computer programs such as SPICE and ASTAP have been employed for such circuit simulation. However, these are proven to be less and less adequate as the circuits become more dense and more complex on a single chip.

A variety of algorithms for the efficient and accurate simulation of complex circuits have been proposed for example, Nagle, "SPICE2. A Computer Program to Simulate Semiconductor Circuits," University of California, Berkeley, Memo No. ERL-M520, May 1985; Weeks, "Algorithms for ASTAP-A Network Analysis Program," IEEE Transactions on Circuit Theory, Volume CT-20, pp. 628-634, November 1983; Saleh, Kleckner and Newton, Iterated Timing Analysis and SPICE1," Proceedings of the IEEE International Conference on computer Aided Design, Santa Clara, CA., pp. 139-140, September 1983; White and Sagiovanni-Vincentelli, "Relax 2.1-A Waveform Relaxation Based Circuit Simulation Program," Proceedings, International Custom Integrated Circuits Conference, Rochester, NY, pp. 232-236, June 1984. Salkallah and Director, "An Activity Directed Circuit Simulation Algorithm," Proceedings IEEE Conference on

Circuits and Computers, pp. 1032-1035, October 1980.

More recently the implementation of some of these algorithms has been suggested in a multi-processing environment. Circuit simulation algorithms based on iterative methods, such as Gauss-Jacobi, however, while easy to parallelize, usually have slow convergence rates unless some of the parallelization utility is sacrificed. Iterative methods, moreover, are not appropriate for tightly coupled circuits. The RELAX simulator lumps together tightly coupled nodes and solves the sub-networks using a direct method. Such partitioning algorithms also have to deal with conflicting constraints when convergence properties, processor number and load balancing among processors need to be taken into account.

It has also been suggested in the past, that parallel processing using a plurality of computers, or computers having a plurality of central processing units which operate in parallel with each other, and closely coupling the plurality of processors to a shared memory, is a solution to the increasingly complex problem of circuit simulation. One such solution is proposed in Jacob, Newton, and Pederson, "Direct-Method Circuit Simulation Using Multiprocessors," Proceedings of the International Symposium on Circuits and Systems, May 1986. Another such solution is proposed in White, "Parallelizing Circuit Simulation - A Combined Algorithmic and Specialized Hardware Approach, Proceedings of the ICCD, Rye, New York, 1985.

Each of these references, however, recognizes a significant drawback to the utilization of parallel processing for circuit simulation. Since the processors are sharing the same memory, and the processors are in parallel computing values for different circuit elements throughout the LSI or

VLSI circuit, or some defined sub-portion of such circuit, the processors will quite often be solving for values at the same node within the circuit at the same time. Since the processing algorithms also typically involve iterative processes, which update values for a given node based on computations relating to circuit elements connected to the node, and a memory location within the computer is used to store the values at any given time for a particular node, the problem arises that the access to the shared memory must be strictly controlled. Otherwise, more than one processor may write information into a memory location corresponding to a particular node, at the same time another processor is attempting to read previously stored information related to that node or write new information relating to that node, based on its computations with respect to a different circuit element attached to the node.

The solution in the past has been to require special locking mechanisms within the programs controlling the multiple processors in order to prevent such occurrences. This in effect lengthens the total CPU processing time required, in that much time is spent waiting for the access to the particular memory location to be opened to another of the processing units operating in parallel, which desires to read or write information into the particular memory location. In fact, the proposed solution in the past has been to block write access to the entire matrix when one CPU is writing to the matrix.

The Jacob et al. paper, noted above, investigates the parallel implementation of a direct method circuit simulator using SPLICE, and using up to eight processors of a shared memory machine. An efficiently close to forty-five percent

was reported. Much of the failure to achieve a higher percentage efficiency is the result of synchronization of the multiple processors in their access to common shared memory locations.

Parallelization employing such simulation programs as SPICE uses a list of circuit components and parameters, which may be constants or variables, for example, in the case of MOSFET transistors geometric parameters, currents, charges, capacitances, etc., which are then used to calculate the individual components or terms, the sum of which equals the value for each entry at a position in a matrix. The matrix is used to solve a number of simultaneous equations for the variables, for example, voltages, at each node within the circuit. The matrix typically consists of an X by X matrix with X approximately equal to the number of nodes in the circuit. Since the circuit elements are typically connected to at least two nodes, and may often be connected to more than two nodes, the parallelization techniques employed in the past suffer from the disability that the summation of the terms, in order to define the value at the matrix location for a given node, may be influenced by more than one circuit element, and the parallel processors may be computing the effect of two or more circuit elements on the same matrix node location at the same time. This results in the need for the use of some technique, for example, interlocks, to prevent simultaneous writing or reading of data by more than one processor into or from the memory location corresponding to the matrix location.

Once the matrix is loaded, it is known in the art, using SPICE, to solve the matrix using sparse matrix LU decomposition. Parallelization of the matrix solution phase is also important, and presents unique problems. For larger

circuits, the CPU time needed for the matrix solution phase will dominate over that needed for the matrix load phase. Efficient parallelization schemes are known for full matrices, as is reported in Thomas, "Using the Butterfly to Solve Simultaneous Linear Equations", BBN Laboratories Memorandum, March 1985. However, sparse matrices are more difficult to decompose efficiently in parallel. The LU decomposition algorithm has a sequential dependency and the amount of concurrent work which can be done at each step, using SPICE, in a sparse matrix is small. Algorithms detecting the maximum parallelism at each step have been proposed for vectorized circuit simulation. Yamamoto and Takahashi, "Vectorized LU Decomposition Algorithms for Large Scale Circuit Simulation", IEEE Transactions on computer Aided Design, Vol. Cad-4, No. 3, pp. 232-239, July 1985. Algorithms based upon a pivot dependency graph and task queues have been proposed. Jacob et al., supra. The overhead associated with task queues makes the efficiency of these algorithms questionable.

Recognizing the need for an improved circuit simulation apparatus and method, it is a general object of the present invention to provide a circuit simulation apparatus and method for simulating LSI and VLSI circuits which eliminates the costly synchronization requirement of the prior art and, in addition, more efficiently implements the LU decomposition in parallel using multiple processors.

A feature present invention relates to providing memory locations corresponding both to the matrix location and to a specific one of a plurality of terms which determine the ultimate value for the entry at the matrix, and summing the terms from the plurality of memory locations corresponding to a matrix entry to define a single value for the matrix entry.

A further feature of the present invention relates to synchronizing the parallel processors during matrix decomposition by assigning a single processor to a given row within the matrix and setting a flag when the values in a row extending from a diagonal matrix element are ready for use.

The features of the present invention discussed above are not intended to be all-inclusive but rather are representative of the features of the present invention in order to enable those skilled in the art to better appreciate the advance made over the art and to understand the invention. Other features will become more apparent upon reference to the detailed description of the preferred embodiment which is contained below and with reference to the figures of the drawing.

## BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a schematic representation of the prior art SPICE program and circuit simulation apparatus implementing that program;

Fig. 2 is a schematic representation of a method and apparatus according to the present invention for generating the matrix for solution of simultaneous equations necessary for circuit simulation.

## DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

In the past, as shown in Fig. 1, representing the prior art, the parallelization of circuit simulation, for example, using SPICE, uses a list of circuit elements contained within the circuit, and a matrix, with the matrix locations corresponding generally to the circuit nodes. Depending on

the number of circuit elements connected to a node, and/or the type of circuit elements connected to a node, a plurality of different terms can contribute to the ultimate value for the matrix location. The sum of these values is used in solving the simultaneous equations, in order to solve for the necessary variables to conduct the circuit simulation. With the SPICE program the use of parallel processors has been employed to analyze in parallel, different circuit elements and their contributions to the matrix values. However, the parallel processing of different circuit elements, each of which may contribute to different ones of the terms for the same matrix location entry, is not allowable in the past methods of parallel processing. This is so since, in the past, data representing the most recent value for the summation of all of the terms for a given matrix location has been stored in a single memory location, and summed with the values calculated for the next circuit elements influencing the value of the sum of the terms for the given matrix location. Thus, in effect, each of the parallel processors must wait its turn to add its contributing term to sum the value for each matrix location entry.

As shown in Fig. 1, the list of circuit elements contains the parameters, which determine the term of terms contained in the matrix location for a given matrix entry, and a pointer, which points to the memory location for the given matrix entry. In actuality, using, e.g., the VAX11/780, the data is represented in a double precision value requiring eight bytes of data for each memory location.

The present invention employs a list of circuit elements, as shown in Fig. 2, essentially identical to the list employed by circuit simulators using SPICE. The list of circuit

elements contains a plurality of pointers, each of which point to a memory location within an array of memory locations associated with a given matrix entry. There is, therefore, a memory location assigned to each of the plurality of circuit elements which affects the sum of the terms for a given matrix entry. The input from one of the plurality of parallel processors operating on the calculation of the term for a given circuit element, which contributes to the total value for the given matrix entry will be stored in a designated memory location, and the input from another processor operating in parallel on the calculation of the contribution to the total value from a different circuit element will have a different designated memory location. The different memory locations correspond to the particular matrix entry by being included within an array of memory locations corresponding to the given matrix entry, but each corresponds to a contribution from a different circuit element.

In effect, a three dimensional matrix of memory locations is established, with each matrix entry having a plurality of memory locations associated with it, each corresponding to a different circuit element, the influence of which determines a portion of the total value for each entry, which is employed in solving the simultaneous equations represented by the matrix.

Utilization of the resent invention greatly speeds up the loading of the matrix entries for circuit simulation in order to solve the simultaneous equations. At present, the loading of the matrix entries, occupies approximately 80% of the total computing time necessary for circuit simulation. With the utilization of the present invention, including the improvement in the ability to parallelize the matrix loading

portion of the operation, the efficiency of the circuit simulator's utilization of the plurality of parallel processors has been elevated to approximately 90%, as opposed to the 45% reported in the Jacob et al. paper noted above.

The present invention can be implemented on VAX8800 and 8300 computers manufactured by Digital Equipment Corporation which contain dual processors having a shared memory. In addition to defining the memory noted above, the parallelization program employing SPICE is divided into serial phases. Each phase is divided into tasks which are executed serially or concurrently. A phase whose task in executed serially is called a single stream phase whereas a phase whose tasks are executed concurrently is called a multiple stream phase. Single stream phases are executed by a master processor and the multiple stream phases are executed by slave processors. The slave processors are idle when the master is active and the master processor is idle when any of the slave processors is active. Processes are dynamically assigned to processors by the VAX VMS operating system. The VAX VMS operating system also provides event flags (semaphores) used for synchronization between processors.

A general FORTRAN library has been designed for this environment. A program using this library has the following structure:

```
COMMON/LOCAL/LOCAL_VARIABLES
COMMON/SHARED/SHARED_VARIABLES
IF MASTER_PROCESS THEN
     CALL MASTER_CODE
ELSE
     CALL SLAVE_CODE
ENDIF
END
```

Master and Slave processes run the exact same executable file. In the above code MASTER_PROCESS is a logical function from the library. This library has presently seven entries and performs the following functions:

Initialization. This task is performed by the logical function MASTER_PROCESS. It consists of discriminating between a MASTER and SLAVE process, setting up the section of memory shared between the MASTER and SLAVE process, initializing the event flags used for synchronization in creating the slave process. This logical is set to TRUE if the process for executing the code is the master process and set to FALSE if it is the slave process.

Synchronization. This task is performed by four sub-routines. FORK, JOIN, JOIN_EXIT and JOIN_FORK. The sub-routine FORK is called by the master process in order to signal the slave process to proceed and then to wait for them to signal back. The sub-routine JOIN is called by the slave process in order to signal the master process to proceed and then to wait for it to call back. The sub-routine JOIN_EXIT is called by the slave process in order to signal the master process to proceed and then to exit. The sub-routine JOIN_FORK is called by the slave process in order to synchronize two multiple stream phases with no intervening single stream phase.

Interlock Memory Access. This task is performed by two sub-routines locked and unlocked. The variable used in the argument of sub-routines holds the state of the lock for the associated section of shared memory whose access needs to be interlocked. The sub-routine LOCK and UNLOCK use the atomic test and set instruction, BBSSI and BBCCI, of the VAX

architecture.

The interlocks are necessary in certain sub-routines not related to the present invention, for example, synchronization routines and load balancing routines.

The tasks performed in parallel by the circuit simulation apparatus and method according to the present invention are matrix load, matrix LU decomposition and time step computation.

Those skilled in the art will appreciate that by employing the aspects of the present invention just described a number of advantages are achieved over the existing art. In SPICE, as it is known in the art, the matrix load phase for each circuit element includes evaluation and loading of the element's contribution to the matrix entry and the right-hand side of the simultaneous equations, as well as convergence checking of the branched currents. Elements evaluation and convergence checking have no write contention in shared memory. However, the loading of the matrix and the right-hand side requires writing to memory locations which may be shared by several elements. Synchronization of write access to the matrix through a single lock on the whole matrix has been proposed by Jacob, et al., supra. In such a case only one processor can write into memory for the entire matrix at a given time. This leads to contention for shared resources and decreases efficiency.

In the present invention locking the entire matrix, or even some portion thereof, is avoided by the utilization of a memory structure which accommodates the storage of the individual contributions to a given matrix location entry from each of the circuit elements affecting that matrix location entry. The same list of circuit elements is used, as had been

done in the past with SPICE. However, according to the present invention, the number of circuit elements attached to a given node is used to create an array of memory locations associated with the node and the list of circuit elements is augmented by the incorporation of a pointer which directs the storage in a specific one of the array of memory locations of the term indicating the contribution of the particular circuit element to the total entry for the matrix location.

The memory structure so generated may be visualized as a three dimensional memory structure, with two dimensions defining the matrix locations and the third dimension defining the number of circuit elements affecting each matrix location. There is no unused memory in this structure since it has a variable depth in the third dimension.

After the list of circuit elements is processed, and the memory locations within each array of memory locations thereby loaded with the terms representing contributions of the individual circuit elements to the total entry for the matrix location, the contents of the arrays are summed, again using parallel processing, without any contention for access to shared memory, in order to generate a single matrix for the computation of the solutions to the simultaneous equations for the circuit simulation.

Even distribution of tasks among the processors operating in parallel is achieved so that no slave process stays idle while others are computing. A dynamic task allocation is used for the multiple stream phase of matrix load. Since the time needed to load the factor contribution of each given circuit element cannot be estimated exactly, dynamic task allocation is used in lieu of static task allocation. Dynamic task allocation is achieved through an array of tasks whose number

exceeds the number of processors. A task consists of a list of circuit elements to be loaded. Tasks are defined so that each requires approximately the same amount of work. The amount of work needed to load a circuit element is roughly estimated by neglecting bypass and evaluating the CPU time needed to load the given element. Dynamic task allocation minimizes any imbalance which may occur during simulation through device model computation bypass. Task allocation during the summing phase of the matrix load operation is done statically, since the work needed to perform this phase can be divided into tasks requiring roughly the same amount of CPU time. The only interlocked access to shared memory during the matrix load phase is the one required on the array index, which defines the next task when dynamic task allocation is employed. This index is successively read and incremented by all slave processors.

The present invention employs LU matrix decomposition mathematically similar to that performed by SPICE. When the LU decomposition requires pivoting, only a single master processor is used. This is done twice during transient analysis, for full pivoting, once for the first LU decomposition performed in the DC operating point computation and second for the first LU decomposition of the transient wave form computation. For all other decompositions, pivoting is used only relatively infrequently when a diagonal element used as a pivot is less than a predetermined threshold.

During the time the parallel processing is being used for the LU decomposition, the present invention relates to synchronizing the parallel processors so that they will perform decomposition operations only on valid data. This is accomplished by assigning each slave processor a set of rows

of the circuit matrix. When the SPICE decomposition algorithm progresses to a particular diagonal element, each processor updates the rows in its assigned set. The usual linked list of matrix entries below the diagonal is, according to the present invention, broken down into separate lists based upon the rows assigned to each processor. A flag is associated with each diagonal element to ensure that it is never used before its final value is available. Use of this flag results in an efficient synchronization scheme. The flags are shared data accessed by multiple processors in the read mode, but only one in the write mode, so that no locks are needed.

This technique remains efficient for up to about four processors operating in parallel. Beyond this, optimum reordering and assignment of the rows may be necessary to improve the efficiency of the algorithm. The present invention makes use only of the existing SPICE version 2G5 reordering algorithm.

An heuristic algorithm assigns rows to the slave processors. The first step of this algorithm is to divide the circuit matrix into blocks of consecutive columns such that the slave processors can work within blocks without synchronization. The blocks are found by scanning the set of matrix columns from left to right and assigning them to blocks so that within a block no dependency among diagonal elements exists when performing LU decomposition. Then in each block rows containing nonzero subdiagonal elements are assigned to slave processors by determining the number of updates necessary to complete a row and dividing the amount of work assigned to the slave processors during the LU decomposition so that it is balanced among them. For row assignment, the blocks are processed from right to left.

Parallelization of the time step computation does not present a major problem as the computation of the local truncation error for each energy storage element is independent. Each slave processor is assigned a set of energy storage elements and computes the minimum time step required by its set of energy storage elements. The master processor then computes the minimum time step among the time steps returned to it by the slave processors. The energy storage elements are assigned to the slave processors so that the work among them is balanced.

The preferred embodiment of the present invention has been described in order to comply with the requirements of the Patent Statutes, and is not intended to limit the present invention to any particular embodiment. Rather, those skilled in the art will appreciate that many modifications and changes may be made to the present invention without departing from the scope of the invention, and the inventors intend that the appended claims cover such changes and modifications as come within the scope and content of the appended claims.

What is claimed as new and desired to be secured by Letters Patent of the United States is:

## CLAIMS

1. In a computing apparatus employing a plurality of central processing units CPUs, operating in parallel, and shared memory shared by the CPUs, and wherein the plurality of CPUs are operated to create a matrix utilized to solve complex multivariable circuit equations for the purpose of circuit simulation evaluation, with the matrix entries assigned according to a matrix of circuit nodes, a method of avoiding the need for synchronization of the access to the shared memory by the plurality of CPUs, comprising the steps of:

determining the number of circuit elements connected to each node corresponding to a matrix entry;

assigning a plurality of distinct memory locations to each matrix entry according to the number of circuit elements connected to the node corresponding to the matrix entry, with each memory location assigned to a specific circuit element;

directing the CPUs to write into the respective memory location associated with a respective circuit element connected to the respective node when operating on the contribution of the respective circuit element to the matrix entry associated with the respective node.

2. The method of claim 1 further comprising the step of:

Summing the contents of the plurality of memory locations associated with a respective matrix entry to obtain value for the matrix entry to be used in computing the solution of the matrix for the solution of a plurality of simultaneous equations.

3. A computing apparatus having a plurality of central

processing units (CPUs) and a memory shared by the CPUs, and wherein the CPUs are operated in parallel to solve complex multivariable circuit equations in a matrix format, with the matrix entry locations assigned according to a matrix of circuit nodes, comprising:

means for determining the number of circuit elements connected to each node corresponding to a matrix entry;

means for assigning a plurality of distinct memory locations to each matrix entry according to the number of circuit elements connected to the corresponding node, with each respective memory location assigned to a respective circuit element;

means for directing each of the CPUs to write into the respective memory location associated with the respective circuit element when operating on the contribution of the respective circuit elements to the respective matrix entry corresponding to the respective node.

4. The apparatus of claim 3, further comprising:

means for summing the contents of the respective memory locations associated with a respective one of the matrix entries.

5. In a computing apparatus employing a plurality of central processing units (CPUs) operating in parallel to decompose a matrix utilized to solve complex multivariable circuit equations for the purpose of circuit simulation evaluation, a method of synchronizing the matrix decomposition to prevent the CPUs from operating on invalid data during the matrix composition, comprising the steps of:

assigning to each of the plurality of CPU's the tasks

associated in updating all of the matrix entries within selected rows of the matrix as required for matrix update during matrix decomposition;

generating an indicator associated with each diagonal entry in the matrix when the one of the plurality of processors has completed updating the matrix entries in the row in which the diagonal entry resides;

preventing any of the others of the plurality of CPU's from updating any entries in a row assigned to such other CPU, which update depends on a given matrix diagonal entry, unless the presence of the indicator for the given matrix diagonal element is detected.

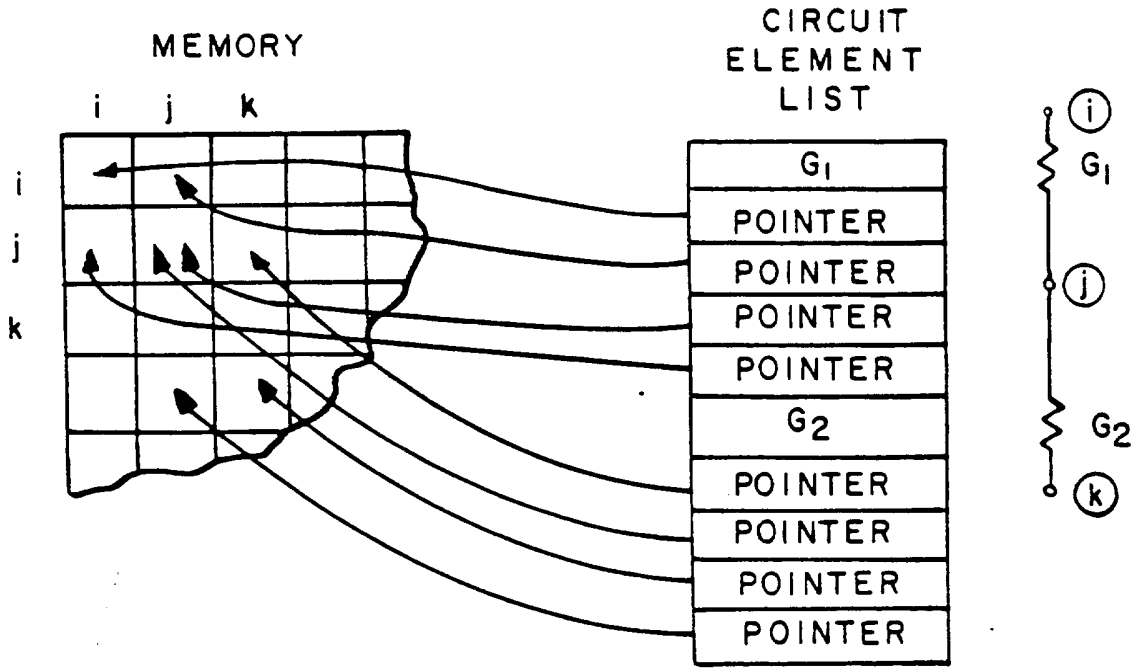



FIG. 1 PRIOR ART

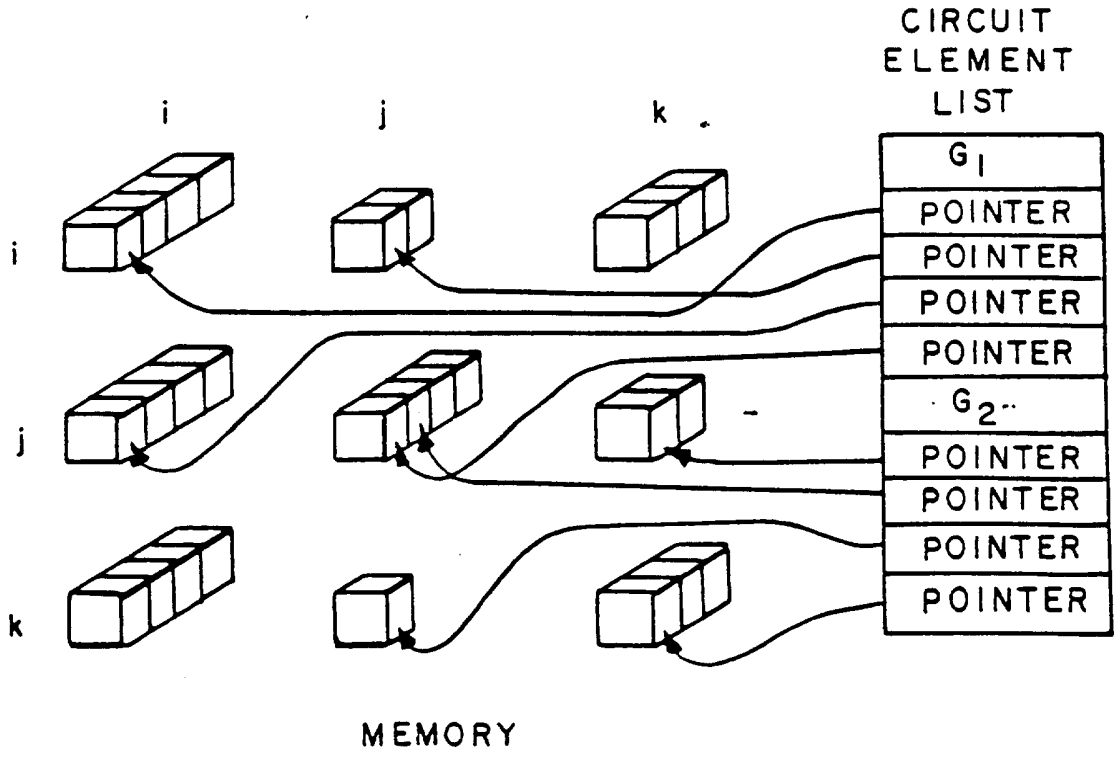



FIG. 2

# INTERNATIONAL SEARCH REPORT

International Application No PCT/US87/02079

## I. CLASSIFICATION OF SUBJECT MATTER (if several classification symbols apply, indicate all) [3]

According to International Patent Classification (IPC) or to both National Classification and IPC

IPC (4): G06F 15/31, 15/32, 15/324, 15/347, 15/60

U.S. CL. 364/200

## II. FIELDS SEARCHED

Minimum Documentation Searched [4]

| Classification System | Classification Symbols |
|---|---|
| U.S. | 364/200, 300, 488, 489, 754, 900 |

Documentation Searched other than Minimum Documentation to the Extent that such Documents are Included in the Fields Searched [5]

## III. DOCUMENTS CONSIDERED TO BE RELEVANT [14]

| Category * | Citation of Document, [16] with indication, where appropriate, of the relevant passages [17] | Relevant to Claim No. [18] |
|---|---|---|
| Y | N, IEEE International Symposium on Circuits and Systems, Volume 1, issued May 1986 (Piscataway, New Jersey), "Direct-Method Circuit Simulation Using Multiprocessors", see pages 170-172. | 1-4 |
| Y | N, IEEE Transaction on Computer Aided Design, Volume Cad-4 No. 3, issued July 1985, "Vectorized LU Decomposition Algorithms for Large Scale Circuit Simulation", see pages 232-239. | 5 |
| A | US, A, T915,008 (GUSTAVSON) 9 October, 1973, See the entire document. | 5 |
| A | US, A, 3,629,843 (SCHEINMAN) 21 December, 1971, See the entire document. | 1-4 |

* Special categories of cited documents: [15]

"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

"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

## IV. CERTIFICATION

| Date of the Actual Completion of the International Search [2] | Date of Mailing of this International Search Report [2] |
|---|---|
| 23 November 1987 | 05 JAN 1988 |
| **International Searching Authority [1]** | **Signature of Authorized Officer [20]** |
| ISA/US | Ayni Mohamed<br>AYNI MOHAMED |

**FURTHER INFORMATION CONTINUED FROM THE SECOND SHEET**

**V. ☐ OBSERVATIONS WHERE CERTAIN CLAIMS WERE FOUND UNSEARCHABLE [10]**

This international search report has not been established in respect of certain claims under Article 17(2) (a) for the following reasons:

1. ☐ Claim numbers ....., because they relate to subject matter [12] not required to be searched by this Authority, namely:

2. ☐ Claim numbers ....., because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out [13], specifically:

**VI. ☒ OBSERVATIONS WHERE UNITY OF INVENTION IS LACKING [11]**

This International Searching Authority found multiple inventions in this international application as follows:

I. Claims 1-4, drawn to a system for avoiding synchronization of access by plural processors to a shared memory; class 364 subclass 200.

II. Claim 5 is drawn to a method for synchronization of a matrix decomposition; class 364 subclass 200.

1. ☒ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims of the international application.

2. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims of the international application for which fees were paid, specifically claims:

3. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claim numbers:

4. ☐ As all searchable claims could be searched without effort justifying an additional fee, the International Searching Authority did not invite payment of any additional fee.

Remark on Protest

☐ The additional search fees were accompanied by applicant's protest.

☐ No protest accompanied the payment of additional search fees.