Accelerated Waveform Methods for Parallel Transient Simulation of Semiconductor Devices

Mark W. Reichelt* Andrew Lumsdaine† Jacob K. White‡

Abstract

In this paper we compare accelerated waveform relaxation algorithms to pointwise methods for the transient simulation of semiconductor devices on parallel machines. Experimental results are presented for simulations on small clusters of workstations and on an Intel iPSC/860. The results show that accelerated waveform methods are competitive with standard pointwise methods on serial machines, but are significantly faster on loosely-coupled MIMD machines.

1 Introduction

The computational expense and growing importance of performing semiconductor device transient simulation, along with the increasing availability of parallel computers, suggest that parallel algorithms be developed and used for these simulation problems. Although SIMD type parallel machines have been shown to be effective for device transient simulation [1, 2], as MIMD machines become increasingly more popular and cost-effective, it is important that efficient algorithms be developed for them as well. To obtain highest performance on a MIMD parallel computer, it is critical that a numerical method avoid frequent parallel synchronization [3]. The waveform relaxation (WR) approach to solving time-dependent problems is such a method, because in parallel WR, iterates are communicated between processors only after having been computed over a time interval [4, 5, 6]. Parallel pointwise methods, on the other hand, must communicate iterates after each timestep computation.

As with any iterative scheme, efficiency of WR depends on rapid convergence, and there have been several investigations into accelerating WR, including multigrid [7], Krylov-subspace [8, 9], and convolution SOR techniques [10, 11]. In this paper we extend the results in [8, 9, 10, 11] and provide experimental results using waveform methods on two different parallel machines for performing transient simulation on a variety of MOS devices.

*The MathWorks, Inc., 24 Prime Park Way, Natick, MA 01760 (mwr@mathworks.com).

†Dept. of Comp. Sci. and Eng., University of Notre Dame, Notre Dame, IN 46556 (Andrew.Lumsdaine@nd.edu).

‡Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, MA 02139 (white@rle-vlsi.mit.edu).

2 Device Transient Simulation

Charge transport within a semiconductor device is assumed to be governed by the Poisson equation, and the electron and hole continuity equations [12, 13]. Given a two-dimensional rectangular discretization mesh, the device equation system is typically discretized with a finite-difference formula applied to the Poisson equation, and an exponentially-fit finite-difference formula applied to the continuity equations (the Scharfetter-Gummel method [13]). On an N-node mesh, this spatial discretization yields a sparse differential-algebraic initial-value problem (IVP) consisting of $3N$ equations in $3N$ unknowns, denoted by

$$\begin{aligned} \boldsymbol{F}_1(\boldsymbol{u}(t),\boldsymbol{n}(t),\boldsymbol{p}(t)) &= 0 \\ \tfrac{d}{dt}\boldsymbol{n}(t) + \boldsymbol{F}_2(\boldsymbol{u}(t),\boldsymbol{n}(t),\boldsymbol{p}(t)) &= 0 \\ \tfrac{d}{dt}\boldsymbol{p}(t) + \boldsymbol{F}_3(\boldsymbol{u}(t),\boldsymbol{n}(t),\boldsymbol{p}(t)) &= 0 \\ \boldsymbol{F}_i(\boldsymbol{u}(0),\boldsymbol{n}(0),\boldsymbol{p}(0)) &= 0, \quad i = 1,2,3 \end{aligned} \tag{1}$$

where $t \in [0,T]$, and $\boldsymbol{u}(t), \boldsymbol{n}(t), \boldsymbol{p}(t) \in \mathbb{R}^N$ are vectors of normalized potential, electron concentration, and hole concentration. Here, $\boldsymbol{F}_1, \boldsymbol{F}_2, \boldsymbol{F}_3 : \mathbb{R}^{3N} \rightarrow \mathbb{R}^N$ are specified component-wise as

$$F_{1_i}(u_i,n_i,p_i,u_j) = \frac{kT}{q}\sum_j \frac{d_{ij}\epsilon_{ij}}{L_{ij}}(u_i-u_j) - qA_i\,(p_i-n_i+N_{D_i}-N_{A_i})$$

$$F_{2_i}(u_i,n_i,u_j,n_j) = \frac{kT}{qA_i}\sum_j \frac{d_{ij}\mu_{n_{ij}}}{L_{ij}}\Big[n_i B(u_i-u_j) - n_j B(u_j-u_i)\Big] + R_i$$

$$F_{3_i}(u_i,p_i,u_j,p_j) = \frac{kT}{qA_i}\sum_j \frac{d_{ij}\mu_{p_{ij}}}{L_{ij}}\Big[p_i B(u_j-u_i) - p_j B(u_i-u_j)\Big] + R_i$$

The sums above are taken over the silicon nodes j adjacent to node i. For each node j adjacent to node i, L_{ij} is the distance from node i to node j, d_{ij} is the length of the side of the Voronoi box that encloses node i and

bisects the edge between nodes i and j, and ϵ_{ij}, $\mu_{n_{ij}}$ and $\mu_{p_{ij}}$ are the dielectric permittivity, electron and hole mobility, respectively, on the edge between nodes i and j. The Bernoulli function, $B(x) = x/(e^x - 1)$, is used to exponentially fit potential variation to electron and hole concentration variations, and effectively upwinds the current equations.

3 Accelerated Waveform Methods

The standard approach used to solve the differential-algebraic equation (DAE) system (1) is to discretize the system in time with a low-order implicit integration method. The resulting sequence of nonlinear algebraic systems is typically solved with some variant of Newton's method and/or relaxation [12, 14]. This approach can be disadvantageous for a parallel implementation, especially for MIMD parallel computers having a high communication latency, since the processors will have to synchronize repeatedly (at every Newton iteration) for each timestep.

A more effective approach to solving (1) with a parallel computer is to use waveform relaxation (WR) to decompose the DAE system into subsystems before time discretization [6]. The WR algorithm has several computational advantages. Since it is an iterative method, WR avoids factoring large sparse matrices. WR can exploit multi-rate behavior, using different timesteps to resolve different solution components. Finally, WR is well suited to parallel computation, because of a low communication/computation ratio. However, when applied to solving the device equation system (1), the WR algorithm converges slowly unless acceleration techniques, such as convolution SOR [10, 11] or waveform GMRES [8, 9], are used.

3.1 Convolution SOR

Convolution SOR (CSOR) is a generalized waveform extension of the well-known successive overrelaxation (SOR) method used to accelerate the convergence of relaxation methods for solving linear systems of equations [15]. To abbreviate the description of the CSOR algorithm, we will consider the problem of numerically solving the linear initial-value problem,

$$\dot{\boldsymbol{x}}(t) + \boldsymbol{A}\boldsymbol{x}(t) = \boldsymbol{b}(t) \quad \text{with} \quad \boldsymbol{x}(0) = \boldsymbol{x}_0, \qquad (2)$$

where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{b}(t) \in \mathbb{R}^n$ is given for all $t \in [0, T]$, and $\boldsymbol{x}(t) \in \mathbb{R}^n$ is to be computed.

A waveform relaxation algorithm using CSOR for solving (2) is shown in Algorithm 1. In iteration $k+1$, each waveform $\hat{x}_i^{k+1}$ is computed as in ordinary Gauss-Seidel WR, and then is moved slightly farther in the iteration direction by convolution with a CSOR parameter, function $\omega(t)$. The convolution allows the CSOR method to correctly account for the frequency-dependence of the spectrum of the Gauss-Jacobi WR operator by, in effect, using a different SOR parameter for each frequency.

Algorithm 1 (Convolution SOR)

1. *Initialize:* Pick $\boldsymbol{x}^0(t)$ with $\boldsymbol{x}^0(0) = \boldsymbol{x}_0$.
2. *Iterate:* For $k = 0, 1, \ldots$ until converged
   - *Solve* for $\hat{x}_i^{k+1}(t)$ with $\hat{x}_i^{k+1}(0) = x_{0_i}$,

$$\left(\tfrac{d}{dt} + a_{ii}\right)\hat{x}_i^{k+1}(t) = b_i(t) - \sum_{j=1}^{i-1} a_{ij}x_i^{k+1}(t) - \sum_{j=i+1}^{n} a_{ij}x_i^k(t)$$

   - *Overrelax* to compute $x_i^{k+1}(t)$,

$$x_i^{k+1}(t) = x_i^k(t) + \int_0^t \omega(\tau)\cdot\left[\hat{x}_i^{k+1}(t-\tau) - x_i^k(t-\tau)\right]d\tau$$

3.2 Waveform GMRES

The basic iteration used by WR for solving (2) (with the splitting $\boldsymbol{A} = \boldsymbol{M} - \boldsymbol{N}$) is

$$\dot{\boldsymbol{x}}^{k+1}(t) + \boldsymbol{M}\boldsymbol{x}^{k+1}(t) = \boldsymbol{N}\boldsymbol{x}^k(t) + \boldsymbol{b}(t)$$

The solution $\boldsymbol{x}$ to (2) is thus a fixed point of the WR algorithm, satisfying the integral operator equation

$$(\boldsymbol{I} - \mathcal{K})\boldsymbol{x} = \boldsymbol{\psi}. \qquad (3)$$

Here, we define (3) on the space $\mathbb{H} = \mathbb{L}_2([0, T], \mathbb{R}^N)$, $\boldsymbol{I} : \mathbb{H} \to \mathbb{H}$ is the identity operator, $\mathcal{K} : \mathbb{H} \to \mathbb{H}$ is defined by

$$(\mathcal{K}\boldsymbol{x})(t) = \int_0^t e^{(s-t)\boldsymbol{M}}\boldsymbol{N}\boldsymbol{x}(s)ds,$$

and $\boldsymbol{\psi} \in \mathbb{H}$ is given by

$$\boldsymbol{\psi}(t) = e^{-t\boldsymbol{M}}\boldsymbol{x}(0) + \int_0^t e^{(s-t)\boldsymbol{M}}\boldsymbol{b}(s)ds.$$

Krylov-subspace methods can be used to accelerate the convergence of WR, but as $\mathcal{K}$ is not self-adjoint, a variant suitable for non-self-adjoint operators must be used [8, 9]. One such method, shown in Algorithm 2, is waveform GMRES (WGMRES), an extension of the generalized minimum residual algorithm (GMRES) [16] to the space $\mathbb{H}$.

Algorithm 2 (Waveform GMRES)

1. *Start:* Set $\boldsymbol{r}^0 = \boldsymbol{\psi} - (\boldsymbol{I} - \boldsymbol{\mathcal{K}})\boldsymbol{x}^0$, $\boldsymbol{v}^1 = \boldsymbol{r}^0/\|\boldsymbol{r}^0\|$
2. *Iterate:* For $k = 1, 2, \ldots$, until satisfied do:
   - $h_{j,k} = \langle (\boldsymbol{I} - \boldsymbol{\mathcal{K}})\boldsymbol{v}^k, \boldsymbol{v}^j \rangle$, $j = 1, 2, \ldots, k$
   - $\hat{\boldsymbol{v}}^{k+1} = (\boldsymbol{I} - \boldsymbol{\mathcal{K}})\boldsymbol{v}^k - \sum_{j=1}^{k} h_{j,k}\boldsymbol{v}^j$
   - $h_{k+1,k} = \|\hat{\boldsymbol{v}}^{k+1}\|$
   - $\boldsymbol{v}^{k+1} = \hat{\boldsymbol{v}}^{k+1}/h_{k+1,k}$
3. *Form approximate solution:* $\boldsymbol{x}^k = \boldsymbol{x}^0 + \boldsymbol{V}^k\boldsymbol{y}^k$, where $\boldsymbol{y}^k$ minimizes $\|\beta\boldsymbol{e}_1 - \bar{\boldsymbol{H}}^k\boldsymbol{y}^k\|$,

To apply WGMRES to the nonlinear device system (1), the system is linearized with waveform Newton (WN) [17] and WGMRES is used to solve the resulting sequence of time-varying linear IVPs [9].

4 Parallel Implementation

Various parallel solution methods have been implemented in the WR-based device transient simulation program pWORDS, which supports computation on the Intel iPSC/860 as well as workstation clusters running the Parallel Virtual Machine (PVM) software [18]. The pWORDS program uses a manager/worker scheme in which a host program assigns tasks to compute nodes and gathers their results when they are finished.

4.1 Parallel Waveform Methods

The waveform solution methods in the pWORDS program use a vertical mesh-line blocking scheme in which the variables for the nodes in each vertical mesh line are computed together. To maximize parallelization for CSOR, the blocks are processed in red/black order [15] — WGMRES also uses this ordering to determine the splitting $\boldsymbol{A} = \boldsymbol{M} - \boldsymbol{N}$.

To begin a parallel WR computation, the host program reads an input file that specifies the device geometry and discretization mesh as well as the voltage boundary conditions. The host program then partitions the device mesh into non-overlapping blocks of adjacent mesh lines and assigns one block to each compute node. The WR iteration performed by each compute node overlaps communication and computation in order to minimize the effect of communication time and is shown in Algorithm 3.

Algorithm 3 (Parallel Red/Black WR)

Choose waveforms that satisfy initial conditions.

Iterate:

1. *Send* black line solutions needed by other nodes.
2. *Solve* (1) for red lines that do not require black line solutions from other nodes.
3. *Receive* black line solutions (BLOCKING).
4. *Solve* (1) for remaining red lines.
5. *Send* red line solutions needed by other nodes.
6. *Solve* (1) for black lines that do not require red line solutions from other nodes.
7. *Receive* red line solutions (BLOCKING).
8. *Solve* (1) for remaining black lines.

Each iteration of Algorithm 3 contains only two blocking communications — steps 3 and 7. Little synchronization between the compute nodes is therefore required, and communication that is required consists of large packets of information, i.e., entire waveforms.

4.2 Pointwise Newton-GMRES

In our experience, the most efficient serial algorithm for device transient simulation was the pointwise Newton-GMRES algorithm. In this algorithm, block-Jacobi preconditioned GMRES [16] is used to solve the linear systems arising at each Newton iteration of each timestep of an implicit integration formula applied to (1). The pointwise Newton-GMRES method in pWORDS uses the same vertical-line blocks as the waveform methods, but communication is required for each GMRES iteration. Although many of these communication operations are overlapped with local computation, the communication latency may be so large that it cannot be hidden by the (relatively) small amount of computation done at a single timestep. Furthermore, a significant amount of synchronization results from the inner-product computations within each GMRES iteration.

5 Experimental Results

The three different MOS devices of Figure 1 were used to construct six simulation examples, each device being subjected to either a drain voltage pulse with

device	description	mesh	unknowns
ldd	lightly-doped drain	15×20	656
soi	silicon-on-insulator	18×24	856
kar	abrupt junction	19×31	1379

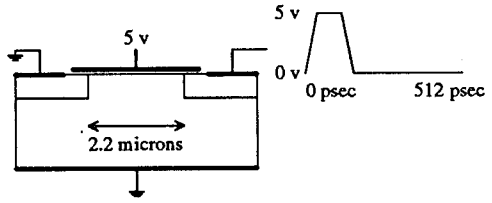

Figure 1: Description of devices and illustration of the drain-driven **karD** example.

the gate held high (the **D** examples), or a gate voltage pulse with the drain held high (the **G** examples). The (**karG64** and **soiG64**) examples were constructed by refining the meshes of **karG** and **soiG** to contain 64 vertical lines. Zero-volt Dirichlet boundary conditions were imposed by ohmic contacts at the source and along the bottom of the substrate and Neumann reflecting boundary conditions were imposed along the left and right sides. The drain-driven **karD** example is shown in Figure 1.

The experiments compared parallelized pointwise Newton/GMRES, WRN [17], WN/WGMRES, and WRN with CSOR acceleration. The backward Euler method with 256 fixed timesteps was used for all experiments, on a simulation interval of 51.2 or 512 picoseconds. Although the use of global uniform timesteps precludes multirate integration (one of the primary computational advantages of WR on a serial machine), it also simplifies the problem of load-balancing. The convergence criterion for all experiments was that the maximum relative error of any terminal current over the simulation interval be less than 10^{-4}. The initial guess for WRN and for the accelerated waveform methods was produced by performing 16 or 32 WR iterations beginning with flat waveforms extended from the initial conditions.

Table 1 shows a comparison of the execution times (in wall clock seconds) required to complete a transient simulation of the **karD** example using WRN, WN/WGMRES, and CSOR on an Ethernet-connected PVM workstation cluster consisting of eight IBM RS/6000 workstations as compute nodes and one Sun SparcStation 2 as host. Despite differences in compute node processing power, the mesh was divided as evenly as possible among the nodes — no load balancing was attempted. Note that the execution time for pointwise Newton-GMRES increased by a factor of 5.67 when parallelized on two processors.

Method	# Procs	Time
Pointwise (Direct)	1	2462.48
Pointwise (GMRES)	1	1221.98
Pointwise (GMRES)	2	6931.86
WRN	1	8230.23
WRN	2	4469.91
WRN	4	2712.58
WRN	8	1571.92
WN/WGMRES	1	*
WN/WGMRES	2	*
WN/WGMRES	4	925.60
WN/WGMRES	8	504.50
WRN with CSOR	1	1665.58
WRN with CSOR	2	884.64
WRN with CSOR	4	541.76
WRN with CSOR	8	316.08

Table 1: Execution times (in wall clock seconds) for transient simulation of the **karD** example on a PVM workstation cluster. A * indicates that the experiment could not run because of memory limitations.

Table 2 shows a comparison of the execution times (in wall clock seconds) required to complete a transient simulation of the **karD** example using WRN and CSOR on the Intel iPSC/860 hypercube. Here, pointwise Newton-GMRES does exhibit some speed-up when parallelized, but the speed-up flattens out after four processors.

Table 3 shows a comparison of the best serial and the best parallel execution times (in wall clock seconds) for the eight examples on the iPSC/860. The waveform methods display a remarkable scalability, with a roughly linear speedup up to 32 processors (the **soiG64** and **karG64** examples). Of the examples not

Method	# Procs	Time
Pointwise (GMRES)	1	1333.07
Pointwise (GMRES)	2	837.41
Pointwise (GMRES)	4	643.92
Pointwise (GMRES)	8	674.72
Pointwise (GMRES)	16	757.19
WRN	4	4378.82
WRN	8	2256.21
WRN	16	1224.32
WRN with CSOR	4	895.50
WRN with CSOR	8	460.27
WRN with CSOR	16	248.93

Table 2: Execution times (in wall clock seconds) for transient simulation of the **karD** example on the iPSC/860.

Example	Newton-GMRES	CSOR (Blocks/Proc) 8	4	2
lddD	1184.68	NA	382.77	201.23
lddG	989.74	NA	235.67	128.59
soiD	366.11	405.93	225.41	120.15
soiG	401.42	417.40	225.74	117.31
karD	1262.49	895.50	460.27	248.93
karG	1492.17	590.88	308.29	165.61
soiG64	*	987.17	507.20	260.13
karG64	*	1386.32	713.97	373.53

Table 3: Execution times (in wall clock seconds) for single processor pointwise Newton-GMRES and parallel CSOR simulation of eight examples on the iPSC/860. A * indicates that the experiment could not run because of memory limitations.

limited by the memory of a single iPSC/860 node, parallel CSOR was nearly an order of magnitude faster than serial Newton-GMRES in the best case.

6 Conclusion

The comparison of accelerated waveform relaxation algorithms to pointwise methods showed that accelerated waveform methods are competitive with standard pointwise methods on serial machines, and that accelerated waveform methods are significantly faster on commonly available loosely-coupled MIMD machines. In particular, parallel accelerated waveform methods achieved a nearly linear speed-up on the 32 processor Intel iPSC/860 hypercube, whereas parallel versions of standard pointwise methods exhibited only limited parallel speed-up. These results suggest that as MIMD machines and cluster-based computing become more prevalent, accelerated waveform methods will gain in importance for solving time-dependent problems.

Current work is focused on refining the theory for CSOR and WGMRES, developing a multirate implementation of pWORDS, and applying accelerated waveform methods to new application areas.

Acknowledgments

This work was supported by IBM, ARPA contract N00014-91-J-1698, and NSF grants MIP-8858764 A02 and CCR92-09815. The authors thank Professors L. N. Trefethen, D. Rose, P. Lanzcron, A. Toomre and O. Nevanlinna for many valuable suggestions. Prof. A. Patera provided iPSC/860 time for initial development and experimentation. Subsequent iPSC/860 time was provided by the San Diego Supercomputer Center. J. Squyres made several improvements to pWORDS and conducted many numerical experiments.

References

[1] C. L. Gardner, P. J. Lanzkron, and D. J. Rose, "A parallel block iterative method for the hydrodynamic device model," *IEEE Trans. CAD*, vol. 10, pp. 1187–1192, September 1991.

[2] D. Webber, E. Tomacruz, R. Guerrieri, T. Toyabe, and A. Sangiovanni-Vincentelli, "A massively parallel algorithm for three-dimensional device simulation," *IEEE Trans. CAD*, vol. 10, pp. 1201–1209, September 1991.

[3] A. Agarwal, "Limits on interconnection network performance," *IEEE Trans. Parallel Distrib. Sys.*, pp. 398–412, October 1991.

[4] U. Miekkala and O. Nevanlinna, "Convergence of dynamic iteration methods for initial value problems," *SIAM J. Sci. Statist. Comput.*, vol. 8, pp. 459–482, July 1987.

[5] J. K. White and A. Sangiovanni-Vincentelli, *Relaxation Techniques for the Simulation of VLSI Circuits*. Boston: Kluwer Academic Publishers, 1987.

[6] 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. CAD*, vol. 1, pp. 131–145, July 1982.

[7] C. Lubich and A. Osterman, "Multi-grid dynamic iteration for parabolic equations," *BIT*, vol. 27, pp. 216–234, 1987.

[8] A. Lumsdaine, M. Reichelt, and J. White, "Conjugate direction waveform methods for transient two-dimensional simulation of MOS devices," in *Proc. International Conference on Computer-Aided Design*, (Santa Clara, California), pp. 116–119, November 1991.

[9] A. Lumsdaine, *Theoretical and Practical Aspects of Parallel Numerical Algorithms for Initial Value Problems, with Applications*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1992.

[10] M. Reichelt, J. White, and J. Allen, "Waveform frequency-dependent overrelaxation for transient two-dimensional simulation of MOS devices," in *Proc. Workshop on Numerical Modeling of Processes and Devices for Integrated Circuits*, (Seattle, WA), pp. 161–166, May 1992.

[11] M. Reichelt, *Accelerated Waveform Relaxation Techniques for the Parallel Transient Simulation of Semiconductor Devices*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1993.

[12] R. E. Bank, W. C. Coughran, Jr., W. Fichtner, E. Grosse, D. Rose, and R. Smith, "Transient simulation of silicon devices and circuits," *IEEE Trans. CAD*, vol. 4, pp. 436–451, October 1985.

[13] S. Selberherr, *Analysis and Simulation of Semiconductor Devices*. New York: Springer-Verlag, 1984.

[14] K. Mayaram and D. Pederson, "CODECS: A mixed-level device and circuit simulator," in *Proc. International Conference on Computer-Aided Design*, (Santa Clara, California), pp. 112–115, November 1988.

[15] D. M. Young, *Iterative Solution of Large Linear Systems*. Orlando, FL: Academic Press, 1971.

[16] Y. Saad and M. Schultz, "GMRES: A generalized minimum residual algorithm for solving nonsymmetric linear systems," *SIAM J. Sci. Statist. Comput.*, vol. 7, pp. 856–869, July 1986.

[17] R. Saleh and J. White, "Accelerating relaxation algorithms for circuit simulation using waveform-newton and step-size refinement," *IEEE Trans. CAD*, vol. 9, pp. 951–958, September 1990.

[18] A. Beguilin *et al.*, "A users' guide to PVM parallel virtual machine," ORNL/TM 11826, Oak Ridge National Laboratories, Oak Ridge, TN, 1992.