Accelerating Relaxation Algorithms for Circuit Simulation Using Waveform-Newton and Step-Size Refinement

RESVE A. SALEH, MEMBER, IEEE, AND JACOB K. WHITE, MEMBER, IEEE

Abstract—**A new relaxation algorithm for circuit simulation that combines the advantages of iterated timing analysis (ITA) and waveform relaxation (WR) is described. The method is based on using an iterative stepsize refinement strategy with a waveform-relaxation-Newton (WRN) algorithm. All three relaxation techniques, ITA, WR, and WRN, are compared and experimental results that indicate the strengths and weaknesses of the methods are presented. In addition, a new convergence proof for the waveform-Newton (WN) method for systems with nonlinear capacitors is provided. Finally, it is shown that the step-refined WRN algorithm can be implemented on a parallel processor in such a way that different subsystems can be processed in parallel and the solution at different timepoints of the same subsystem can also be computed in parallel.**

I. Introduction

THE IMPLICIT multistep integration algorithms used in general-purpose circuit simulation programs, like SPICE2 [1] and ASTAP [2], have proven to be reliable but are computationally expensive when applied to large systems. This is because each step of the numerical integration requires the implicit solution of a large nonlinear algebraic system. If the circuit simulation program is intended for the simulation of mostly MOS digital circuits, then it is possible to exploit the properties of these types of circuits to improve the simulator's efficiency. In particular, the fact that MOS digital circuits can be partitioned into loosely or unidirectionally coupled subsystems can be exploited by using iterative decomposition algorithms, and the fact that the different nodes in MOS digital circuits often change at very different rates can be exploited by using multirate integration techniques [3]-[5].

One common approach to exploiting the unidirectionality and multirate behavior of MOS circuits is to use some kind of relaxation technique (for a comprehensive list of references see [5]). Historically, the first use of relaxation in circuit transient analysis was as a substitute for Newton's method to solve the nonlinear algebraic systems generated by implicit integration methods. Early programs based on this approach [6], [7] attempted only to produce approximate results, and performed only one relaxation iteration for each timestep. The techniques used in these programs were referred to as "timing analysis," and when the relaxation iteration was carried to convergence to produce accurate results, the algorithm was referred to as iterated timing analysis (ITA) [5]. A second approach to using relaxation for circuit transient analysis is to apply the relaxation directly to the differential equation system, replacing the solution of a large differential equation system with the solution of a collection of small differential equation subsystems [8], [9]. This second approach is referred to as waveform relaxation (WR) because the iterates are functions, or waveforms, over the simulation interval.

The ITA algorithm, because it is a relaxation applied after discretization, cannot exploit multirate behavior as easily as the WR algorithm [10]. However, ITA is more efficient in many cases than standard WR. This is partly because typical ITA-based programs use a relaxation-Newton scheme, in which the nonlinear relaxation iteration equations are solved only approximately with a single Newton iteration [11]. In each iteration of WR, the nonlinear differential equations are solved accurately, and this is computationally expensive. Perhaps the next obvious step is to try a relaxation-Newton approach within the WR algorithm. That is, use a single iteration of some kind of waveform-Newton algorithm to approximately solve the WR iteration equations. This approach is the main focus of this paper.

It is reasonably straightforward to derive the waveform-Newton (WN) and waveform-relaxation-Newton (WRN) algorithms [12]-[14], and show that WRN has similar convergence properties to standard WR. In addition, the iteration equations for WRN are time-varying linear differential equations and are easier to solve than the nonlinear differential iteration equations of WR. However, WRN does not prove to be much more efficient than WR when simulating most digital circuit examples, because WR typically converges in fewer than five iterations whereas WRN may take many more.

Manuscript received July 25, 1988; revised July 19, 1989. This work was supported by the Defense Advanced Research Projects Agency under Contracts NESC-N39 and N00014-87-K-825, Hewlett-Packard, NSERC of Canada, and by grants from IBM, Phillips, Texas Instruments, and MICRO.

R. A. Saleh is with the Coordinated Science Laboratory, Department of Electrical and Computer Engineering, University of Illinois, Urbana, IL 61801-3082.

J. K. White is with the Research Laboratory of Electronics and the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA.

IEEE Log Number 9036177.

The fact that the WRN algorithm may take many iterations to converge makes it possible to improve the efficiency of the algorithm by solving the iteration equations crudely at first, and then increasing the accuracy with each iteration [15]. With WRN, it is particularly efficient to use coarse timesteps in the early iterations, and then refine the step sizes as the iterations approach convergence. Note that such a technique is not so helpful with WR because WR usually converges so rapidly that there are not enough iterations to do meaningful refinement. Also, the Newton method used at each timestep in the WR iterations may not converge if very large timesteps are used.

In this paper, we describe the WRN algorithm in detail and provide simulation results of an implementation of WRN with iterative stepsize refinement in the SPLAX program. We begin in the next section by presenting the WN algorithm [16] and describing some of its limitations, as well as presenting the WRN algorithm. In Section III, the WRN algorithm with iterative timestep refinement strategy is described, and its suitability for parallel computation is mentioned. In Section IV, we compare the WRN algorithm in our SPLAX program with the ITA method used in the SPLICE program [10], [17] and the WR algorithm used in the RELAX2 program [8], [18]. Experimental results that indicate the strengths and weaknesses of the three techniques are presented.

II. Waveform-Relaxation Newton (WRN)

The differential equations that describe a typical MOS digital circuit can be constructed using nodal analysis [19], and such an approach leads to a system of n equations of the form:

$$\frac{d}{dt} q(v(t), u(t)) = g(v(t), u(t)) \tag{1}$$

where $q(v(t), u(t)) \epsilon \mathbb{R}^n$ is the vector of sums of capacitive charges at each node, $g(v(t), u(t)) \epsilon \mathbb{R}^n$ is the vector of sums of resistive currents at each node, $u(t) \epsilon \mathbb{R}^m$ is the vector of input voltages, $v(t) \epsilon \mathbb{R}^n$ is the vector of unknown node voltages, and n is the number of circuit nodes excluding the reference node.

A function-space iterative Newton method [16] can be used to solve (1). The idea is that the differential equation is linearized about an initial guess waveform whose value at time zero matches the given initial condition. Then, the guess waveform is updated by solving the resulting linearized differential equation (with the same initial conditions at time zero). The original differential equation is then relinearized about the updated guess, and the process is repeated until convergence is achieved. In particular, solving (1) can be thought of as finding a waveform $v(t)$ such that

$$(F(v))(t) \equiv \frac{d}{dt} q(v(t), u(t)) - g(v(t)) \equiv 0. \tag{2}$$

The Jacobian of $F(v)$, $J_F(v)$, is then given by the Frechet derivative as

$$(J_F(v)\delta)(t) = \frac{d}{dt}\left[\frac{\partial q(v(t), u(t))}{\partial v}\delta(t)\right] - \frac{\partial g(v(t), u(t))}{\partial v}\delta(t). \tag{3}$$

This leads to an iteration update differential equation given by

$$\frac{d}{dt}\left[q(v^k(t), u(t)) + \frac{\partial q(v^k(t), u(t))}{\partial v}\delta^{k+1}(t)\right] = g(v^k(t), u(t)) + \frac{\partial g(v^k(t), u(t))}{\partial v}\delta^{k+1}(t) \tag{4}$$

where $\delta^{k+1}(t) = v^{k+1}(t) - v^k(t)$ [18]. Note that if $v^k(0) = v_0$ then $\delta(0) = 0$.

As WN is just the function-space extension of the classical Newton–Raphson algorithm, it will converge *quadratically* when the iterated value is close to the correct solution [16]. The WN algorithm also has the attractive property that it converges *globally* when applied to equations of the form of (1), given mild assumptions about the behavior of the charge and current functions, $q(v)$ and $g(v)$, and provided that the initial guess waveforms match the initial conditions for the differential equations. In particular, we have the following theorem about the convergence of the WN algorithm, the proof of which is outlined in the appendix.

Theorem 1: If a system is of the form of (1) in which $\partial q / \partial v$ is differentiable, Lipschitz continuous, and has a uniformly bounded inverse with respect to v for all u; g is differentiable and Lipschitz continuous; and $v^0(t)$ is a continuous, differentiable function such that $v^0(0) = v_0$; then the sequence of waveforms, $\{v^k\}$, generated by the WN algorithm converges uniformly to the solution of (1) ■.

Proof 1: See Appendix A.

2.1. Limitations of WN

The iteration equations for the WN algorithm are time-varying linear differential equations that are easier to solve numerically than the original system of nonlinear differential equations. For example, if the WN iteration equations are solved with the same discretization techniques as used for the classical direct method, then since the equations are linear, only a single matrix solution need be performed at each timepoint. Also, linear time-varying systems can be solved with a variety of efficient numerical techniques other than the standard discretization methods, such as collocation and spectral methods [12].

Note that the discretized WN algorithm is as ill-suited to the simulation of large problems as the classical direct methods, because it still requires the solution of large linear equation systems, and cannot easily exploit multirate behavior. However, as the discretized WN algorithm and the classical direct methods are of such a similar nature, it is perhaps useful to compare their relative efficiencies. Clearly, for linear problems, the classical direct and the discretized WN methods are identical. However, the discretized WN can be much less efficient than classical direct methods when used to simulate circuits containing highly nonlinear elements such as diodes. This is due to the difference in how the classical and WN methods obtain the initial guesses for their respective Newton methods. In the classical approach, generating an initial guess for the Newton method implies projecting the behavior of the system forward by one timestep. In WN, one projects the behavior of the system forward for an entire waveform. Therefore, in the classical method the initial guess is almost certainly in or near the region of quadratic convergence for the Newton method, whereas that would rarely be true for WN.

To demonstrate this difficulty with WN, consider the problem illustrated in Fig. 1, a simple resistor–diode circuit with a grounded capacitor. Fig. 2 shows the waveform iterations obtained using WN to solve the circuit, given an initial guess of $v^0(t) = 0$ for all $t\epsilon[0, 1]$. Note that the first computed waveform $v^1(t)$ is quite far from the correct solution, and subsequent iterations move very slowly back to the correct solution. This slow convergence is common when applying Newton methods to exponential nonlinearities, given a poor initial guess. In this case, over 50 iterations are necessary to achieve satisfactory convergence.

A wide variety of limiting techniques can be used to improve the convergence of WN in these situations. For example, the change in the waveform from one iteration to the next could be limited, in a way analogous to step limiting in the standard algebraic Newton method [1]. There is a well-known strategy that is particularly useful for MOS circuits, where the only diodes in the circuit are associated with the source-to-substrate and drain-to-substrate junctions, and are usually reverse-biased. Instead of using the correct derivative of the diode current with respect to the voltage across the diode, the derivative can be approximated by the diode current divided by the voltage across the diode, an approximation known as "line-through-the-origin." Since only the Jacobian is altered, if the so created pseudo-Newton method converges, it converges to the correct result. The danger is that if the diode is substantially forward biased, an unlikely event for MOS circuits, this pseudo-Newton method may not converge no matter how close the initial guess.

2.2. *The WRN Algorithm*

One useful application of the WN algorithm, and the main focus in this paper, is to combine it with the WR algorithm to construct the waveform extension of the relaxation-Newton algorithm, as shown in Algorithm 1.

Algorithm 1 - (WRN Gauss–Seidel Algorithm)

The superscript k denotes the iteration count, the subscript $i\epsilon\{1, \cdots, N\}$ denotes the component index of a vector, ϵ is a small positive number, and $v^{k+1,i} = [v_i^{k+1}, \cdots, v_{i-1}^{k+1}, v_i^k; \cdots, v_n^k]^T$. $k \leftarrow 0$.
Guess waveform $v^0(t)$; $t\epsilon[0, T]$ such that $v^0(0) = v_0$ (for example, set $v^0(t) = v_0$, $t\epsilon[0, T]$;

```
repeat {
    $k \leftarrow k + 1$
    for all ($i$ in $N$) {
        solve
            $\frac{d}{dt}[q_i(v^{k+1,i}(t), u(t)) + \frac{\partial q_i(v^{k+1,i}(t), u(t))}{\partial v_i}(v_i^{k+1}(t) - v_i^k(t))]$
            $- [g_i(v^{k+1,i}(t), u(t)) + \frac{\partial g_i(v^{k+1,i}(t), u(t))}{\partial v_i}(v_i^{k+1}(t) - v_i^{k+1}(t))] = 0$
        for ($v_i^{k+1}(t)$; $t\epsilon[0, T]$, with the initial condition
        $v_i^{k+1}(0) = v_{i0}$.
    {
} until ($\| v^{k+1} - v^k \| \leq \epsilon$).
```

Note that, in the above WRN algorithm, the WR iteration equations are solved approximately by performing one step of the WN method with each waveform relaxation iteration. This is analogous to the single Newton iteration strategy of the nonlinear relaxation methods used in ITA. Also, like the WR algorithm, each equation in Algorithm 1 is a differential equation in one unknown variable v_i^k but, in this case, the nonlinear differential iteration equations have been replaced by simpler time-varying linear differential equations.

Given the global convergence properties of both the original WR and the WN algorithms, it is not surprising

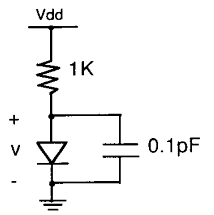


Fig. 1. A simple resistor-diode example.

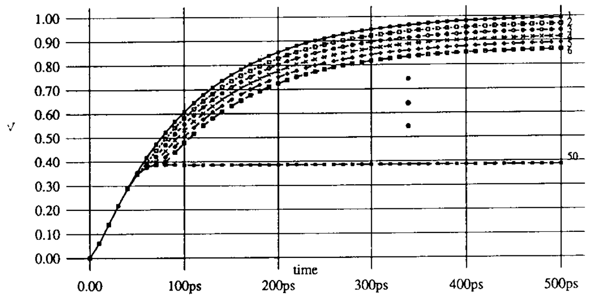


Fig. 2. WN iterations for resistor-diode example.

that the WRN algorithm has global convergence properties. The proof for this is quite similar to the proof of the WR and WN convergence theorems and is not included here.

Theorem 2: If the assumptions in Theorem 1 are satisfied, then the sequence $\{v^k\}$ generated by the WRN algorithm converges to the solution of (1) on any bounded interval $[0, T]$. ■

III. Timestep Control Strategy for WRN

As mentioned above, the amount of computation performed in the early iterations of the WRN can be reduced by using coarse numerical integration timesteps to solve the differential relaxation equations initially, and then refining the timesteps as the iterations progress. Specifically, the first relaxation iteration is computed with a user-supplied maximum allowed numerical integration timestep. For subsequent relaxation iterations, the integration timesteps are chosen to be the same as those in the previous iteration unless the *a posteriori* local truncation error estimate for the timestep from the previous iteration is too large. In that case, half the previous iteration timestep is used.

To demonstrate this idea, consider the sequence of waveforms in Fig. 3. For the first iteration the maximum timestep is used and this produces the waveform shown in Fig. 3(a). In this case, the computed LTE at T is too large and, therefore, on the second iteration, the window interval is divided in half and two time-steps are taken, as shown in Fig. 3(b). In the second iteration, the LTE is too large at time point $T/2$ but acceptable at time point T. Therefore, on the third iteration, only the first half interval is divided, as shown in Fig. 3(c).

The iterative refinement strategy has the advantage that, in general, the timesteps will be placed more efficiently to control the truncation error than if the standard predicted truncation error criteria is used. This is because the timestep selection is based only on more accurate *a posteriori* error estimates available from previous relaxation iterations. However, there are situations where too many time points will be placed. In particular, if, in some region of time, a "wavefront" moves through the interval as the iterations progress, many timesteps will be placed in the wavefront's path. This behavior is illustrated in Fig. 4 where the waveform for a particular node in a small circuit is shown after three different iterations. During the early iterations, the steps near the beginning of the window interval are refined due to the transition. The rest of the interval is not adjusted since a large step is appropriate when the waveform is latent. However, on subsequent iterations, the transition moves ahead in time, and therefore, the timesteps are refined locally to capture it. Unfortunately, the points placed near the beginning of the window are still recomputed. This problem gets progressively worse as the transition proceeds to the end of the window, as shown in the third graph. One way to overcome this problem is to *remove* time points on subsequent

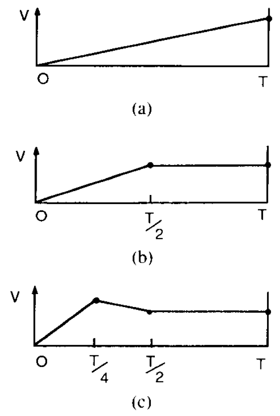


Fig. 3. WRN time-step control.

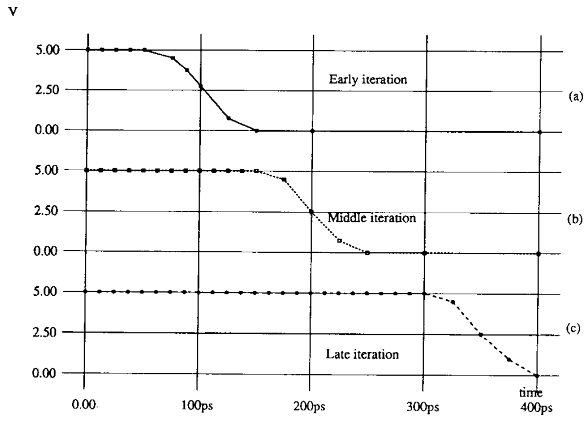


Fig. 4. Wavefront propagation during iterations.

iterations if the LTE is very small at a particular time point. This is not a trivial optimization as such an approach can be unstable. That is, time points which are added on one iteration may be removed on the next iteration, and then added again on the third iteration, etc. However, a careful implementation of this approach would reduce the overall computation in these situations.

3.1. Parallelizing WRN with Timestep Refinement

The WRN algorithm combined with the timestep refinement strategy also has several advantages that make it appropriate for use on parallel processors [20]. For any decomposition method, the decomposed subsystems can be solved independently on parallel processors [14]. Also, as with any waveform relaxation method, solving the decomposed subsystems is a significant computation involving numerically integrating the independent differential equations over some interval of time. The WRN algorithm has an additional advantage for parallel computation. Since the discretizations times are selected by a timestep refinement strategy, they are, therefore, known *a priori* (i.e., before beginning the calculation of the next iteration waveform). Once the discretization times are known, the linearized system at each of the discrete times can be computed in parallel since the iteration update equations for WRN are based on linearization about a previous iteration waveform, once the discretization times are known, the linearized system at each of the discrete times can be computed in parallel. The result is that not only can the decomposed subsystems be computed in parallel, but most of the computation for each of the timesteps for each of the decomposed subsystems can be computed in parallel.

In order to see how the timestep parallelism can be exploited, consider using WN to solve the simple equation

$$\dot{x}(t) = f(x(t)) \quad x(0) = x_0 \tag{5}$$

where $x(t) \in \mathbb{R}^n$, and $m: \mathbb{R}^n \rightarrow \mathbb{R}^n$. The WN iteration equation is then

$$\dot{x}^{k+1}(t) = f(x^k(t)) + \frac{\partial f(x^k(t))}{\partial x}(x^{k+1}(t) - x^k(t))$$
$$x^{k+1}(0) = x^k(0) = x_0 \tag{6}$$

where $x^k(t)$ is the kth Newton iterate. Discretizing (6) with backward-Euler leads to a sequence of M linear equations, where M is the number of timesteps. The equation that must be solved to compute $x^{k+1}(\tau_j)$, where τ_j is the time of the jth timestep, is given by

$$\frac{1}{\tau_j - \tau_{j-1}}(x^{k+1}(\tau_j) - x^{k+1}(\tau_{j-1})) = f(x^k(\tau_j)) + \frac{\partial f(x^k(\tau_j))}{\partial x}(x^{k+1}(\tau_j) - x^k(\tau_j)). \tag{7}$$

Reorganizing (7) leads to

$$\left[\frac{1}{\tau_j - \tau_{j-1}} I - \frac{\partial f(x^k(\tau_j))}{\partial x}\right](x^{k+1}(\tau_j)) = \frac{1}{\tau_j - \tau_{j-1}} x^{k+1}(\tau_{j-1}) + f(x(\tau_j)) - \frac{\partial f(x^k(\tau_j))}{\partial x} x^k(\tau_j) \tag{8}$$

where I is the identity matrix in $\mathbb{R}^n$. The parallelism that can be exploited by simultaneously evaluating M timesteps can be seen by examining (8). Clearly, the evaluation and LU decomposition of the matrix on the left-hand side of (8), and the computation of all the terms on the right-hand side of (8) except the $(1/\tau_j - \tau_{j-1})x^{k+1}(\tau_{j-1})$ term can be performed for all M timesteps simultaneously.

Once the parallelizable portion of (8) is complete, there is still some computation that must be performed serially to update the $x^{k+1}(\tau_j)$ values. This serial computation involves a matrix backsolve to compute $x^{k+1}(\tau_1)$, which is then used to complete the right-hand side for the equation for $x^{k+1}(\tau_2)$, from which $x^{k+1}(\tau_2)$ can be computed with a matrix backsolve, which is then used to complete the right-hand side of the equation for $x^{k+1}(\tau_3)$, etc. In total, this serial section involves M matrix backsolves and $M - 1$ multiplication and additions to complete the right-hand sides. However, the backsolves are by far the smallest part of the timestep calculation, so this serial section is not too computationally expensive, and therefore, processing timesteps in parallel can be effective. The initial results of an implementation of this approach are quite promising and the interested reader may obtain further information in [20].

IV. Program Implementation and Simulation Results

In this section, we compare the basic algorithms used in the three programs, SPLICE3 [10], based on ITA, RELAX2 [18], based on waveform relaxation, and a new program, SPLAX [14], based on WRN with iterative timestep refinement. In order to make meaningful comparisons of the effectiveness of the three algorithms, we tried to keep the algorithms used in the programs as close to each other as possible. All three programs are written in C, and use the same algorithms to first partition a large problem into loosely coupled subcircuits, and then order the subcircuits for the relaxation process.

There are some unavoidable differences. Both SPLAX and RELAX2 use waveform relaxation-based algorithms and, because of the nonuniform way in which WR converges, it is much more efficient if the simulation interval $[0, T]$ is broken into subintervals, or windows, $[0, t_1]$, $[t_1, t_2]$, $\cdots$, $[t_n, T]$. It is difficult to determine a good window size *a priori*, and the two programs do not use the same strategy for picking these windows. In RELAX2, an adaptive windowing algorithm is used, as the interaction between window size and converge speed is reasonably well understood for WR. However, in the case of WRN, a good window size tends to be a function of the equation nonlinearities rather than the differential equation dynamics. To minimize convergence problems due to equation nonlinearities, the SPLAX program uses relatively small window sizes compared to RELAX2. The selection of optimal window sizes is still an "open" research area, although the heuristics used in SPLAX work well in practice.

Also, the SPLAX program uses the second-order backward difference formula rather than the trapezoidal rule to numerically integrate the iteration equations. This is because the iterative timestep refinement technique used in SPLAX can use very large timesteps for early WRN iterations, and if the trapezoidal method were used, it would produce spurious oscillatory solutions (see [1]). Although the timestep refinement would eventually remove the spurious oscillation, it may occasionally cost a few extra relaxation iterations. This would seem to put the SPLAX program at a slight disadvantage, as the second-order backward difference method does have larger truncation error than the trapezoidal method, and would normally require more timesteps for comparable accuracy. This does not seem to be the case however, perhaps because the backward-difference method is being used in combination with the *a posteriori* LTE control described above.

4.1. Simulation Results

In Table I, the classical direct methods and the ITA, standard WR, and WRN methods are compared using a number of example circuits. The examples are: a critical path from a microprocessor control circuit, *Microc*, the logic for a successive approximation register, *Scdac*, a dynamic memory cross section, *Dram*, a static memory cross section, *Sram*, and a digital filter, *Digfi*. In *Microc*, the waveforms exhibit multirate behavior, mainly in the form of latency, and the tradeoffs in WR and ITA balance to produce similar run times, but WRN is slightly faster than either method since it exploits multirate behavior completely using relatively inexpensive iterations. The circuit is too small for relaxation methods to be of significant benefit though, and the classical direct method performs best. The circuits *Scdac* and *Dram* exhibit latency and are coupled, and therefore, ITA is faster than WR, but WRN with its combined benefits is again faster than either. The WR algorithm is fastest for the circuit *Digfi*, because the partitioner breaks the circuit into completely unidirectional blocks, and therefore, WR converges in one iteration. Note that for *Digfi*, WRN is six times slower than WR, and WRN proved to be even slower than direct methods! In *Sram*, WR is again the fastest, because although there is some localized coupling due to the gate–source and gate–drain capacitance (which are not included in the *Digfi* example), *Sram* is essentially unidirectional.

The above results imply that, while WRN is faster for realistic MOS circuits with local bidirectional coupling, it is not very effective when used to simulate idealized MOS circuits which ignore gate–source and gate–drain capacitances. To provide stronger evidence of this behavior, *Sram* was simulated a number of times with a range of gate–source and gate–drain capacitances, as controlled by the thin-oxide thickness, *tox*, values. As Table II demonstrates, as the value of *tox* increases, and therefore, the gate–source and gate–drain capacitance decreases, the ratio of the runtime of WRN to WR also increases.

An area of future work suggested by these results is to consider an algorithm which combines both the standard WR and WRN methods into one simulator. The choice of which method to use for a given circuit would then be based on the unidirectionality and linearity characteristics of the subcircuits that comprise the circuit. The portions of the circuit that contain elements with highly nonlinear device characteristics or exhibit predominantly unidirectional signal flow could be solved using WR while the remaining portions that feature moderate coupling or weakly nonlinear device characteristics could be solved using WRN. This modified WR method is expected to be even more effective as it exploits the advantages of standard WR and WRN. However, such an approach will require a method to automatically select the appropriate algorithm for each portion of the circuit and this will likely be the key research activity for this composite algorithm.

TABLE I
DIRECT VERSUS SPLICE3 VERSUS RELAX2 VERSUS SPLAX

name	Nodes	Direct	,ITA	WR	WRN
Microc	56	31.3	47.1	47.7	41.8
Scdac	150	290.1	302.1	355.8	278.5
Dram	300	650.2	582.3	861.7	535.9
Sram	129	333.9	238.7	178.8	331.3
Digfi	378	641.1	323.8	167.0	749.5

TABLE II
RUNTIME RATIOS OF SPLAX TO RELAX2 AS *tox* VARIES

tox	125Å	250Å	500Å	1000Å	∞
Splax/Relax2	1.7	1.85	2.3	2.4	8.9

V. CONCLUSIONS

A new circuit simulation algorithm, based on combining WRN with an iterative step-size refinement scheme, has been described and implemented in the SPLAX program. In addition, a new convergence proof for the WN has been presented. It has been shown experimentally that the new algorithm is more effective than the standard WR method when simulating realistic MOS digital circuits, and it out performs ITA when simulating moderately coupled, multirate circuits. However, it is not as well suited to the simulation of unidirectional circuits, for which the standard WR is more appropriate, and has difficulty on certain highly nonlinear problems. The results presented in this paper also identify the key strengths and weaknesses of each of the three relaxation-based methods and the classical direct methods. One area of future research is to choose the most appropriate method for the simulation of a given circuit based on static information provided by the partitioner, such as the number of subcircuits, their sizes, and the degree of coupling between them, and dynamic information, such as the amount of latency and multirate behavior exhibited over previous runs and the nature of the device nonlinearities in the circuit. This is one of the key activities in our current research work. Secondly, the WRN algorithm has certain features that can be exploited on parallel processors and this aspect provides an additional advantage over other methods. Our preliminary results using this approach are very encouraging and the effective use of parallel processors to further speedup circuit simulation is another area of focus in our ongoing research on parallel relaxation-based circuit simulation.

APPENDIX
PROOF OF THEOREM 1

The detailed proof of Theorem 1 is somewhat complicated. To avoid an unnecessarily lengthy presentation, but still convey where the necessary conditions stem from, several not quite obvious lemmas will presented without proof.

The following norm on the space of functions that map $[0, T] \to \mathbb{R}^n$ will be used throughout the proof.

Definition 1: The β norm on a function v: $[0, T] \to \mathbb{R}^n$ is defined as $\max_{\tau \in [0,T]} e^{-B\tau} \| v(t) \|$ where B is some positive number. The β norm is denoted by $\| v \|_B$.

Note that the β norm is really a continuous family of norms, one for each positive number B. On the space of continuous differentiable functions that map $[0, T] \to \mathbb{R}^n$, all β norms are equivalent

Lemma 1: If f: $\mathbb{R}^n \to \mathbb{R}^n$ has a derivative J_f: $\mathbb{R}^n \to \mathbb{R}^{n \times n}$, which is Lipschitz continuous with some Lipschitz constant l_f, then

$$\|f(y) - f(x) - J_f(x)(x - y)\| < l_f \|x - y\|^2. \quad (9)$$

Lemma 1 can be thought of as a remainder theorem for the Taylor series.

Lemma 2: Given $F(v)$ as in (2) and $J_F(v)$ as in (3), it follows that

$$\left\| \int_0^t F(x)(\tau) - F(y)(\tau) - J_F(y)(x - y)(\tau)\, d\tau \right\|_B < \left(l_{\dot q} + \frac{T l_{\dot g}}{B} \right) \|x - y\|_B^2 \quad (10)$$

for any x, y in the space of continuous differentiable functions that map $[0, T] \to \mathbb{R}^n$, where $l_{\dot q}$ is the Lipschitz constant for $\partial q / \partial v$ and $l_{\dot g}$ is the Lipschitz constant for $\partial g / \partial v$.

Lemma 2 can be derived by exploiting properties of the β norm [8], and applying Lemma 1.

Lemma 3: Let v^k, v^{k+1} be two iterates generated by (4), and $F(v)$ be as given in (2), then

$$\|v^{k+1} - v^k\|_B < l_{\dot q - 1} \left(1 + \frac{T l_{\dot g}}{B} \right) \left\| \int_0^t F(v^k)(\tau)\, d\tau \right\|_B \quad (11)$$

where $l_{\dot q - 1}$ is the bound on $(\partial q / \partial v)^{-1}$.

Lemma 3 can also be shown using β norm properties.

To prove the theorem, let v^k, and v^{k+1} be the k and $k + 1$st WN iterates. Then

$$\left\| \int_0^t F(v^{k+1})(\tau) - F(v^k)(\tau) - J_F(v^k)(v^{k+1} - v^k)(\tau)\, d\tau \right\|_B < \left(l_{\dot q} + \frac{T l_{\dot g}}{B} \right) \|v^{k+1} - v^k\|_B^2 \quad (12)$$

by Lemma 2. By definition of the WN algorithm

$$F(v^k)(\tau) - J_F(v^k)(v^{k+1} - v^k)(\tau) = 0 \quad (13)$$

and therefore, (12) can be reduced to

$$\left\| \int_0^t F(v^{k+1})(\tau)\, d\tau \right\| < \left(l_{\dot q} + \frac{T l_{\dot g}}{B} \right) \|v^{k+1} - v^k\|_B^2. \quad (14)$$

By Lemma 3

$$\frac{1}{l_{\dot q - 1} \left(1 + \frac{T l_{\dot g}}{B} \right)} \|v^{k+2} - v^{k+1}\|_B < \left\| \int_0^t F(v^k)(\tau)\, d\tau \right\|_B. \quad (15)$$

Substituting into (14) leads to

$$\|v^{k+2} - v^{k+1}\|_B < \left(l_{\dot q - 1} \left(1 + \frac{T l_{\dot g}}{B} \right) \right) \left(l_{\dot q} + \frac{T l_{\dot g}}{B} \right) \|v^{k+1} - v^k\|_B^2. \quad (16)$$

The theorem follows by noticing that the β norm of $\| v^{k+1} - v^k \|_B$ can be made as small as desired by increasing B, because $v^{k+1}(0) - v^k(0) = 0$. The result is that by picking a B large enough, (16) can be written as

$$\|v^{k+2} - v^{k+1}\|_B < \gamma \|v^{k+1} - v^k\|_B \quad (17)$$

where $\gamma < 1$ and is a function of B, $l_{\dot q - 1}$, $l_{\dot q}$, $l_{\dot g}$, and T. It, therefore, follows that v^k is a cauchy sequence, and therefore, converges. ■

Acknowledgment

The authors would like to thank A. Sangiovanni-Vincentelli and A. R. Newton for their guidance and assistance in this work, as well as Ken Kundert, Prof. O. Nevanlinna, F. Odeh, Albert Ruehli, and Don Webber for the many valuable discussions about waveform methods, and Gih-Guang Hung for generating the results.

References

[1] L. W. Nagel, "SPICE2: A computer program to simulate semiconductor circuits," Electron. Res. Lab. Rep. No. ERL-M520, Univ. of California, Berkeley, May 1975.

[2] W. T. Weeks, A. J. Jimenez, G. W. Mahoney, D. Mehta, H. Qassemzadeh, and T. R. Scott, "Algorithms for ASTAP—A network analysis program," *IEEE Trans. Circuit Theory*, vol. CT-20, pp. 628–634, Nov. 1973.

[3] C. William Gear, "Automatic multirate methods for ordinary differential equations," in *Information Processing 80*. Amsterdam, The Netherlands: North-Holland.

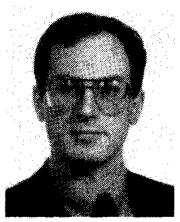
[4] K. Sakallah and S. W. Director, "An activity-directed circuit simulation algorithm," in *Proc. IEEE Int. Conf. Circuits and Computers*, Oct. 1980.
[5] A. R. Newton and A. L. Sangiovanni-Vincentelli, "Relaxation-based electrical simulation," *IEEE Trans. Electron Devices*, vol. ED-30, pp. 1184-1207, Sept. 1983; also *SIAM J. Sci. Stat. Computing*, vol. 4, no. 3, Sept. 1983; also *IEEE Trans. Computer-Aided Design*, vol. CAD-3, pp. 308-330, Oct. 1984.
[6] B. R. Chawla, H. K. Gummel, and P. Kozak, "MOTIS-An MOS Timing Simulator," *IEEE Trans. Circuits Syst.*, vol. CAS-22, pp. 901-909, Dec. 1975.
[7] S. P. Fan, M. Y. Hsueh, A. R. Newton, and D. O. Pederson, "MOTIS-C a new circuit simulator for MOS LSI circuits," in *Proc. IEEE Int. Symp. Circuit and Systems*, Apr. 1977.
[8] E. Lelarasmee, A. E. Ruehli, and A. L. Sangiovanni-Vincentelli, "The waveform relaxation method for time domain analysis of large scale integrated circuits," *IEEE Trans. Computer-Aided Design*, vol. 1, pp. 131-145, July 1982.
[9] 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," in *Proc. Int. Conf. Computer Design*, Rye, New York, Oct. 1984.
[10] R. Saleh and A. R. Newton, "An event-driven relaxation-based multirate integration scheme for circuit simulation," in *Proc. Int. Symp. Circuits and Systems*, Philadelphia, Pennsylvania, May 1987, pp. 600-603.
[11] J. M. Ortega and W. C. Rheinbolt, *Iterative Solution of Nonlinear Equations in Several Variables*. New York: Academic Press, 1970.
[12] M. Guarini and O. A. Palusinski, "Integration of partitioned dynamical systems using waveform relaxation and modified functional linearization," in *Summer Computer Simulation Proc.*, Vancouver, Canada, July 1983.
[13] W. M. G. Von Bokhoven, "An activity controlled modified waveform relaxation method," in *Proc Int. Symp. Circuits and Systems* Newport Beach, CA, May 1983.
[14] J. White, R. Saleh, A Sangiovanni-Vincentelli, and A. R. Newton, "Accelerating relaxation algorithms for circuit simulation using waveform-Newton, step-size refinement and parallel techniques," in *Proc. Int. Conf. on CAD*, Santa Clara, CA, Nov. 1985, pp. 5-7.
[15] O. Nevanlinna, Private Notes, 1987.
[16] Kantorovich, L. V. and G. P. Akilov, *Functional Analysis in Normed Spaces*. Oxford, U.K.: Pergammon, 1964.
[17] R. A. Saleh, J. E. Kleckner, and A. R. Newton, "Iterated timing analysis and SPLICE1," in *Proc. Int. Conf. on Computer-Aided Design*, Santa Clara, CA, Sept. 1983.
[18] J. White and A. S. Vincentelli, *Relaxation Techniques for the Simulation of VLSI Circuits*. Norwell, MA: Kluwer Academic, 1986.
[19] C. A. Desoer and E. S. Kuh, *Basic Circuit Theory*. New York: McGraw Hill, 1969.
[20] R. Saleh, D. Webber, E. Xia, and A. Sangiovanni-Vincentelli, "Parallel waveform Newton algorithms for circuit simulation," in *Proc. Int. Conf. Comp. Design*, Port Chester, New York, Oct. 1987, pp. 660-668.

*

Resve A. Saleh (M'79) obtained the B.Eng. degree in electrical engineering from Carleton University, Ottawa, Canada, in 1979, and the M.S. and Ph.D. degrees from University of California, Berkeley in 1983 and 1986, respectively.

He has worked in industry for Mitel Corporation, Kanata, Ont., Canada, Tektronix, Beaverton, OR, Toshiba Corporation, Kawasaki, Japan, and Shiva Multisystems, Menlo Park, CA. He joined the University of Illinois in 1986 where he is currently an Assistant Professor directing research in mixed-mode simulation and parallel processing. His research interests also include analog CAD and synthesis. Dr. Saleh has served on the technical committees of the Custom Integrated Circuits Conference and the Design Automation Conference since 1987, and was a member of the organizing committee of the MidWest Symposium on Circuits and Systems in 1989.

*



Jacob K. White (S'80-M'83) received the B.S. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, and the M.Sc. and Ph.D. degrees in electrical engineering and computer science from the University of California, Berkeley.

He worked at the IBM T. J. Watson Research Center from 1985 to 1987. In 1987, he joined the Massachusetts Institute of Technology where he is currently an Assistant Professor. His current interests are in developing serial and parallel numerical methods for problems in circuit and device analysis.

He received a Faculty Career Development Award from Analog Devices from 1987 to 1989, and obtained a Presidential Young Investigator Award in 1988.