PARALLEL IMPLEMENTATION OF VLSI HED CIRCUIT SIMULATION

INFORMATICA 2/91

Srilata Raman
University of Iowa, Iowa City, U.S.A.
Lalit Mohar Patnaik
Indian Institute of Science, Bangalore, India
Jurij Šilc
Marjan Špegel
Jožef Stefan Institute, Ljubljana, Slovenia

Keywords: circuit simulation, direct method, waveform relaxation, parallel algorithm, parallel computer architecture

The importance of circuit simulation in the design of VLSI circuits has channelised research work in the direction of finding methods to speedup the highly compute-intensive problem of circuit simulation. Attempts have been made to find better algorithms and to use parallel architectures to accelerate the simulation task. This paper deals with the two well-known circuit simulation algorithms – direct methods and relaxation method. The issues involved in parallelizing these algorithms and various computer architectures that have been reported in the literature are presented in this paper.

IZVEDBA VZPOREDNE SIMULACIJE VLSI VEZIJ – Potreba po simulaciji VLSI vezij pri njihovem snovanju je usmerila raziskovalno delo v iskanje metod za pohitritev računsko intenzivenega postopka simulacije vezij. Predmet raziskav je iskanje učinkovitejših algoritmov ter uporaba vzporednih arhitektur za izvajanje simulacije. V članku sta obravnavana dva dobro znana algoritma za simulacijo vezij: direktna in relaksacijska metoda. Prikazane so vzporedne različice omenjenih algoritmov ter podan pregled računalniških arhitektur, ki so namenjene izvedbi vzporedne simulacije VLSI vezij.

1 Introduction

The complexity of VLSI circuits is growing with the improvement in manufacturing methods and advent of new technologies. This has manifested itself in the increasing number of devices on a single chip. Design verification of VLSI chips has become indispensable to ensure that the circuit meets its requirements. The simulation of integrated circuit (IC) chips at the electrical level constitutes the most important design verification step. However, the dramatic increase in the complexity of ICs has burdened the capabilities of traditional circuit simulators like SPICE2 [Nag75]. Gate-level logic simulators [ST75] and switch-level simulators [HHL82] can verify circuit functions and provide first order timing information more than three orders of magnitude faster than detailed circuit simulator. However, it is necessary to perform accurate electrical simulation to verify circuit performance for critical path, memory design, and analog circuit blocks, and to detect dc circuit problems such as noise margin errors or incorrect logic threshold. Once of the most common analyses performed by circuit simulators and most expensive in terms of computer time is nonlinear, time-domain transient analysis of electrical circuits. This analysis provides precise electrical waveform information if device models and parasitics of the circuit are characterized accurately. Traditional circuit simulators like SPICE required excessive CPU time to generate voltage and current waveforms for circuits containing more than a few hundred transistors. As an example, a 700 MOSFET circuit analyzed for 4 μs of the simulated time with an average 2 ns time step, takes approximately 4 CPU hours on a VAX 11/780 VMS computer with floating-point accelerator hardware using SPICE2.

This situation has spurred vigorous research aimed at reducing the cost of circuit simulation. A number of approaches have been used to overcome the drawbacks of conventional circuit simulators. The time required to evaluate complex device models has been reduced using the table look-up models [CGK75]. Special-purpose microcode has been used for reducing the time required to solve linear systems and node tearing techniques have been used to exploit circuit latency by bypassing

the solution of the subcircuits whose states are not changing. In addition high performance computers with vector processing capabilities like CRAY [CA79] have been used to exploit parallelism and pipelining available in the circuit simulation program. But, circuit simulation programs are not well suited to such computers. The reason is that as the circuit matrix is sparse and has an irregular structure, the data gather-scatter time dominates the overall program execution time [CA79]. This means that fetching the data stored in memory and writing it back after it has been processed poses a bottleneck. All the above approaches have been found to result in an order of magnitude speedup over SPICE.

A recent development is the use of relaxation methods [NSV84] for solving the set of ordinary differential equations describing the circuit under analysis rather than using the direct sparse matrix methods on which standard circuit simulators are based. Simulators using this method have been shown to provide guaranteed accuracy [NSV84] with upto two orders of magnitude speed improvement for large circuits [LRSV82]. This new method has been found to offer much greater speedups on special-purpose hardware designed to exploit the particular features of the relaxation algorithms [DN84]. In the following sections the direct and relaxation-based algorithms and issues in their parallel implementation are presented.

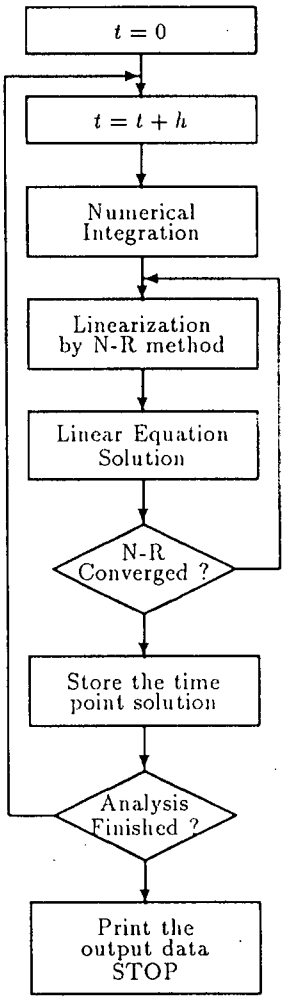

Figure 1: Flowchart of Direct Method of Circuit Simulation.

2 Direct-method for Circuit Simulation

The most common approach to solving the circuit equations in time-domain analysis employs three basic numerical methods [NSV84]: an implicit integration method, the Newton–Raphson (N-R) method and sparse Gaussian elimination method. These tree methods constitute the standard method of circuit simulation on which conventional circuit simulators like SPICE [Nag75] and ASTAP [WJM+73] are based. The analysis portion of a circuit simulation program determines the numerical solution of a mathematical representation of the circuit. The mathematical system of equations for physical circuit is obtained by representing each element in the circuit by its mathematical model. The system of equations describing the complete circuit is given by the model equations and the Kirchoff's current and voltage laws applied to the interconnection of the circuit elements. As a result algebraic-differential equations of the form,

$$F(\dot{\mathbf{x}}, \mathbf{x}, t) = 0 \qquad (1)$$

are obtained. Here, $\mathbf{x} \in \mathbb{R}^N$ is the vector unknown circuit variables, $\dot{\mathbf{x}} \in \mathbb{R}^N$ is the time derivative of $\mathbf{x}$ and F is a nonlinear operator.

Transient analysis determines the time-domain response of the circuit over a specified time-interval $(0, T)$. The flowchart of the standard circuit simulation method is shown in Fig.1.

The entire simulation time interval is divided into a number of discrete time points $(0, t_1, t_2 \ldots, t_n, t_{n+1} \ldots, T)$. $\mathbf{x}^n$, the information from the previous time point is used to predict the solution $\mathbf{x}^{n+1}$

at t_{n+1}. A stiffly stable integration formula like Backward-Euler (BE) with variable time is used to descretize the nodal equation to yield a set of nonlinear, algebraic equations of the form

$$g(\mathbf{x}) = 0 \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^N$ is the vector of unknown variables at time t_{n+1}. The above equations are solved using N-R algorithm to yield a set of sparse linear equations of the form,

$$\mathbf{Ax} = \mathbf{b} \tag{3}$$

where $\mathbf{A} \in \mathbb{R}^{N \times N}$ is a matrix related to the Jacobian of g and $\mathbf{b} \in \mathbb{R}^N$. These equations are solved using direct methods like Gaussian Elimination (GE) or sparse LU decomposition.

The major computation in circuit simulation lies in formulating and solving the system of linear algebraic equations simultaneously. It has been shown in [NP78] that the storage and computer time required by circuit simulation increase rapidly with the size of the circuit, measured in terms of the number of circuit components. Thus for transient analysis, the standard circuit simulators are cost-effective only when the circuit size is limited to a few hundred devices. VLSI circuits with over 10.000 devices impose severe strain on standard circuit simulators. This has necessitated development of alternative circuit simulators. The standard circuit simulators have yet another drawback. For most circuits, the fraction of nodes that change their voltage values at a given point in time decreases as the circuit size increases. So only the circuit equations representing the active nodes need to be solved at any time, bypassing the solution of equations of the nodes which are not active at the time instant. Circuit simulators must exploit this time sparsity or latency because the computational complexity of the GE method applied to an $N \times N$ dense matrix is proportional to $O(N^3)$ whereas the computational complexity of the GE method for sparse matrices is proportional to $O(N^a)$, $1.2 \leq a \leq 1.5$. The performance of standard circuit simulators is compromised for large circuits because they solve the set of equations describing the entire circuit simultaneously irrespective of whether a given node is active or not. Relaxation-based methods help in overcoming these drawbacks.

3 Relaxation Algorithm

The basic concepts of relaxation-based algorithm have been described in great detail in [NSV84]. Relaxation methods can be used with a variety of IC technologies though they are particularly suited to the analysis of large MOS digital ICs. Relaxation-based circuit simulators make an important assumption that a two terminal capacitor is connected from each node of the circuit to the reference node. This assumption is satisfied by the circuits where parasitic capacitances are present between circuit interconnect and ground or the terminals of active circuit elements. Under this assumption, the nodal equations of a circuit are given by,

$$\mathbf{C}(\mathbf{v}(t), \mathbf{u}(t))\dot{\mathbf{v}}(t) = -\mathbf{q}(\mathbf{v}(t), \mathbf{u}(t)), \tag{4}$$

$$\mathbf{v}(0) = \mathbf{V} \tag{5}$$

for $0 \leq t \leq T$ where, $\mathbf{v}(t) \in \mathbb{R}^n$ is vector of node voltages at time t, $\mathbf{V}$ is the given initial values of $\mathbf{v}$, $\dot{\mathbf{v}}(t) \in \mathbb{R}^n$ is vector of time derivatives of $\mathbf{v}(t)$, $\mathbf{u}(t) \in \mathbb{R}^n$ is input vector at time t, $\mathbf{C} : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times n}$ is nodal capacitance matrix, $\mathbf{q} : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^n$, and $\mathbf{q}(\mathbf{v}(t), \mathbf{u}(t)) = [q_1(\mathbf{v}(t), \mathbf{u}(t)), \ldots, q_n(\mathbf{v}(t), \mathbf{u}(t))]^T$, where q_i is sum of currents charging the capacitors connected to node i.

The two common relaxation methods used are the Gauss-Seidel (GS) and the Gauss-Jacobi (GJ) method. Relaxation methods can be used for the solution of equations (4) in different ways. Fig.2 illustrates the levels at which relaxation methods can be applied. Linear relaxation method is applied at the linear equation level and consists of replacing the GE method for solving equation (3) by GJ or GS. Nonlinear relaxation methods are applied at the nonlinear equation level and augment the N-R method applied to equation (2). They replace the linear equation solution based on sparse-matrix techniques. Relaxation methods when applied directly to the system of nonlinear algebraic equations describing the circuit are termed Waveform Relaxation (WR) [NSV84]. As a result of this, the system is decomposed into decoupled subsystems of algebraic-differential equations corresponding to decoupled dynamical subcircuits. Each decoupled subcircuit is then analyzed for the entire simulation time interval using the standard simulation techniques. Such a decomposition permits latency to be exploited. Decomposition into subcircuits permits a trade-off between the amount of time spent

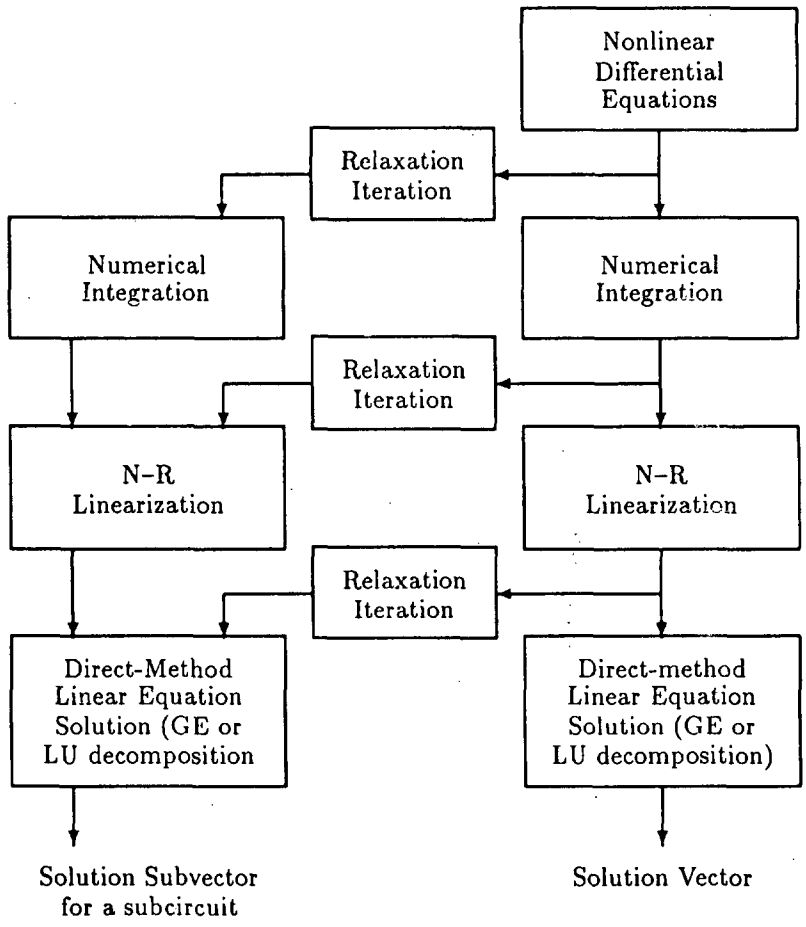

Figure 2: Relaxation Applied at Different Levels of Analisis.

on any single processor at an iteration and the time spent communicating the results of the analysis. This is a key requirement for deriving maximum efficiency from a parallel processing environment.

The WR algorithm using GS iteration is given in Fig.3. Superscript k is the iteration count, subscript i is the component index of a vector, ε is a small positive number, N is the maximum node number and n is the number of circuit variables.

Modifications in the WR algorithm help in improving the speed of convergence. Instead of solving each differential equation for one unknown (point relaxation), the system of differential equations can be partitioned into subsystems having more than one equation (block relaxation). Each decomposed circuit is then solved using standard simulation techniques. Each subcircuit can be analyzed independently [NSV84] from $t = 0$ to $t = T$, using its own time step sequence, controlled by the integration method. As opposed to this, in a standard circuit simulator entire circuit is analyzed over the total simulation time using only one common time step sequence. The number of time step for each subcircuit is thus less in WR decomposition which is a definite computational advantage. Latency of the circuit can be exploited by incorporating bypass techniques. Without losing accuracy the analysis of subcircuits is bypassed for some time intervals knowing the information obtained from the previous time point or previous iteration. The bypass techniques have been described in [NSV84] in detail.

WR methods have guaranteed convergence and have proven to be effective decomposition methods for the analysis of large scale MOS circuits. However, they do suffer from a few drawbacks. The

$k \leftarrow 0$;

guess waveform $\mathbf{v}^0(t) \quad \forall t \in [0,T]$ such that $\mathbf{v}^0(0) = \mathbf{V}$;

repeat

$k \leftarrow k+1$;

for each i solve

$$\sum_{j=1}^{i} C_{i,j}(v_1^k, \ldots, v_i^k, v_{i+1}^{k-1}, \ldots, v_N^{k-1}, u)\ \dot{v}_j^k +$$

$$\sum_{j=i+1}^{n} C_{i,j}(v_1^k, \ldots, v_i^k, v_{i+1}^{k-1}, \ldots, v_N^{k-1}, u)\ \dot{v}_j^{k-1} +$$

$$q_i(v_1^k, \ldots, v_i^k, v_{i+1}^{k-1}, \ldots, v_N^{k-1}, u) = 0$$

for $v_i^k(t)$, $t \in [0,T]$ with initial condition $v_i^k(0) = V_i$

until $\max_{1 \le i \le n} \max_{t \in [0,T]} \mid v_i^k(t) - v_i^{k-1}(t) \mid < \varepsilon$

Figure 3: Algorithm WR–GS.

waveforms of the unknowns at the current iteration have to be stored for computing the waveforms at the next iteration. For large circuits the amount of storage required can be very large. Another problem crops up when there is a logic feedback between the decomposed subcircuits. The speed of convergence of the WR algorithm in such a case becomes very slow unless a good initial guess for the unknown variables is provided. The problem of storage in WR methods can be overcome by dividing the simulation time interval into "windows", $[0,T_1],[T_1,T_2],\ldots[T_{n-1},T_n]$. WR is applied to the first window, $[0,T_1]$ and the values of the node voltages at T_1 are used as the initial conditions for the analysis of the second window. This procedure is repeated until all the windows have been analyzed. This approach helps in rapid convergence.

An obvious advantage of the relaxation algorithms is that they are amenable to parallel implementation. The solution of each node is effectively decoupled from the others and it is possible to allocate a separate processor for each decoupled node equation. If the circuit has been partitioned into subcircuits, they can be analyzed concurrently on different processors. Special purpose hardware can be designed to suit the algorithm. With this brief introduction to the circuit simulation methods, the next section is a survey of the attempts made to speedup circuit simulation using different parallel architectures

4 Need for Parallel Processing

Parallel processing is a technological imperative of computation in VLSI CAD. The economics of VLSI fabrication ensures that parallel computing systems will dominate serial systems in both absolute performance cost. Processing speed is a major concern in circuit simulation. As multiprocessors serve to decrease the program runtime, parallel processing for circuit simulation is a natural evolutionary step. Attempts have been made to implement both the direct and relaxation methods for circuit simulation on parallel architectures. Before such an implementation is attempted, it is necessary to ensure that the algorithm maps well on the architecture to derive the best utilization of the processors. Pipelined architectures [WW86], bus-based architectures [JNP86], hypercube [Mat86], crossbar and multistage switch networks [JNP86] have been used in the past for circuit simulation. The nature of the algorithm shows that relaxation methods are more amenable to parallel implementation that direct methods. This follows because iterative methods have decoupling inherent in them. However, relaxation-based simulators like SPLICE and RELAX have been found to be inappropriate for tightly-coupled circuits due to the convergence

problems encountered in such circuits.

Relaxation algorithms have conflicting requirements when implemented to exploit parallelism. These simulators solve the subcircuits in parallel. Partitioning the circuit to form subcircuits has to be done judiciously so that within a subcircuit tight coupling exists. This ensures that few iterations are required to converge to a solution. However, putting the tightly-coupled nodes in one sub-circuit might lead to a small number of large subcircuits. This situation however, limits the extent of parallelism available. In contrast large number of circuits with few nodes per sub-circuit result in increased parallelism but at the same time number of iterations required for a solution increases as the tightly-coupled nodes may not be together in the same sub-circuit. A balance has to be struck between the number of subcircuits and the number of iterations. Each of the subcircuits is solved independently in parallel on a processor using the direct method.

As even the relaxation algorithm uses direct methods for solving the subcircuits efforts have been directed towards parallelizing direct methods in addition to the relaxation methods. As elucidated in an earlier section, the direct method of circuit simulation involves integrating the set of nonlinear ordinary differential equations modeling the circuit based on the fastest changing circuit variable (this is the variable that requires smallest number of time step for convergence). The time step for the direct method is governed by this variable. The determination of the fastens changing variable can be done in parallel by allocating a set of nodes to each processor to compute the time step [JNP86] for each node and hence the time step for the direct method analysis. The resulting set of nonlinear algebraic equations is solved using N-R iterative technique. Each iteration involves finding the linear equivalent circuit for all nonlinear elements. This again can be done concurrently on a number of processors. The set of sparse linear equations so obtained is decomposed into subcircuits and these subcircuits are solved in parallel using GE method.

Direct method has been implemented on the bus-based Sequent Balance computer, Omega network, cross-bar switch, and BBN Butterfly [JNP86]. Circuit simulation on circuits with upto 50 nodes has been shown by *Jacob et al.* in [JNP86] to have an efficiency of nearly 45% with upto 8 processors. Large circuits are expected to yield better performance because the amount of time spent in contention for shared memory will reduce. It is observed as reported in [JNP86] that as the number of processors increases to the thousands, the algorithm that works on smaller multiprocessors break down due to contention for various system resources. The approach that has been suggested by *Jacob et al.* in [JNP86] is to use clusters of processors to solve smaller parts of the problem (that is, the subcircuits) and then solve between the clusters for the solution of the overall circuit. A hierarchical arrangement of multiprocessors which will grow in a regular fashion to simulate progressively larger circuits has been proposed in [JNP86].

The problem of long runtimes required by SPICE has brought about vectorization of circuit simulation using supercomputers as reported in [MITM87]. The two most time consuming parts of SPICE, namely, computation of the circuit matrix elements and solution of sparse linear equations have been vectorized. *Mikami et al.* have shown [MITM87] that if the vectorized solution of linear equations is ten times faster than its scalar version, it results in a simulator that is 1.1 times faster that its scalar version [MITM87]. The computation time of SPICE based circuit simulation is found to be reduced to one-eighth of the scalar computation time.

The direct method implemented on a SIMD architecture has been found to result in a speedup between 5–7.7 for small and medium sized circuits as shown by *Vladimirescu et al.* in [Vla87]. SIMD avoid problems related to synchronization of different processors. The task of evaluating the models for circuit devices like MOSFETs, and bipolar junction transistors is parallelized. However, the update of the Jacobian matrix and linear equation are implemented as sequential processes only.

The potential of the relaxation methods for parallel processing has motivated implementation of the method on pipelined and multiprocessor architectures. The natural circuit decomposition available in relaxation techniques can be exploited for its parallel implementation. The use of GJ iterative methods provides ample parallelism to the relaxation algorithm. In this method, the relaxation algorithm makes use of the waveforms computed at

the previous iteration for all the subcircuits. All the subcircuits can then be analyzed independently by different processors. The drawback of GJ method is that it is slow in convergence.

The major problem encountered in parallelizing the WR algorithm is that MOS digital circuits are highly directional. It is important to follow the directionality when performing the relaxation computation, otherwise the WR method becomes inefficient. Many iterations are required for convergence if the computation does not follow the signal flow. In the case of large digital circuits, the output of gates have usually more than one fan out and so it is possible to order the computation so that subcircuit can be computed in parallel, but the directionality of the circuit can still be followed by the relaxation computation. Though this limits the parallelism available, it preserves the efficiency of the method.

It is possible to parallelize the WR algorithm while preserving a strict ordering of computation of the subcircuit waveforms by pipelining the waveform computation. In [WW86] *White and Weiner* adopt an approach where the circuit is divided into a number of subcircuits. The first processor starts computing the transient response of a subcircuit for one time point. After the computation corresponding to the first time point is over, the second processor starts computing the response for the first time point for the second subcircuit. At next step, a third processor starts computations for the first time point for the third subcircuit and so on. This is an instance of time point pipelining and has been implemented on a Sequent Balance 8000 computer with a single bus shared memory system as elucidated in [WW86]. The timepoint pipelining algorithm makes efficient use of the available processors. *White et al.* have observed that this algorithm running on the Balance 8000 runs substantially faster than the serial WR algorithm running on a VAX/780 [Whi85]. Thus, an EPROM with 348 FETs take 212 s on VAX/780 whereas pipelined implementation with 9 processors takes 182 s.

Some amount of parallelism can be achieved by using GS iterative method also though it is sequential in nature. *Saleh* has described the implementation of the WR algorithm using GS iterations on different number of processors in [Sal87]. The circuit is presented as a graph with directed edges. The nodes represent the subcircuits and the edges represent the connection between the subcircuits. The directed edges between the nodes give the precedence relation between the tasks. The width of the graph is the maximum size of any independent subset of tasks and these tasks can be solved in parallel. So, all subcircuits at the same level in the graph are computed in parallel for the same iteration. This approach has been found to be good for circuits having wide graphs. Reasonable speedup is possible over the uniprocessor version if the circuit is large enough and only a small number of processors are available. This method proves to be ineffective on circuits with narrow graphs and does not give significant advantages over the GJ method.

CONCISE, a GJ relaxation-based circuit simulator, implemented on a hypercube by *Mattisson* [Mat86] promises to give a very good performance for circuit simulation which takes large fraction of the computing cycles on many high performance computers. In [Mat86] a point relaxation has been used. The circuit simulation program is mapped onto the cube by partitioning the Jacobian matrix **A** into concurrent processes. The linear equation solution phase, that is, Jacobi iteration, involves considerable communication between the processors. The N-R linearization step is completely decoupled and so is concurrently executed. The performance of CONCISE program for different test circuits and implementation details are given in [Mat86].

The hypercube in [Mat86] is, however, modeled by a concurrent program that does not take into account architectural features of a hypercube like the message passing scheme of communication. We have come up with an implementation of the WR algorithm for circuit simulation on a hypercube in HIRECS [Ram88]. Where as CONCISE in [Mat86] makes use of point relaxation, HIRECS is based on block relaxation. A novel circuit partitioning approach based on the heuristic method of Simulated Annealing has been used in HIRECS. An additional feature of HIRECS is that it models all the architectural features of the hypercube. HIRECS has been simulated using the programming language SIMULA on a DEC 1090 system. The speedup obtained from the simulation study for different test

Test Circuit	Number of Processors	Speedup	Efficiency
Inverter	2	1.8	90%
	4	3.1	77%
	8	5.9	73.7%
Multiplexer	2	1.85	92%
	4	3.6	90%
	8	6.0	75%

Table 1: Performance of HIRECS.

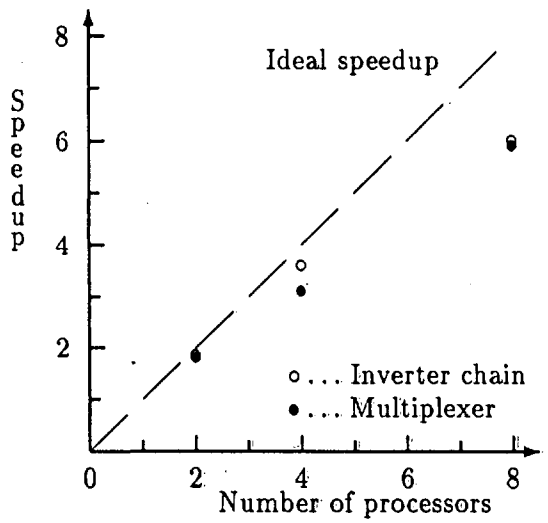

Figure 4: Speedup vs. Number of Processors.

circuits is presented in Table 1 and the performance curves are illustrated in Fig.4. The ideal speedup and the speedup actually obtained for the test circuits are shown in Fig.4. We have observed that for fewer number of processors, a near-linear speedup is possible. as the number of processors increases, the communication time of among the processors increases thereby reducing the speedup. This is because in HIRECS synchronization of the processors is carried out after all the variables of the subcircuits allocated to the processors have converged at all time points over window. Hence the time for which a processor waits till all the others have finished computation over the window also increases with the number of processors. Due to lack of circuit data, HIRECS could not be tested for circuits with large number of nodes. However, we except it to perform equally well for large circuit also.

Improvements in computer architecture allow large circuits to be run without any change in the simulation techniques. Circuit size continue to increase with the progress in technology and existing computer architectures are reaching their performance limits due to constraints on the fundamental speed of light. This has prompted development of an experimental relaxation-based circuit simulator on a massively parallel processor (MPP) – the Connection Machine reported by *Webber et al.* in [WSV87]. The Connection Machine is an MPP with upto 65,536 processors and uses SIMD architecture. The simulator in [WSV87] uses GJ iteration at the nonlinear equation level with point relaxation and a single step of N-R method. Though point relaxation causes slow convergence, it has been found to work well for large class of circuits. Block relaxation is not found to be suitable on the Connection Machine. This is because the data is less uniform in block methods. The matrices to be solved may be of different sizes which make it difficult to exploit the parallelism on the Connection Machine. For an EPROM circuit, the point relaxation on the Connection Machine has been experimentally found by *Webber et al.* [WSV87] to be 30 times faster than the direct method on a MicroVax. The results show that the execution is nearly independent of the size of the problem for circuits without tight coupling. Connection Machine is good for very large problems though it is extremely slow for small problems. The largest circuit that can be run on the Connection Machine is about 10,000 nodes.

5 Conclusion

In this paper we have surveyed the two well known methods for circuit simulation – direct and relaxation. The parallel implementation of these meth-

ods has been considered. The architectures used for the simulation problem as reported in the literature and the observations from our experiments have been presented. It follows from the discussion that considerably higher performance can be achieved by using a special-purpose multiprocessor in which the interconnection of the processors and the design of processors are turned to the circuit simulation task. This is particularly true for the relaxation-based algorithms. Present research work includes finding good partitioning schemes for dividing the circuit into tightly coupled subcircuits, investigation of optimal techniques for finding the simulation time steps and mapping the algorithms to the best possible hardware.

References

[CA79] D. A. Calahan and W. G. Ames. Vector Processors: Models and Application. *IEEE Trans. Circuits Syst.*, CAS-26(9), September 1979.

[CGK75] B. R. Chawla, H. K. Gummel, and P. Kozak. MOTIS – An MOS Timing Simulator. *IEEE Trans. Circuits Syst.*, CAS-22(12):901–909, December 1975.

[DN84] J. T. Deutsch and A. R. Newton. A Multiprocessor Implementation of Relaxation-Based Electrical Circuit Simulation. In *Proc. 21th Design Automation Conf.*, pages 350–357, 1984.

[HHL82] M. H. Heydemann, G. D. Hachtel, and M. Lightner. Implementation Issues in Multiple Delay Switch Level Simulation. In *Proc. Int'l Conf. on Circ. and Comp.*, pages 46–52, September 1982.

[JNP86] G. K. Jacob, A. R. Newton, and D. O. Pederson. Direct Method Circuit Simulation using Multiprocessors. In *IEEE Proc. ISCAS*, 1986.

[LRSV82] E. Lelarasmee, A. E. Ruehli, and A. L. Sangiovanni-Vincentelli. The Waveform Relaxation Method for the Time-Domain Analysis of Large Scale Integrated Circuits. *IEEE Trans. Computer-Aided Design*, CAD-1(3):131–145, August 1982.

[Mat86] S. Mattison. CONSINE – A Simulation Program on Hypercube. Technical report, Lund University, 1986.

[MITM87] M. Mikami, J. Ishibashi, N. Tahara, and G. Matsuoka. Vectorization of SPICE Circuit Simulator on FACOM VP Series Supercomputer. In *Proc. 2nd Int'l Conf. on Supercomputing*, pages 29–34, 1987.

[Nag75] L. W. Nagel. SPICE2: A Computer Program to Simulate Semiconductor Circuits. Technical Report Memo.No.ERL-M 520, UCB, Berkeley, May 1975.

[NP78] A. R. Newton and D. O. Pederson. Analysis Time, Accuracy and Memory Requirement Tradeoffs in SPICE2. In *IEEE Proc. ISCAS*, pages 6–9, 1978.

[NSV84] A.R. Newton and A.L. Sangiovanni-Vincentelli. Relaxation-Based Electrical Simulation. *IEEE Trans. Computer-Aided Design*, CAD-3(4):308–331, October 1984.

[Ram88] S. Raman. HIRECS: Hypercube Implementation of Relaxation-Based Circuit Simulation. Master's thesis, Dept. of Computer Science and Automation, Indian Institute of Science, Bangalore, India, July 1988.

[Sal87] R. A. Saleh et al. Parallel Waveform Newton Algorithms for Circuit Simulation. In *Proc. ICCD*, pages 660–663, 1987.

[ST75] S. A. Szygenda and E. W. Thompson. Digital Logic Simulation in a Time-Based, Table-Driven Environment: Part 1 Design Verification. *IEEE Computer*, 7(3):24–36, March 1975.

[Vla87] A. Vladimirescu et al. A Vector Hardware Accelerator with Circuit Simulation Emphasis. In *Proc. 24th Design Automation Conf.*, pages 90–94, 1987.

[Whi85] J. White et al. Accelerating Relaxation Algorithms for Circuit Simulation using Waveform Newton, Iterative Step Size Refinement, and Parallel Techniques. In *Proc. ICCAD*, pages 5–7, 1985.

[WJM+73] W. T. Weeks, A. J. Jimenez, G. W. Mahoney, D. Mehta, H. Qassemzadeh, and T. R. Scott. Algorithm for ASTAP – A Network Analysis Program. *IEEE Trans. Circuit Theory*, CT-20(11):624–628, November 1973.

[WSV87] D. M. Webber and A. L. Sangiovanni-Vincentelli. Circuit Simulation on the Connection Machine. In *Proc. 24th Design Automation Conf.*, pages 108–113, 1987.

[WW86] J. White and N. Weiner. Parallelizing Circuit Simulation – A Combined Algorithmic and Specialized Hardware Approach. In *Proc. ICCD*, pages 438–441, 1986.