RELAX2.1: A WAVEFORM RELAXATION BASED CIRCUIT SIMULATION PROGRAM

Jacob White and A.L. Sangiovanni-Vincentelli

Department of EECS

University of California at Berkeley

Berkeley, CA 94720

Abstract

Waveform Relaxation (WR) algorithms have been proven to be effective in the transient analysis of large scale integrated circuits. The latest version of the RELAX series of waveform relaxation simulators for MOS digital circuits, RELAX2.1, is described. The newest version includes an automatic partitioning of circuits, a better waveform error control method to improve WR convergence properties, and a dynamic windowing scheme to increase convergence speed in the case of circuits with logic feedback. Results comparing RELAX2.1 and SPICE2 runtimes simulating industrial circuits are presented.

1. INTRODUCTION

VLSI circuits, which often contain more than 50,000 devices, can not be economically simulated with computer programs that calculate circuit transient response by numerically integrating a complete set of simultaneous nonlinear differential equations. This method is general, but the computational complexity grows too rapidly as the number of devices increases. SPICE[9], a program which uses this approach, can take several hours (on a VAX11/780) to simulate circuits with only a few hundred devices.

Several approaches have been developed that reduce the computation time required to perform accurate simulation by exploiting the structure of certain classes of problems. Most of these approaches have focused on digital MOS circuits, and have exploited the mostly unidirectional properties of MOS transistors and the latency of digital circuits[1-5]. In this paper we will describe one particular method, Waveform Relaxation(WR)[1], and discuss its implementation in the RELAX2.1 program. In the following two sections we will describe the basic WR algorithm, and discuss some of the previous work on waveform methods. In Section 4, we will give an overview of the RELAX2.1 program and describe in detail the partitioning algorithm used to decompose large circuits, the circuit simulator, and the dynamic windowing system. Finally, in Section 5, we will present results and conclusions.

2. THE BASIC WR ALGORITHM

The RELAX2.1 program uses the "Gauss-Seidel" WR algorithm, one of a family of WR methods[1]. To describe the algorithm, we will consider it as applied to the following general system, in which MOS circuit equations can usually be formulated.

$$C(v,u)\dot{v} + f(v,u) = 0; \quad v(0) = V \tag{1.1}$$

where $C: \mathbb{R}^n \times \mathbb{R}^r \to \mathbb{R}^{n \times n}$ is a symmetric 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 $f: \mathbb{R}^r \times \mathbb{R}^n \times \mathbb{R}^r \to \mathbb{R}^n$ is a continuous function each component of which represents the net current charging the capacitor at each node due to the pass transistors, the other conductive elements and the controlled current sources.

Algorithm 1 (WR Gauss-Seidel Algorithm for solving Eqn. (1.2))

Comment:
The superscript k denotes the iteration count, the subscript i denotes the component index of a vector and ε is a small positive number.

```
$k \leftarrow 0$;
guess $v^0(t)$ ; $t \in [0, T]$ so that $v^0(0) = V$
   (for example, set $v^0(t) = V, t \in [0, T]$);

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 $v_i^k(0) = V_i$.

   }
}
until $(\max_{1 \le i \le n} \max_{t \in [0,T]} |v_i^k(t) - v_i^{k-1}(t)| \le \varepsilon)$
that is, until the iteration converges. ▪
```

Note that Eqn. (1.1) 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.

If is assumed that there is a grounded capacitor, linear or nonlinear, at every node in the circuit then it can be shown that the waveforms generated by the Gauss-Seidel WR algorithm *will always converge to the correct waveform independent of the initial guess.* This strong convergence result was proved in [1].

3. PREVIOUS WORK IN WR METHODS

Due to their favorable numerical properties, WR algorithms have captured considerable attention. WR algorithms have been applied to the solution of piecewise-linear differential equations[6], they have been used in mixed-mode simulators[7], and special purpose multi-processor architecture is being studied to implement the WR algorithm.

The Waveform Relaxation algorithm was tested for MOS circuits in a research program RELAX. This version of RELAX accurately computed the transient behavior of large circuits up to 60 times faster than SPICE[8,9]. However, the original RELAX program was limited in the types of circuits it could be used to simulate. A new program, RELAX2[10], was a first attempt to develop a more general circuit simulation program using Waveform Relaxation. RELAX2 handled a broader class of circuits, and was used to experiment with new techniques to improve the convergence and computational efficiency of the WR method[2].

Experience simulating MOS digital circuits using the RELAX2 program uncovered several problems with the WR algorithm. The first problem was that circuits with logic feedback loops (finite state machines, asynchronous circuits, digital oscillators) sometimes converged in a very nonuniform manner, taking many iterations to converge, and the number of iterations required was proportional to the length of the simulation interval.

Another problem with the WR method was that although theorems exist that guarantee the global convergence of the WR algorithm, these theorems require that the node voltage waveforms of the decomposed subsystems be computed exactly. Of course, numerical methods commonly used in circuit simulation programs do not solve differential equations exactly. Instead, a numerical integration method (such as Backward-Euler or the Trapezoidal rule) is used to approximate the original differential system by a sequence of algebraic systems corresponding to a collection of discrete timepoints. For most numerical integration methods, the error in this discretization approximation is a function of the timesteps, and these timesteps are chosen small enough so that the waveforms are computed to some user-supplied accuracy. Unfortunately, if the errors are not controlled in a particular manner, the WR algorithm may not converge.

Finally, the RELAX2 program was difficult to use because it was still necessary for a user to decompose his circuit into subcircuits. This procedure required some expertise on the part of the user, as well as a great deal of patience.

4. THE RELAX2.1 PROGRAM

An improved version of RELAX2, RELAX2.1, solves many of the problems mentioned in the previous section. It uses the BLT(Berkeley Language Translator)[11] input-processor to allow the user to define his circuit in as flat or hierarchical a fashion as desired. The RELAX2.1 program then automatically repartitions the circuit into subcircuits in such a way that the WR algorithm convergences rapidly. The RELAX2.1 program uses a second-order integration method combined with a very careful error control scheme so that non-convergence due to discretization inaccuracies is avoided in most cases. Finally, this new version uses a "dynamic windowing" scheme to retain fast convergence for the case of circuits with logic feedback.

Another advantage of the RELAX2.1 program is that it has more advanced models than the original RELAX2 program. At present the user can select the first-order Schichmann-Hodges model or the more accurate charge conserving Yang-Chatterjee model[12]

4.1. OVERVIEW OF THE RELAX2.1 PROGRAM

In this section we present an brief overview of the steps performed in the RELAX2.1 program when simulating a circuit. A detailed description of the major steps is contained in the sections that follow.

The first step in simulating a circuit using the RELAX2.1 program is to create the circuit description file. In this file a user must specify device model parameters, circuit topology, analysis specifications, and plotting requests. The circuit topology can be described in as hierarchical or flat a form as the user desires. This circuit description file is used as an input to the BLT(Berkeley Language Translator) program. This program performs syntax checking, assigns numbers to all the nodes, and expands any hierarchy. An intermediate file is produced by BLT, and it contains a flattened description of the original circuit, along with the information about the model parameters and simulation requests.

The RELAX2.1 program reads the flattened intermediate file description produced by BLT. Before applying the WR algorithm, the flattened circuit is decomposed into a collection of *subcircuits*. This is done by first partitioning the circuit into clusters of tightly coupled nodes. Then the elements (e.g. transistors, resistors, capacitors) that connect to any of the nodes in a given cluster are gathered together to make the subcircuits. Once the entire circuit has been carved up into subcircuits, the subcircuits are ordered, or scheduled, starting with subcircuits that are connected to the user-defined inputs and then following the natural directionality of the circuit (as much as possible).

After a large circuit has been broken up into subcircuits, and these subcircuits have been ordered, the RELAX2.1 program begins the waveform relaxation process.

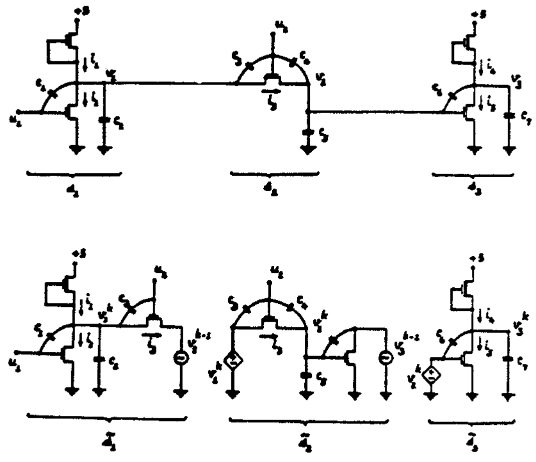


Fig 1- A decomposed Circuit

An initial guess is made for each of the node voltage waveforms. Then a transient analysis is performed on each of the subcircuits in the order determined above. To perform the analysis, those nodes in the subcircuit that where not part of the cluster around which the subcircuit was built, are treated as external time-varying voltage sources (See Fig. 1). The values for the external voltage sources are either the initial guess waveforms, or if the subcircuit containing the external node was simulated previously, that computed waveform. As the node waveforms are computed, they replace the existing waveforms (initial guesses or previous iterations), and the process is repeated until the waveforms converge.

However, the WR algorithm becomes inefficient when used to simulate digital circuits with logical feedback(e.g. finite state machines, ring oscillators, etc.) for many cycles. For this reason the RELAX2.1 program does not actually perform the relaxation iterations by computing the transient behavior of each subcircuit for the entire user-defined simulation interval. Instead, the RELAX2.1 program uses a modified WR algorithm, in which the relaxation is only performed for a small piece of the user-defined simulation interval at a time. Exactly how large a piece of the waveform, refered to as a *window*, to use is determined automatically, at the beginning of every WR iteration.

4.2. 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. As an example, consider the circuit in Fig. 2, a two inverter chain separated by a resistor-capacitor network. In this case the resistor-capacitor network models wiring delay, so the resistor is small compared to the output impedance of the inverter. The WR algorithm using node-by-node decomposition was used to solve this system, as in Fig. 3. The voltage waveforms computed for the capacitor node for the forth, sixth, and ninth iteration of the WR algorithm are plotted in Fig. 4. As is clear from the figures, the WR algorithm converges very slowly in this case. However, if the capacitor node and the output of the first inverter are solved together, convergence is quite rapid, as indicated by Figs. 5 and 6.

As the example above indicates, it greatly accelerates the convergence of the relaxation process if groups of tightly coupled nodes are solved together as one subsystem or subcircuit. For this reason the RELAX2 program groups together tightly coupled nodes into subcircuits before beginning the relaxation process. This partitioning of the original circuit into subcircuits is done in two passes. In the first pass transistor drain and source node pairs, nodes

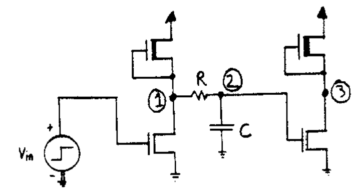


Fig 2 - Inverters with Delay

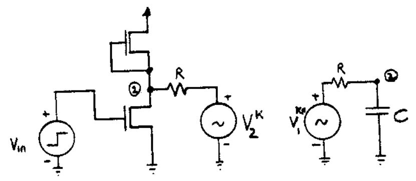


Fig 3 - Decomposed Circuit

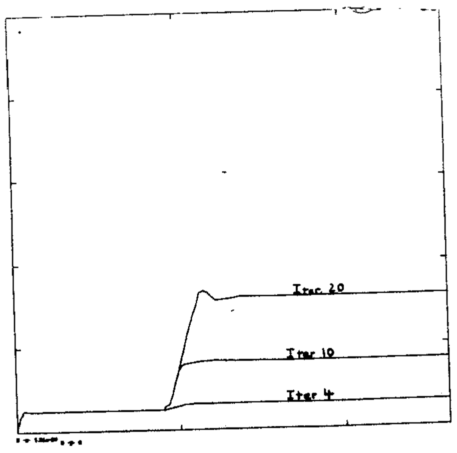


Fig 4 - WR Iterations

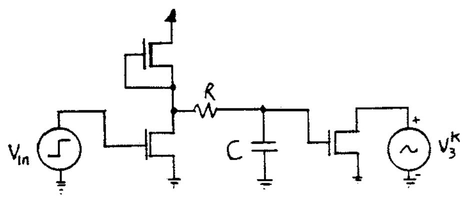


Fig. 5 - Partitioned Circuit

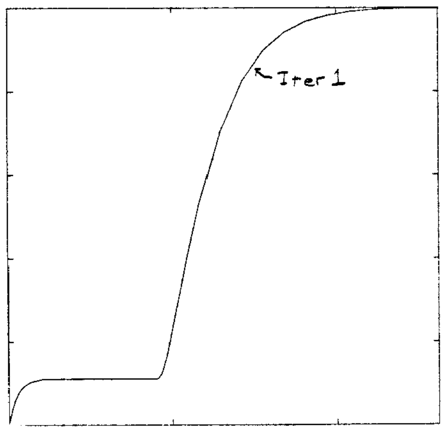


Fig 6 - First Iteration

connected by large floating capacitors, and nodes connected by small floating resistors, are grouped together. In the second pass, nodes connected by tight feedback loops are grouped together. The specific algorithm presently used in RELAX2.1 is the following:

Algorithm 2 (RELAX2 Partitioning Algorithm)

```
Comment:
    First pass - Tie together nodes coupled tightly by elements
        for each MOS transistor:
            group together the drain and source nodes.

        for each diode:
            group together the anode and cathode nodes.

        for each linear resistor:
            if the parallel combination of all the linear
            resistors at either node of the resistor is
            greater than one third the resistor value,
            group together the two resistor nodes.

        for each linear capacitor:
            if the sum of all the linear capacitors at either
            node of the capacitor is less than one third the
            capacitor value group together the capacitor
            nodes.

Comment:
    Second pass - Find tight feedback loops
        repeat {
          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)
```

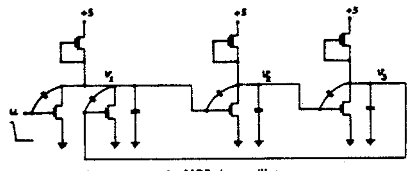
In the above partitioning algorithm, any nodes connected by the source and drain of a transistor are tied together. This may be too conservative, and may not break the circuit up into fine enough subcircuits. So far, the largest subcircuit produced from the industrial circuits we tested contained 23 nodes, and that is not overly large. However, as we apply the method 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 tests to detect tight coupling.

Also, this algorithm is not likely to work well for bipolar circuits, as the base of a bipolar transistor is not as isolated as the gate of an MOS transistor. As we are presently adding bipolar models to RELAX2.1 we will be experimenting with the partitioning algorithm in conjunction with bipolar circuits in the near future.

4.3. WINDOWSIZE DETERMINATION

The WR algorithm used in RELAX2.1 becomes inefficient when used to simulate digital circuits with logical feedback(e.g. finite state machines, ring oscillators, etc.) for many cycles. However, the WR algorithm can still be very efficient if the relaxation is only performed on a piece of the waveform to be computed at a time.

As an example, consider the WR algorithm applied to the ring oscillator in Fig. 7. The voltage waveforms for the output of the first inverter computed by the first, second, and third iteration of the waveform relaxation are graphed in Fig. 8. The graph demonstrates a unique property of the WR algorithm when applied to circuits with logic feedback: the error is not reduced at every time point in the waveform. Instead, each iteration lengthens the interval of time, starting from zero, for which the waveform is correct.



An MOS ring oscillator.

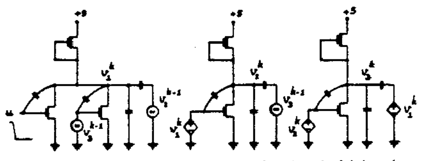


Fig. 7. The decomposition of the circuit in Fig. 4 at the kth iteration of Algorithm 4.1.

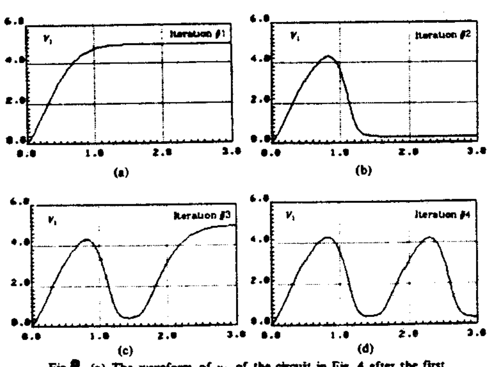


Fig. 8. (a) The waveform of v_1 of the circuit in Fig. 4 after the first WR iteration. (b) The waveform of v_1 of the circuit in Fig. 4 after the second WR iteration. (c) The waveform of v_1 of the circuit in Fig. 4 after the third WR iteration. (d) The waveform of v_1 of the circuit in Fig. 4 after the fourth WR iteration.

If instead, the simulation interval is broken up into "windows" corresponding roughly to the ring oscillator's period of oscillation, and the relaxation is only performed over one "window" at a time, then convergence can be achieved in few iterations per window. There is a trade-off, however. If the windows are too small some of the advantages of waveform relaxation are lost. One cannot take advantage of a digital circuit's natural latency over the entire waveform, but only in that window; the scheduling overhead increases when the windows become smaller, as each circuit lump must be scheduled once for each window; and if the windows are made very small, timesteps chosen to calculate the waveforms will be limited by the window size rather than by the discretization error, and unnecessary calculations will be performed.

In the RELAX2.1 program the "windowsize" is determined dynamically, by two criteria. The first criterion is to pick the windowsize to limit the number of timepoints required to represent each node waveform in a window. This puts a strict *a priori* upper bound on the amount of storage needed for the waveforms, and thus allows the RELAX2.1 program to aviod dynamically allocating and deallocating waveform storage. The second criterion is to try to pick the windowsize so that the convergence of the WR is rapid, so that the waveforms approach the correct solution in a uniform manner over the entire window. The RELAX2.1 program presently uses the following windowsize determination algorithm:

Algorithm 3 (RELAX2 Windowing Algorithm)

```
Comment:
        starttime = Beginning of the window
        stoptime = End of the window
        endtime = End of user-defined simulation interval
        usedpts = Max. # of points used in the last window
        maxpts = Max. # of points in a waveform buffer

    if(Not entirely converged in this window) then {
        Shorten window waveforms overran buffers.
        if(usedpts >= maxpts) then {
          stoptime = starttime +
              (lastwindowsize * maxpts * 0.7)/usedpts;
        }
        Else just do the same window again.
        else {
          stoptime = laststoptime;
        }
    }
    else {
        starttime = stoptime;
        stoptime = laststoptime +
            (lastwindowsize * maxpts * 0.7)/usedpts;
    }
```

As indicated by the above algorithm, we have not incorporated information about how rapidly the WR algorithm is converging in the choice of window size. This is because we have not found an algorithm using rate of convergence information that works better than the algorithm based only on controlling the number of computed timepoints. Also, we have no good way of generating an initial guess for the windowsize. Presently, we are considering adding a simplified critical path analyzer to RELAX2.1 to provide that initial guess.

4.4. TRANSIENT ANALYSIS

Relaxation methods have guaranteed convergence properties when applied to circuits for which there is a grounded linear or nonlinear capacitor at every node. This is required by the RELAX2.1 program, and it will generate an error message and abort if any node is not connected to a grounded capacitor.

Since a grounded capacitor is already required by the RELAX2.1 program to guarantee relaxation convergence, this assumption is exploited heavily in the method for performing the transient analysis. Unlike many standard approaches used in simulators like SPICE[19], the node voltages are guaranteed to be state variables for the differential equations that describe the circuit system. For this reason error control can be performed by considering only the node voltages.

The differential equations for a subcircuit can be formulated using nodal analysis as follows:

$$C(v,u)\dot{v} + f(v,u) = 0; \quad v(0) = V \qquad (4.1)$$

In order to improve the numerical properties of the equation description of the system, charge, rather than voltage, is used as the state variable. This leads to a modified form for the system equations:

$$\dot{q} + \hat{f}(q,u) = 0; \quad q(0) = Q \qquad (4.2)$$

where $q = q(v)$ is such that $\dot{q} = C(v,u)\dot{v}$ and $f(v,u) = \hat{f}(q(v),u)$.

As in standard circuit simulators, the RELAX2.1 program solves Eqn 4.2 using a numerical integration method with varying timesteps. Since the major aim of the RELAX2.1 program is to simulate digital circuits, an integration method that would be efficient for that type of problem was chosen. In general, low order one-step methods are recommended for problems with rapid transitions, like digital circuits. However, first order methods are not accurate enough to insure the waveform relaxation convergence. For these reasons the trapezoidal rule, a second-order one-step method, was chosen.

Given a timestep h, the trapezoidal integration method applied to Eqn. 4.2 yields:

$$q(v(t+h)) - q(v(t)) - \qquad (4.3)$$

$$0.5h(\, f(v(t+h),u) + f(v(t),u)\,) = 0$$

In order to solve the algebraic system generated by Eqn. 4.3, a classical Newton-Raphson method is used. The iteration equation for the Newton-Raphson method for solving $F(x) = 0$ is

$$J_F(x^k)(x^k - x^{k-1}) = -F(x^{k-1}) \qquad (4.4)$$

where J_F is the jacobian of F with respect to x. If the Newton algorithm is used to solve Eqn. 4.3 for $v(t+h)$ the cost function, $F(v(t+h))$, is:

$$F(v(t+h)) = \qquad (4.5)$$

$$q(v(t+h)) - q(v(t)) - 0.5h(\, f(v(t+h),u) + f(v(t),u)\,)$$

and the jacobian of $F(v(t+h))$, $J_F(v(t+h))$ is easily shown to be:

$$J_F(v(t+h)) = C(v(t+h),u) + 0.5h\frac{\partial f}{\partial v}(v(t+h)) \quad (4.6)$$

Although the charge formulation, Eqn. 4.2, is numerically better behaved than the voltage formulation, Eqn. 4.1, using the charge formulation does present a problem in error control. In the WR algorithm, voltage is the relaxation variable, so it must be computed accurately to insure the relaxation convergence. This implies that even though charge is the state variable in the above equations, the integration timesteps should be chosen to computing accurate voltages. This strategy has worked well so far in RELAX2.1, but if very nonlinear charge equations are used for the MOS models, controlling errors in voltage may not be sufficient.

5. EXPERIMENTAL RESULTS USING RELAX2.1

Benchmark comparisons between RELAX2.1 and SPICE2 using industrial circuits and the first-order MOS models are encouraging (See table below), and we hope to provide comparisons between RELAX2.1 and SPICE2 using the more advanced MOS model by the Conference time.

Circuit	Mosfets	Diodes	Nodes	SPICE2	RELAX2.1	Ratio
Ring Osc.	7	0	3	17s*	5.1s*	3.3
uP Control	116	116	66	1400s*	82s*	17
Cmos Memory	344	277	151	10400s*	360s*	28
4-bit counter	259	0	170	4300s*	320s*	14

*On Vax11/780 running Unix

6. CONCLUSIONS AND ACKNOWLEDGEMENTS

Several features of the RELAX2 program have been presented which allow the program to simulate a broad class of MOS digital circuits without requiring numerical expertise on the part of the user. Expansion of the RELAX2.1 program continues, bipolar circuits, more advanced mosfet models, and a dc solver using global newton methods are presently being added. A parallel version of the RELAX2.1 program is under development, and will be tested on a 68000-based multiprocessor.

The algorithmic part of this research has been sponsored DARPA under contract NESC-N39. a grant of the IBM company and MICRO has provided the support for the development of RELAX2.1. We wish to thank R. Saleh, who provided examples and helped debug the RELAX2.1 program. In addition we would like thank G. Casinovi, J. Deutsch, J. Klechner, R. Newton, and the other members of the CAD Research group at Berkeley, as well as J. Beetem, W. Donath, F. Odeh, L. Reyna, and A. Ruehli, at IBM's Yorktown Research group for many valuable discussions and suggestions.

REFERENCES

[1] 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.
[2] B.R. Chawla, H.K. Gummel, and P. Kozah, "MOTIS - an MOS timing simulator," *IEEE Trans. Circuits and Systems*, Vol. 22, pp. 901-909, 1975
[3] A. R. Newton, "The Simulation of Large Scale Integrated Circuits", *Memorandum UCB/ERL M78/52*, July 1978.

[4] P. Yang, I.N. Hajj, and T.N. Trick, "Slate: A Circuit Simulation Program with Latency Exploitation and Node-tearing" *Proc. IEEE Int. Symp. on Circ. and Syst.*, pp. 143-147, 1977
[5] K. Sakallah and S.W. Director, "An Activity-directed Circuit simulation algorithm," *Proc. IEEE ICCC'80 Conf. (Rye, NY)*, pp. 337-340, Oct. 1980.
[6] J. Kaye and A. Sangiovanni-Vincentelli, "Solution of piecewise linear ordinary differential equations using waveform relaxation and Laplace transforms", *IEEE Transactions on Circuits and Systems, IEEE Transactions on Automatic Control, IEEE Transactions on Systems, Man, and Cybernetics Joint Special Issue on Large Scale systems*, pp. 353-357, June. 1983.
[7] H. De Man, "Mixed-Mode Simulation for MOS-VLSI: Why, Where and How?" *Proc. 1982 Int. Symp. on Circ. and Syst.*, pp. 699-701, Rome, Italy, May 1982.
[8] E. Lelarasmee and A. Sangiovanni-Vincentelli, "Relax: a new circuit simulator for large scale MOS integrated circuits", *Proc. 19th Design Automation Conference*, Las Vegas, Nevada, pp. 682-690, June 1982.
[9] L.W. Nagel, "SPICE2: A computer program to simulate semiconductor circuits," Electronics Research Laboratory *Rep. No. ERL-M520, University of California, Berkeley*, May 1975.
[10] 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.
[11] J. D. Crawford, "A Unified Hardware Description Language for CAD Programs," *ERL Memo No. UCB/ERL M79/64*, University of California, Berkeley, May 1978.