

Waveform Iterative Techniques for Device Transient Simulation on Parallel Machines*

Andrew Lumsdaine[†] Mark W. Reichelt[‡]

Abstract

In this paper we describe our experiences with parallel implementations of several different waveform algorithms for performing transient simulation of semiconductor devices. Because of their inherent computation and communication structure, waveform methods are well suited to MIMD-type parallel machines having a high communication latency — such as a cluster of workstations. Experimental results using pWORDS, a parallel waveform-based device transient simulation program, in conjunction with PVM running on a cluster of eight workstations demonstrate that parallel waveform techniques are an efficient and faster alternative to standard simulation algorithms.

1 Introduction

The enormous computational expense and growing importance of device transient simulation, as well as the increasing availability of parallel computing environments, suggest that specialized, easily parallelized, algorithms be developed for transient simulation of MOS devices [5]. The easily parallelized waveform relaxation (WR) algorithm was shown to be a computationally efficient approach to device transient simulation [11], even though the WR algorithm typically requires hundreds of iterations to achieve an accurate solution. In [8], it was demonstrated that the convergence rate of waveform relaxation for device transient simulation could be greatly improved by the application of Krylov-subspace acceleration.

In this paper, we extend the work in [11] and [8] by describing parallel implementations of several different waveform algorithms for performing transient simulation of semiconductor devices. Because of their inherent computation and communication structure, waveform methods are well suited to MIMD-type parallel machines having a high communication latency — such as a cluster of workstations. Experimental results using pWORDS, a parallel waveform-based device transient simulation program, in conjunction with the Parallel Virtual Machine (PVM) environment [2] running on a cluster of eight workstations demonstrate that waveform techniques are an efficient alternative to standard simulation methods.

In the next section, we begin with a brief review of waveform methods. The device transient simulation problem is described in Section 3 and implementation details of the pWORDS program are provided in Section 4. Section 5 contains experimental results comparing traditional methods with parallel waveform methods for performing device

* This paper originally appeared in *Proc. Sixth SIAM Conference on Parallel Processing for Scientific Computing*, pp. 237–245, Norfolk, VA, 1993.

[†]Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556; Andrew.Lumsdaine@nd.edu. This work was supported in part by National Science Foundation grant CCR92-09815.

[‡]Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139; mwr@rle-vlsi.mit.edu. This work was supported by a grant from IBM, the Defense Advanced Research Projects Agency contract N00014-91-J-1698, and the National Science Foundation.

transient simulation. Finally, in Section 6 we present our conclusions and suggestions for future work.

2 Waveform Iterative Methods

Consider the problem of numerically solving the linear time-varying initial-value problem,

$$(1) \quad \begin{aligned} \left(\frac{d}{dt} + \mathbf{A}(t)\right)\mathbf{x}(t) &= \mathbf{f}(t) \\ \mathbf{x}(0) &= \mathbf{x}_0 \end{aligned}$$

where $\mathbf{A}(t) \in \mathbb{R}^{N \times N}$, $\mathbf{f}(t) \in \mathbb{R}^N$ is a given right-hand side, and $\mathbf{x}(t) \in \mathbb{R}^N$ is the unknown vector to be computed over the simulation interval $t \in [0, T]$. The traditional numerical approach for solving (1) is to begin by discretizing the system in time with an implicit integration rule (since large systems of equations are typically stiff) and then solving the resulting matrix problem at each time step. 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 for each timestep.

A more effective approach to solving (1) with a parallel computer is to decompose the problem at the ODE level. That is, the large system is decomposed into smaller subsystems, each of which is assigned to a single processor. The total system is solved iteratively by solving the subsystems independently, using fixed values from previous iterations for the variables from other subsystems. This dynamic iteration process is known as waveform relaxation (WR) [6] or as the Picard-Lindelöf iteration [9].

2.1 Linear Systems

In (1), let $\mathbf{A}(t) = \mathbf{M}(t) - \mathbf{N}(t)$ be a splitting of $\mathbf{A}(t)$. The waveform relaxation algorithm based on this splitting is expressed as

ALGORITHM 2.1. (WAVEFORM RELAXATION FOR LINEAR SYSTEMS)

1. *Initialize:* Pick \mathbf{x}^0

2. *Iterate:* For $k = 0, 1, \dots$

$$\begin{aligned} \text{Solve } \left(\frac{d}{dt} + \mathbf{M}\right)\mathbf{x}^{k+1} &= \mathbf{N}\mathbf{x}^k + \mathbf{f} \\ \mathbf{x}(0) &= \mathbf{x}_0 \end{aligned}$$

for \mathbf{x}^{k+1} on $[0, T]$.

The solution \mathbf{x} to (1) is thus a fixed point of the WR algorithm, satisfying the integral operator equation

$$(2) \quad (\mathbf{I} - \mathcal{K})\mathbf{x} = \boldsymbol{\psi}.$$

Here, (2) is defined on the space $\mathbb{H} = \mathbb{L}_2([0, T], \mathbb{R}^N)$, $\mathbf{I} : \mathbb{H} \rightarrow \mathbb{H}$ is the identity operator, $\mathcal{K} : \mathbb{H} \rightarrow \mathbb{H}$ is defined by

$$(\mathcal{K}\mathbf{x})(t) = \int_0^t \boldsymbol{\Phi}_M(t, s)\mathbf{N}(s)\mathbf{x}(s)ds$$

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

$$\boldsymbol{\psi}(t) = \boldsymbol{\Phi}_M(t, 0)\mathbf{x}(0) + \int_0^t \boldsymbol{\Phi}_M(t, s)\mathbf{f}(s)ds,$$

and $\boldsymbol{\Phi}_M$ is the state transition matrix [3] associated with $\mathbf{M}(t)$.

Waveform relaxation for solving (2) is expressed in operator equation form simply as

$$(3) \quad \mathbf{x}^{k+1} = \mathcal{K}\mathbf{x}^k + \boldsymbol{\psi}.$$

Since for any finite interval $[0, T]$, the operator \mathcal{K} has zero spectral radius, this process will produce iterates \mathbf{x}^k that converge to the solution \mathbf{x} of (2), or equivalently, to the solution of (1). A more detailed analysis of convergence for the linear time-invariant case can be derived by considering (2) on the interval $[0, \infty)$ in which case \mathcal{K} has non-zero spectral radius [9].

2.2 Krylov-Subspace Acceleration

As shown in [7], Krylov-subspace methods can be applied to (2) to accelerate the convergence of WR, but as \mathcal{K} is not self-adjoint, only methods suitable for non-self-adjoint operators are suitable. One such method is waveform GMRES (WGMRES), an extension of the generalized minimum residual algorithm (GMRES) [12] to the space \mathbb{H} .

ALGORITHM 2.2. (WAVEFORM GMRES)

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

The two fundamental operations in Algorithm 2.2 are the operator-function product, $(\mathbf{I} - \mathcal{K})\mathbf{p}$, and the inner product, $\langle \cdot, \cdot \rangle$. When solving (2) in the space \mathbb{H} , these operations are as follows:

Operator-Function Product: To calculate $\mathbf{w} \equiv (\mathbf{I} - \mathcal{K})\mathbf{p}$:

1. Solve $(\frac{d}{dt} + \mathbf{M})\mathbf{y} = \mathbf{N}\mathbf{p}$
 $\mathbf{y}(0) = \mathbf{p}_0 = 0$
 for $\mathbf{y}(t)$, $t \in [0, T]$ to obtain $\mathbf{y} = \mathcal{K}\mathbf{p}$.
2. Set $\mathbf{w} = \mathbf{p} - \mathbf{y}$

Inner Product: The inner product $\langle \mathbf{x}, \mathbf{y} \rangle$ is given by $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^N \int_0^T x_i(t)y_i(t)dt$.

Step 1 of the operator-function product is equivalent to one step of waveform relaxation, hence WGMRES has as its core the standard WR iteration (with the concomitant parallelizability). Moreover, the inner product can be computed by N separate integrations of the pointwise product $x_i(t)y_i(t)$, which can be performed in parallel, followed by a global sum of the results.

2.3 Nonlinear Systems

Many interesting applications are modeled as nonlinear systems, e.g.,

$$(4) \quad \begin{aligned} \frac{d}{dt}\mathbf{x}(t) + \mathbf{F}(\mathbf{x}(t), t) &= 0 \\ \mathbf{x}(0) &= \mathbf{x}_0. \end{aligned}$$

where $\mathbf{F} : \mathbb{R}^{N+1} \rightarrow \mathbb{R}^N$. Algorithm 2.1 is easily generalized to the nonlinear case. For example, the Jacobi WR algorithm can be used to solve (4) by solving the scalar equations

$$\begin{aligned} \frac{d}{dt}x_i^k(t) + f_i(x_1^k(t), \dots, x_{i-1}^k(t), x_i^{k+1}(t), x_{i+1}^k(t), \dots, x_n^k(t), t) &= 0 \\ x_i^k(0) &= x_{0_i} \end{aligned}$$

at each iteration k for each component i of \mathbf{x} . Significant efficiency can be gained by only approximately solving these scalar equations at each iteration. In particular, an accurate solution of each nonlinear equation will require multiple Newton iterations at each timestep. Instead, by only taking one Newton iteration using an initial guess obtained from the previous waveform iterate at each timestep, one obtains the waveform relaxation Newton (WRN) algorithm [13].

In order to use the waveform Krylov-subspace methods, Newton's method is applied to (4) — in a process sometimes referred to as the waveform Newton method (WN) [13] — to obtain the following iteration:

$$(5) \quad \begin{aligned} \left(\frac{d}{dt} + \mathbf{J}_F(\mathbf{x}^m)\right) \mathbf{x}^{m+1} &= \mathbf{J}_F(\mathbf{x}^m) \mathbf{x}^m - \mathbf{F}(\mathbf{x}^m) \\ \mathbf{x}^{m+1}(0) &= \mathbf{x}_0. \end{aligned}$$

Here, \mathbf{J}_F is the Jacobian of \mathbf{F} . We note that (5) is a linear time-varying system to be solved for \mathbf{x}^{m+1} , which can be accomplished with WGMRES. The resulting WN/WGMRES algorithm, a member of the class of hybrid Krylov methods [4], is given below.

ALGORITHM 2.3. (WAVEFORM NEWTON/WGMRES)

1. *Initialize:* Pick \mathbf{x}^0
2. *Iterate:* For $m = 0, 1, \dots$ until converged
 - Linearize (4) to form (5)
 - Solve (5) with WGMRES
 - Update \mathbf{x}^{m+1}

3 Device Transient Simulation

A semiconductor device is assumed to be governed by the Poisson equation, and the electron and hole continuity equations:

$$\begin{aligned} \frac{kT}{q} \nabla \cdot (\epsilon \nabla u) + q(p - n + N_D - N_A) &= 0 \\ \nabla \cdot \mathbf{J}_n - q \left(\frac{\partial n}{\partial t} + R \right) &= 0 \\ \nabla \cdot \mathbf{J}_p + q \left(\frac{\partial p}{\partial t} + R \right) &= 0 \end{aligned}$$

where u is the normalized electrostatic potential, n and p are the electron and hole concentrations, \mathbf{J}_n and \mathbf{J}_p are the electron and hole current densities, N_D and N_A are the donor and acceptor concentrations, R is the net generation and recombination rate, q is the magnitude of electronic charge, and ϵ is the spatially dependent dielectric permittivity [1, 15].

The current densities \mathbf{J}_n and \mathbf{J}_p are given by the drift-diffusion approximations:

$$\begin{aligned} \mathbf{J}_n &= -qD_n(n \nabla u - \nabla n) \\ \mathbf{J}_p &= -qD_p(p \nabla u + \nabla p) \end{aligned}$$

where D_n and D_p are the diffusion coefficients, which are assumed here to be related to the electron and hole mobilities by the Einstein relations, that is $D = \frac{kT}{q}\mu$. \mathbf{J}_n and \mathbf{J}_p are typically eliminated from the continuity equations using the drift-diffusion approximations, leaving a differential-algebraic system of three equations in three unknowns, u , n , and p .

Given a rectangular mesh that covers a two-dimensional slice of a MOSFET, a common approach to spatially discretizing the device equations is to use a finite-difference formula to discretize the Poisson equation, and an exponentially-fit finite-difference formula to discretize the continuity equations (the Scharfetter-Gummel method) [14]. On an N -node mesh, the spatial discretization yields a differential-algebraic system of $3N$ equations in $3N$ unknowns denoted by

$$\begin{aligned}\mathbf{f}_1(\mathbf{u}(t), \mathbf{n}(t), \mathbf{p}(t)) &= 0 \\ \mathbf{f}_2(\mathbf{u}(t), \mathbf{n}(t), \mathbf{p}(t)) &= \frac{d}{dt}\mathbf{n}(t) \\ \mathbf{f}_3(\mathbf{u}(t), \mathbf{n}(t), \mathbf{p}(t)) &= \frac{d}{dt}\mathbf{p}(t)\end{aligned}$$

where $t \in [0, T]$, and $\mathbf{u}(t), \mathbf{n}(t), \mathbf{p}(t) \in \mathbb{R}^N$ are vectors of normalized potential, electron concentration, and hole concentration, respectively.

4 Implementation

As reported in [11], the node-by-node Gauss-Jacobi WR algorithm will typically require many hundreds (or even thousands) of iterations to converge, severely limiting the efficiency of WR-based device simulation. Moreover, assigning each node to a separate processor in a parallel implementation would require on the order of a thousand processors or more (the number of nodes typically necessary for accurate device simulation). Since the WR algorithm is better suited to a coarse-grained MIMD type of architecture, such a fine-grained division of the problem is not necessary or desirable.

Instead of the node-by-node approach, pWORDS collects groups of mesh nodes into blocks and solves the nodes in each block simultaneously. In particular, the nodes in each vertical line of the discretization mesh are grouped together into blocks — this has been shown to be a particularly effective blocking strategy for MOSFET simulation [11].

The main WR routine in the pWORDS program uses a red/black block Gauss-Seidel scheme in which the system of equations governing the nodes in each vertical mesh line is solved using a backward-difference integration formula. The implicit algebraic systems generated by the backward-difference formula are solved with Newton's method and the linear equation systems generated by Newton's method are solved with sparse Gaussian elimination.

As previously mentioned, computing the operator-waveform product used by the waveform Krylov-subspace methods requires performing one step of traditional waveform relaxation. Therefore, the operator-waveform product can be accomplished with a function call to the WR routine already implemented within pWORDS — the vertical mesh line preconditioning scheme inherent to the WR routine will automatically be used by the Krylov-subspace method as well.

4.1 Parallelization

The pWORDS program uses a Master/Slave approach in which the supervisory Master program initiates the actual parallel simulation process by invoking S copies of a Slave program on S PVM client machines.

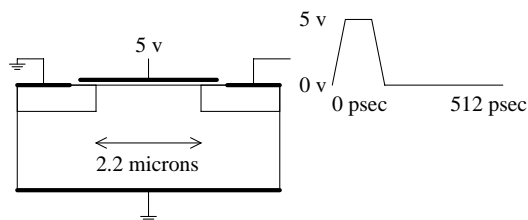


FIG. 1. *Illustration of the experimental two-dimensional N-channel MOS transistor model.*

The Master program reads in the device input file that specifies the geometry and the voltage boundary conditions imposed upon the device, as well as the 2D spatial discretization mesh. Given a rectangular mesh with C vertical lines, the Master splits the mesh into S non-overlapping blocks consisting of C/S adjacent vertical lines and sends each block to a Slave for solution.

In addition to the block of lines that each Slave solves, each Slave also contains storage for the two vertical lines on either side of the contiguous block. These “pseudo-lines” are used to store the solutions generated by the Slaves controlling those adjacent vertical lines.

To describe the algorithm running on each Slave machine, we alternately assign a “color,” either red or black, to the vertical mesh lines. The Slave WR iteration, which overlaps communication and computation in order to minimize the effect of communication time, is as follows:

1. *Send* the black line solutions needed for pseudo-lines on other Slaves and *compute* the solution for those red lines that do not depend on black pseudo-line solutions.
2. *Receive* black pseudo-line solutions and *compute* solutions for the remaining red lines.
3. *Send* the red line solutions needed for pseudo-lines on other Slaves and *compute* the solution for those black lines that do not depend on red pseudo-line solutions.
4. *Receive* red pseudo-line solutions and *compute* solutions for the remaining black lines.

5 Experimental Results

Numerical experiments were conducted using a two-dimensional n-channel MOS transistor model discretized with a 19×31 mesh. The experiments simulated the effect of a 5 volt pulse applied to the drain terminal with the gate terminal held at a constant 5 volts, as shown in Fig. 1.

The experimental parallel computing environment consisted of eight IBM RS/6000 workstations — five models 320, one model 320H, and two models 540 — and one Sun SparcStation 2. The Sun Sparcstation 2 was used as the Master for all experiments and the IBM RS/6000 machines were used as the Slaves. To make the parallel results as meaningful as possible, serial results were obtained on a single model 320, results with two and four processors were obtained on two and four model 320 Slaves, respectively, and results with eight processors were obtained with all eight machines. The mesh was divided as evenly as possible among the Slave processors — no load balancing was attempted.

Table 1 shows a comparison of the execution times (measured in elapsed wall clock seconds) required to complete a transient simulation of the test device using WRN and WN/WGMRES. For all experiments, first order BDF and 256 fixed timesteps were used over a simulation interval of 51.2×10^{-11} seconds. To establish a uniform measure for purposes

Method	# Procs	Time
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
Pointwise (Direct)	1	2462.48
Pointwise (GMRES)	1	1221.98
Pointwise (GMRES)	2	6931.86

TABLE 1

*Execution times (measured in elapsed wall clock seconds) required to complete a transient simulation of the test device using WRN, WN/WGMRES, and point at a time methods. A * indicates that the experiment was not able to be run because of memory restrictions.*

of comparison, the convergence criterion for all experiments was the requirement that the maximum error over the simulation interval in the value of any terminal current be less than one part in 1×10^{-4} . To provide an initial guess for WRN and for WN/WGMRES, 16 and 8 initial WR iterations were performed, respectively, after which WRN and WN/WGMRES required 499 and 75 iterations to converge, respectively.

In addition, Table 1 shows the execution times required to perform traditional point at a time simulation of the test device, using direct and vertical-line preconditioned GMRES (PGMRES) linear system solvers. Parallel runs with the PGMRES point at a time method were conducted, but as is shown in the table, execution time *increased* — a result of the large number of communication and synchronization steps required by PGMRES at each timestep and the high latency of PVM and standard ethernet communication. Parallelization of the point at a time simulation using direct methods was not attempted.

As can be seen from the table, the WN/WGMRES method has very good parallel performance (although because of its large memory requirements, it could not be accommodated by the smaller model 320 machines for runs on just one or two machines). Because it is necessarily more synchronous, WN/WGMRES might appear to be at a disadvantage (when compared to WR or WRN) in a parallel implementation, however its vastly superior convergence rate makes it the clear overall winner.

6 Conclusion

Using only eight processors, WN/WGMRES is more than two times faster than what is, to the authors' knowledge, the fastest serial point at a time method. It should be emphasized that no special hardware was used — only a cluster of workstations. WN/WGMRES is thus a very attractive alternative to traditional simulation methods for device transient simulation — in computer environments which are widely (if not universally) available.

The work reported here reflects the authors' first experiences with parallel implementations of waveform methods for performing transient simulation of semiconductor devices. As such, there are many potential areas to explore to improve the performance of these techniques even further. Current work focuses on the development of a parallel multirate

implementation, implementation for other parallel architectures, development of other techniques for accelerating WR, and the incorporation of timepoint pipelining techniques [16]. Primary among the theoretical questions to be addressed is why WN/WGMRES converges so well in practice, given the somewhat pessimistic results of [10].

Finally, the underlying concepts of the waveform methods are not specific to the device transient simulation problem. Other application areas which require the solution of large systems of differential or differential-algebraic equations can likely benefit from the use of waveform iterative techniques.

Acknowledgments

The authors would like to thank Professor Jonathan Allen, the members of the custom integrated circuits group at MIT, especially Jacob White and Ibrahim Elfadel, and Farouk Odeh of the IBM T. J. Watson research center for many valuable discussions. The first author would also like to thank Jack Dongarra for introducing him to PVM.

References

- [1] R. Bank, W. Coughran, Jr., W. Fichtner, E. Grosse, D. Rose, and R. Smith, *Transient simulation of silicon devices and circuits*, IEEE Trans. CAD, 4 (1985), pp. 436–451.
- [2] A. Beguilin et al., *A users' guide to PVM parallel virtual machine*, ORNL/TM 11826, Oak Ridge National Laboratories, Oak Ridge, TN, 1992.
- [3] R. W. Brockett, *Finite Dimensional Linear Systems*, Wiley, New York, 1970.
- [4] P. Brown and Y. Saad, *Hybrid Krylov methods for nonlinear systems of equations*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 450–481.
- [5] W. Engl, R. Laur, and H. Dirks, *MEDUSA – A simulator for modular circuits*, IEEE Trans. CAD, 1 (1982), pp. 85–93.
- [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 Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1 (1982), pp. 131–145.
- [7] 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.
- [8] A. Lumsdaine, M. Reichelt, and J. White, *Conjugate direction waveform methods for transient two-dimensional simulation of MOS devices*, in International Conference on Computer Aided-Design, Santa Clara, California, November 1991, pp. 116–119.
- [9] U. Miekkaala and O. Nevanlinna, *Convergence of dynamic iteration methods for initial value problems*, SIAM J. Sci. Stat. Comp., 8 (1987), pp. 459–467.
- [10] O. Nevanlinna, *Linear acceleration of Picard-Lindelöf iteration*, Numer. Math., 57 (1990), pp. 147–156.
- [11] M. Reichelt, J. White, and J. Allen, *Waveform relaxation for transient two-dimensional simulation of MOS devices*, in International Conference on Computer Aided-Design, Santa Clara, California, November 1989, pp. 412–415.
- [12] Y. Saad and M. Schultz, *GMRES: A generalized minimum residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
- [13] R. Saleh and J. White, *Accelerating relaxation algorithms for circuit simulation using waveform-Newton and step-size refinement*, IEEE Trans. CAD, 9 (1990), pp. 951–958.
- [14] D. Scharfetter and H. Gummel, *Large-signal analysis of a silicon read diode oscillator*, IEEE Transactions on Electron Devices, ED-16 (1969), pp. 64–77.
- [15] S. Selberherr, *Analysis and Simulation of Semiconductor Devices*, Springer-Verlag, New York, 1984.
- [16] J. K. White and A. Sangiovanni-Vincentelli, *Relaxation Techniques for the Simulation of VLSI Circuits*, Engineering and Computer Science Series, Kluwer Academic Publishers, Norwell, Massachusetts, 1986.