Proceedings of ISCAS 85

PARTITIONING ALGORITHMS AND PARALLEL IMPLEMENTATIONS OF WAVEFORM RELAXATION ALGORITHMS FOR CIRCUIT SIMULATION

Jacob White and A.L. Sangiovanni-Vincentelli
Department of EECS
University of California at Berkeley

Abstract

Since the first applications of Waveform Relaxation (WR) algorithms to the transient analysis of MOS digital circuits, several modifications to the basic method have been used to improve WR efficiency and insure convergence. In particular, convergence problems related to tight feedback loops and inaccurate subcircuit solutions have been addressed and solved with the use of theoretically justified techniques such as dynamic windowing and adaptive error control. However, in order to get reasonably rapid convergence of the the WR method, it is also necessary to partition a large circuit into loosely coupled subcircuits. In this paper a numerically based partitioning method is presented that attempts to break a large circuit into loosely coupled subcircuits by examining estimates for the speed of convergence of the relaxation iteration for candidate partitions. This new partitioning technique has been implemented in the WR based circuit simulator RELAX2.3 and results from that program are presented. Finally, the implementation of the WR algorithm on parallel processors will be presented.

1. INTRODUCTION

The tremendous increase in complexity of circuit design and availability of computing resources has made computer simulation an important and heavily used tool for VLSI design. When accurate computation of the behavior of a VLSI circuit is important to a designer, the only reliable choice is to construct a large system of nonlinear ordinary differential equations (ODE's) that accurately describe the circuit, and use some numerical method to solve the system.

However, the direct numerical techniques used in programs like SPICE[1] can become inefficient for large systems where different state variables are changing at very different rates. This is because direct techniques force every differential equation in the system to be discretized in time identically, and this discretization must be fine enough so that the fastest changing state variable in the system is accurately represented. If it were possible to pick different discretization points, or timesteps, for each differential equation in the system, so that each could use the largest timestep that would accurately reflect the behavior of its associated state variable, then the efficiency of the simulation would be greatly improved.

The Waveform Relaxation (WR) algorithm is a technique that allows this kind of multirate integration. In this method the ODE system that describes a large circuit is first decomposed into possibly many loosely coupled subsystems. The solutions to the ODE subsystems, or waveforms, are calculated by "guessing" the behavior of the surrounding subsystems. The computed waveforms for each of the subsystems are used to improve these guesses, and then the subsystem waveforms are recalculated. The procedure is iterated until the waveforms converge.

Experience simulating MOS digital circuits using the RELAX2.1 program uncovered several problems with the WR algorithm. The first problem was that some circuits converged in a very nonuniform manner, and took many WR iterations to converge, where the number of iterations required was proportional to the length of the simulation interval. Another problem was that the numerical method used to compute the WR iteration waveforms in some cases did not produce sufficiently accurate answers to guarantee WR convergence. Also, WR, as all relaxation techniques, is not efficient when the relaxation is carried out across tightly coupled subcircuits. In SPLICE1[5], and RELAX2, the user had to to carefully decompose his circuit into loosely coupled subcircuits before starting the simulation. This procedure required some expertise on the part of the user, as well as a great deal of patience.

The latest version of RELAX2, RELAX2.3, addresses all these problems. An adaptive algorithm is used to break up long simulations into several shorter intervals, chosen so that the WR algorithm converges rapidly over each window[4]; a more accurate integration method with adaptive error control is used to insure WR convergence[6]; and a hierarchical input processor combined with a circuit partitioner that automatically breaks large circuits into subcircuits. This allows the user to specify his circuit in whatever hierarchical fashion is desired.

The RELAX2.3 program also has more advanced models than the original RELAX2 program. The user can select the first-order Schichmann-Hodges resistive model with a first-order charge-conserving Yang-Chatterjee charge model, or the more accurate full Yang-Chatterjee model which includes short channel and subthreshold conduction effects[7]. In addition, a sparse matrix package was added to improve the program efficiency when solving large subcircuits, and to support a DC solver for efficiently computing intial conditions.

The paper is organized as follows. In Section 2 we present the basic WR algorithm; in Section 3 we describe the new algorithm used in RELAX2.3 for decomposing a large circuit into subcircuits; in Section 4 we examine the effectiveness of the WR algorithm by analyzing several realistic examples. Finally, in Section 5 we present some of the most recent work in adapting the WR algorithm to parallel processors.

2. THE BASIC WR ALGORITHM

The RELAX2.3 program uses the "Gauss-Seidel" WR algorithm, one of the family of WR methods[3]. For that reason, we will only discuss the "Gauss-Seidel" algorithm although many of the following results extend to other members of the WR family.

To describe the algorithm, we will consider it as applied to the following general system, in which MOS circuit equations can usually be formulated.

$$C(v,u)\dot{v} + f(v,u)=0 ; \quad v(0)=V \tag{2.1}$$

where $C:\mathbb{R}^n \times \mathbb{R}^r \to \mathbb{R}^{n \times n}$ is a diagonally dominant matrix-value function in which $-C_{ij}(v,u); i \neq j$ is the total floating capacitance between nodes i and j, $C_{ii}(v,u)$ is the sum of the capacitances of all capacitors connected to node i, and $f:\mathbb{R}^l \times \mathbb{R}^n \times \mathbb{R}^r \to \mathbb{R}^n$ is a continuous function each component of which represents the net current charging the capacitors at each node due to transistors, other conductive elements, and the controlled current sources.

The WR algorithm for solving the above system is as follows:

<u>Algorithm 1 (Gauss-Seidel WR for solving Eqn. (2.1))</u>

```
Comment:
    The superscript $k$ denotes the iteration count,
    the subscript $i$ denotes the component index
    of a vector and $\epsilon$ is a small positive number.
$k \leftarrow 0$ ;
Guess waveform $v^0(t)$ ; $t \in [0,T]$ such that $v^0(0) = v_0$
```

```
repeat {
    $k \leftarrow k+1$
    foreach ( $i$ in $N$ ) {
        solve
        $\sum_{j=1}^{i} C_{ij}(v_1^k, \cdots, v_i^k, v_{i+1}^{k-1}, \cdots, v_n^{k-1}, u)\dot{v}_j^k +$
        $\sum_{j=i+1}^{n} C_{ij}(v_1^k, \cdots, v_i^k, v_{i+1}^{k-1}, \cdots, v_n^{k-1}, u)\dot{v}_j^{k-1} +$
        $f_i(v_1^k, \cdots, v_i^k, v_{i+1}^{k-1}, \cdots, v_n^{k-1}, u) = 0$
        for ( $v_i^k(t) ; t \in [0,T]$ ), given $v_i^k(0) = v_{i_0}$ .
    }
} until ( $\max_{1 \leqslant i \leqslant n} \max_{t \in [0,T]} | v_i^k(t) - v_i^{k-1} | \leqslant \epsilon$ )
```

Note that the differential equation has only one unknown variable v_i^k. The variables $v_{i+1}^{k-1}, \cdots, v_n^{k-1}$ are known from the previous iteration and the variables $v_1^k, \cdots, v_{i-1}^k$ have already been computed.

In [3], the WR algorithm was show to have the following guaranteed convergence property.

Theorem 1: Given a system of equations of the form of Eqn. (2.1) generated by applying nodal analysis to a circuit containing capacitors, resistors, and MOS devices, if the MOS devices are voltage-controlled and have continously differentiable charge models with diagonally dominant Jacobians, and there is a grounded capacitor, linear or nonlinear, at every node, then in Eqn. (2.1) $C(v,u) \in \mathbb{R}^n$ is strictly diagonally dominant for all $v \in \mathbb{R}^n$ and Lipschitz continuous with respect to v for all u, f is Lipschitz continuous with respect to v for all u, and the sequence $\{ v^k \}$ generated by the Gauss-Seidel WR algorithm given above converges uniformly to the solution of Eqn (2.1) for all bounded intervals $[0,T]$.

In the above theorem it was proved that the WR method is a contraction map in the following nonuniform norm on $C([0,T],\mathbb{R}^n)$.

$$\max_{[0,T]} e^{-bt} \| f(t) \|$$

where $b > 0$, $f(t) \in \mathbb{R}^n$, and $\| \bullet \|$ is a norm on $\mathbb{R}^n$. Note that convergence in this norm can be achieved if at each iteration the error is reduced over larger and larger intervals of time. In some cases the WR iteration waveforms converge in just this way.

Although this "nonuniform" convergence guarantees that the WR method is robust, it does not insure computational efficiency. In fact, the computations performed on the last part of the time-interval are essentially wasted. To have a better computational behavior, the WR algorithm should converge "uniformly", i.e., the error should be reduced over the entire time-interval at each iteration.

Consider the following definition:

Definition 1: WR Uniform Iteration Factor Let $v^k : [0,T] \rightarrow \mathbb{R}^n$ be the function generated by the k^{th} iteration of the WR algorithm applied to a system of the form of Eqn. (2.1). Then the *WR uniform iteration factor*, γ, for the system is defined as the smallest positive number such that

$$\max_{[0,T]} \| v^{k+1}(t) - v^k(t) \| \leqslant \gamma \max_{[0,T]} \| v^k(t) - v^{k-1}(t) \|$$

for any $k > 0$, any continously differentiable initial guess v^0, and any piecewise continuous input u.

In order for the WR algorithm to converge in a computationally efficient way, the WR uniform iteration factor should be less than 1. This would imply a "uniform" convergence i.e., the computed iteration waveform is approaching the exact solution over the entire interval.

There are two ways to reduce γ. The first, discussed in [4,6], is to reduce the simulation interval $[0,T]$ until γ is less than one. The second approach, which we will consider in the next section, is to partition the circuit into loosely coupled subsystems. A combination of the two techniques is needed to allow windows of reasonable size.

3. PARTITIONING

In the basic WR algorithm presented above, the node equations are solved as single differential equations in one unknown, and these solutions are iterated until convergence. This kind of node-by-node decomposition can lead to slow convergence in the case where a few nodes in a large system are tightly coupled. For example, consider the simple circuit in Fig. 1. If the floating resistor connecting the two nodes is large relative to the grounded resistors, then the WR algorithm using node-by-node decomposition converges rapidly. However, if the floating resistance is small, then the convergence will be very slow.

As the example above shows, not lumping together tightly coupled nodes and solving them directly can lead to very slow WR convergence. For this reason, the first step of an effective WR-based simulator must be to partition the system, that is, to scan all the nodes in the system and determine which should be lumped together and solved directly. Partitioning well is difficult for several reasons. If too many nodes are lumped together, the advantages of using relaxation will be lost, but if any tightly coupled nodes are not lumped together then the WR algorithm will converge very slowly. In addition, it is

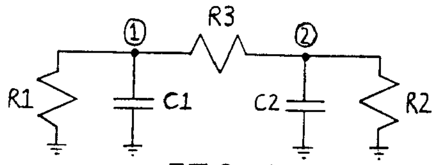

FIG. 1

obviously important that the partitioning step not be computationally burdensome.

Several approaches have been applied to this partitioning problem. One simple approach is to require the user to partition the system [3,5,8]. This technique is reasonable for the simulation of large circuits because large circuits usually are designed as a collection of small, fairly independent pieces as this makes the design easier to understand and manage. Unfortunately, what is a sensible partitioning from a design point of view may not be a good partitioning for the WR algorithm.

Another approach to partitioning is to examine the topology of the circuit to identify functional blocks (i.e. a nand gate or a flip-flop)[9]. The nodes of each functional block are then tied together. This type of partitioning is difficult to perform, since the algorithm must recognize broad classes of functional blocks, and nonstandard blocks may not be treated properly. In addition, as the example at the beginning of this section indicates, techniques that examine only the topology of the circuit may partition the circuit incorrectly.

Since it is the intent of the partitioning to improve the speed of convergence of the relaxation, it is sensible to partition a large circuit with this, rather than topology or functionality, in mind. In this section we will develop an algorithm based on this idea. As it is difficult to get estimates of the speed of WR convergence directly, a relationship will be shown between the convergence speed of WR and that of two simpler iterations. We will then present techniques for estimating the speed of convergence for the simpler iterations, and show how they are used to perform partitioning.

The WR uniform iteration factor γ defined in the previous section is a lower bound on how fast the iteration waveforms approach the exact solution. A good partitioning algorithm will keep this γ small without generating unnecessarily large subcircuits. Unofrtunately, γ is difficult to estimate directly for a given problem. However, we have the following theorems which relate γ to iteration factors applied to a simplified system of equations.

Theorem 2: Let γ be the WR uniform iteration factor for a given system of equations of the form of Eqn. (2.1) solved on [0,T]. Then in the limit as $T \rightarrow \infty$, γ is bounded below by the relaxation iteration factor of the nonlinear Gauss-Seidel relaxation applied to the reduced algebraic system $f(v,u)=0$, for $v \in \mathbb{R}^n$ given any $u \in \mathbb{R}^n$.

Theorem 3: Let γ be the WR uniform iteration factor for a given system of equations of the form of Eqn. (2.1). Then γ is bounded below by the relaxation iteration factor of the linear Gauss-Seidel relaxation applied to the reduced algebraic problem $C(y,u)v = b$, for $v \in \mathbb{R}^n$ given any $y, u, b \in \mathbb{R}^n$.

Since in Eqn. (2.1) $C(v,u)$ is the matrix of linear and nonlinear capacitors, and $f(v,u)$ is the net circuit currents generated by conductances, the two theorems above indicate that it is possible to get lower bound estimates of γ by examining circuits where only the capacitances and conductances are independently present. These estimates are lower bounds. Hence, to decrease γ below a desired α, it is *necessary* to partition in such a way that the iteration factors for the Gauss-Seidel iteration applied to reduced systems are decreased below α.

In order to develop an algorithm for estimating the iteration factor for the conductance problem, we will start with a simple problem for which we can calculate the iteration factor exactly. We will then apply this result, by analogy, to larger problems. Consider a simple three-resistor circuit derived by removing the capacitors from the circuit in Fig. 1. If Gauss-Seidel relaxation is used to solve this two-node problem then the iteration equations can be written by inspection as:

$$v_1^{k+1} = \frac{g_3}{(g_1+g_3)} v_2^k \qquad v_2^{k+1} = \frac{g_3}{(g_2+g_3)} v_1^{k+1}$$

The iteration factor is:

$$\gamma = \frac{g_3}{(g_2+g_3)} \frac{g_3}{(g_1+g_3)}$$

We use this result to devise a heuristic for larger circuits. The circuit in Fig. 1 can be used as a simple model for the coupling between two nodes in a MOS circuit. With this model the MOS transistor will be treated as a nonlinear resistor from its source to its drain. Coupling between the gate and source and gate and drain will be considered later. With this simplification, we can use the following algorithm for partitioning circuits with two-terminal resistances.

Algorithm 2 (Conductance Partitioning)

```
for each ( conductive element in the circuit ) {
    $g_3$ ← maximum element conductance over all $v$ .
    Remove the element from the circuit.
    Replace the rest of the conductances in the circuit
        by their minimum values over all $v$ .
    Compute $g_1$ and $g_2$, the Norton Equivalent
      conductances to ground at the two element terminals.

    If ( $\frac{g_3}{(g_2+g_3)} \frac{g_3}{(g_1+g_3)} > \alpha$ ) {
        Tie the two terminal nodes together.
    }
}
```

Computing the Norton equivalent conductances, *Geq*, at a node can be performed using a simple recursive formula if there exist no loops of conductances among only non-voltage source nodes. Note that this recursion will not be very deep. The recursion will stop at any MOS transistor, because the model for the conductance of an MOS transistor at this point in the computation is zero.

Algorithm 3 (Norton Equivalent Conductance for Node i)

```
$Geq = 0.0$
foreach ( conductive element incident at node i ) {
    G ← element conductance
    node j ← the conductive element's other node.
    If ( node j is a voltage source node) {
        $Geq \leftarrow Geq + G$
    }
    else {
        $Geq j$ ← Norton equivalent conductance at
                node j with this element removed.
        $Geq \leftarrow Geq + (G * Geq j)/(G + Geq j)$
    }
}
```

If the circuit does contain conductance loops among only non-voltage source nodes, the above algorithm can still be used if the recursion is truncated in such a way that no circuit node is visited twice. In this case, only an estimate of the Norton equivalent will be computed.

Unfortunately, it is difficult to relate the parameter α to the iteration factor for the WR, or even the iteration factor for the conductance problem. We loosely justify its use by the following theorem which applies to a special case of one step of the algorithm.

Theorem 4: If removing the conductive element under consideration breaks the conductance circuit into two subcircuits that are connected only at common voltage sources nodes then the iteration factor computed in Alg. 2 is an upper bound on the iteration factor for solving the conductance problem by relaxation across the two subcircuits.

Since the capacitance problem is almost identical to the conductance problem, the capacitance partitioning algorithm follows almost the same strategy as the conductance partitioning. The major difference is that instead of comparing floating capacitances to Norton equivalent conductances, they are compared to equivalent capacitances. These equivalent capacitances are entirely analogous to the equivalent conductances, and can be computed with the same recursive approach used in Alg. 4.

The last step of the partitioning algorithm, is to take into account the gate-voltage controlled current source between source and drain. Since the controlled source is a one-way element, it is only necessary to take these into account when they form feedback loops in the circuit. Adding in this last step, we arrive (finally) at the RELAX2.3 partitioning algorithm.

Algorithm 4 (RELAX2.3 Partitioning Algorithm)

```
Apply Conductance Partitioning
Apply Capacitance Partitioning
repeat { /* Find nested tight feedback loops. */
  for each gate node of a MOS transistor:
    If that transistor's drain node or source node
    is contained in group of nodes which contains
    a gate node of another transistor whose drain
    or source node is in the first transistor's
    group, tie the two groups together
}
until (no more groups are tied together)
```

Along with applying the partitioning algorithm to a variety of MOS digital circuits, we have also applied this partitioning algorithm to a large VHSIC memory circuit with 2900 nodes and over 3500 parasitic components. The results matched our own best attempts to partition the circuit in as many instances as we had the patience to check. However, we still suspect that if the method is applied to larger problems, the subcircuits produced may become quite large. Should this be the case, we plan to extend the present algorithm by performing an additional pass over only the excessively large subcircuits, and subpartitioning them using more sophisticated algorithms. In particular, to use better estimates of the equivalent conductances and capacitances, as we feel the present algorithm may be unnecessarily conservative.

4. EXPERIMENTAL RESULTS USING RELAX2.3

RELAX2.3 has been applied to several industrial circuits and compared to SPICE2G.6. Both programs use the Shichman-Hodges drain current MOS model. SPICE2G.6 uses the Meyer capacitance model, and the RELAX2.3 program uses a level1 Yang-Chatterjee charge conserving model. In an attempt to separate the gains attributable to a faster implementation of direct methods versus the gains attributable to WR, we also compared the running times of RELAX run in "SPICE-mode", i.e., when no partitioning is applied, with the WR-mode. Some of the circuits presented were run at Texas Instruments where RELAX was compared to the version of SPICE modified by Texas Instrument. In these runs, both programs use the complete Yang-Chatterjee MOS model.

Circuit	Devices	SPICE2	RELAX2.3(DIR)	RELAX2.3(WR)
uP Control	232	1400s*	90s*	45s*
Cmos Memory	621	10400s*	995s*	308s*
4-bit Counter	259	4300s*	540s*	299s*
Digital Filter	1082	18000s*	1800s*	520s*
Encode-Decode	3295	115,000s*	5000s*	1500s*
Eprom	348	694**	531**	349**
Inverter Chain	250	439s**	98s**	38s**
VHSIC Memory	2980	15670**	15830**	12650**

*On Vax11/780/Unix using S-H model
**On Vax11/780/VMS using Y-C model and TISpice

5. PARALLEL WAVEFORM RELAXATION ALGORTIHMS

Relaxation techniques are particularly promising for parallel processing because the computational method decomposes naturally the problem. Recent results from a parallel implementation of the Iterated-Timing Analysis (ITA) method for simulating large circuits indicates that significant parallelism is exploitable when relaxation methods are used[10].

The WR algorithm is unique among relaxation methods in that each time a subcircuit is solved, its transient behavior is computed for a large portion of the simulation interval, rather than just for a short increment in time. For this reason, WR seems to be a very promising relaxation method for parallel processors because the overhead to set up the subcircuit for simulation, often a major bottleneck for parallel computations, is an insignificant portion of the time required to calculate the long transient response waveform.

An obvious way of parallelizing WR is to apply the Gauss-Jacobi version of WR. In this algorithm, the relaxation makes use of the waveforms computed at the previous iteration for all the subcircuits. Then, all the subcircuits could be analyzed independently by different processors. One of the difficulties in applying this algorithm is that MOS digital circuits are highly directional, and, if this directionality is not exploited slow convergence may result. For example, consider applying WR to compute the transient response of a chain of inverters. If the first inverter's output is computed first, and the result is used to compute the second inverter's output, which is then used for the third inverter, etc., the resulting waveforms for this first iteration of the WR algorithm will be very close to the correct solution. However, if the second and third inverter outputs are computed in parallel with the first inverter's output, the results will not be close to the correct solution because no reasonable guess for the inverter inputs will be available. For this reason, after partitioning, RELAX2.3 orders the subcircuits so that the directionality of the circuit is followed as closely as possible.

It is possible to parallelize the WR algorithm while still preserving a strict ordering of the computation of the subcircuit waveforms (Gauss-Seidel) by pipelining the waveform computation. In this approach, one processor would start computing the transient response for a subcircuit. Once a first timepoint is generated, a second processor could begin computing the first timepoint for the second subcircuit, while the first processor computes the second timepoint for the first subcircuit. On the next step a third processor could be introduced, to compute the first timepoint for the third subcircuit, and so on. By doing so, some of the advantage of WR for parallel processors is lost. Scheduling must be done every time a timepoint is computed, rather than over a whole waveform.

Since following a strict ordering of the relaxation computation (Gauss-Seidel) does not allow for simple parallelism, and computing the next iteration waveforms for every subcircuit at once (Gauss-Jacobi) allows for greater parallelism, but is not very efficient (converges more slowly), we consider a mixed scheme. The approach is based on the observation that large digital circuits tend to be quite "wide", i.e., the outputs of gates fan out to more than one subcircuit. It is possible to order the computation so that subcircuits in parallel "chains" can be computed in parallel, but the directionality of the circuit can still be followed by the relaxation computation. This will not allow for as much parallelism as the Gauss-Jacobi scheme, but should preserve most of the efficiency of the Gauss-Seidel scheme.

We are experimenting with a probabilistic approach to attempting to follow the ordering of the subcircuits. In this approach a queue of subcircuits in order is created. Then as processors become free, they grab the next subcircuit in the queue until the queue is exhausted. If the queue is empty and all the processors are finished, the queue is reset, and they all start picking up subcircuits again.

This algorithm is probabilistic in the sense that there is no guarantee that the transient computation for a subcircuit will be finished before its output is needed by another subcircuit, but it is likely to have already finished if the circuit is very wide compared to the number of processors. And since all the processors must finish before the next iteration is begun, no subcircuit will be more than one iteration behind. We will be implementing this algorithm on the Sequent Balance 8000 parallel computer, and will have experimental results to present at the Conference.

6. CONCLUSIONS AND ACKNOWLEDGEMENTS

The newest feature of the RELAX2 series of programs, a more general partitioner, has been described. In addition, some of the issues connected with parallelizing WR have been presented. Expansion of the scope of the RELAX2 program continues, bipolar circuits, more advanced mosfet models, and relaxation acceleration techniques are being added. A parallel version of the RELAX2.3 program is under developement, and will be tested on a 32-bit microprocessor-based multicomputer, the Sequent Balance 8000.

The authors wish to thank Richard Newton, J. Deutsch, and R. Saleh for the many discussions on relaxation-based techniques and parallel processing, K. Kundert for the sparse matrix package and P. Moore for the input processor used in Relax2.3, and all the other members of the CAD Research group at Berkeley for their support. In addition, we would like to thank F. Odeh and A. Ruehli at IBM's Yorktown Research group for many valuable discussions.

This research has been sponsored by DARPA under contract NESC-N39, and grants from the IBM company, Philips, Texas Instruments and MICRO.

REFERENCES

[1] L.W. Nagel, "SPICE2: A computer program to simulate semiconductor circuits," Electronics Research Laboratory *Rep. No. ERL-M520, University of California, Berkeley*, May 1975.
[2] C.W. Gear, "*Numerical Initial Value Problems for Ordinary Differential Equations*," Prentice Hall, 1974.
[3] E. Lelarasmee, A. E. Ruehli, A. L. Sangiovanni-Vincentelli, "The waveform relaxation method for time domain analysis of large scale integrated circuits," *IEEE Trans. on CAD of IC and Syst.*, Vol. 1, n. 3, pp.131-145, July 1982.
[4] J. White and A. L. Sangiovanni-Vincentelli, "RELAX2: A Modified Waveform Relaxation Approach to the Simluation of MOS Digital Circuits" *Conf. Proc. IEEE ISCAS*, Vol. 2, pp756-759, Newport Beach, CA, May, 1982.
[5] A. R. Newton, "The Simulation of Large Scale Integrated Circuits", *Memorandum UCB/ERL M78/52*, July 1978.
[6] J. White and A. Sangiovanni-Vincentelli, "Relax2.1 - A Waveform Relaxation Based Circuit Simulation Program" *Proc. 1984 Int. Custom Integrated Circuits Conference* Rochester, New York, June 1984.
[7] P. Yang and P. Chatterjee; "SPICE Modeling for Small Geometry MOSFET Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. CAD-1, No. 4, October 1982, pp.169-182
[8] P. Defebve, J. Beetem, W. Donath, H.Y. Hsieh, F. Odeh, A.E. Ruehli, P.K. Wolff, Sr., and J. White, "A Large-Scale Mosfet Circuit Analyzer Based on Waveform Relaxation" *International Conference on Computer Design*, Rye, New York, October 1984.
[9] C. H. Carlin and A. Vachoux, "On Partitioning for Waveform Relaxation Time-Domain Analysis of VLSI Circuits" *Proc. 1984 Int. Symp. on Circ. and Syst.*, Montreal, Canada, May 1984.
[10] J. T. Deutsch "MSPLICE" University of California, Berkeley, Electronics Research Laboratory, Memorandum No. UCB/ERL-M84/40