CHAPTER 55

Three-Dimensional MOS Device Simulation on a Connection Machine

O. Heinreichsberger*
S. Selberherr*
M. Stiftinger*

Abstract. In this paper we present our experience with the implementation of the three-dimensional semiconductor device simulator MINIMOS on a massively parallel architecture, the Connection Machine CM2. The emphasis of this work is placed on parallel iterative methods for solving the very large sparse linear systems of equations that arise at each step of the nonlinear solution procedure. Both symmetric and nonsymmetric linear systems are solved by conjugate gradient type iterative methods. The implementation of the parallel preconditioner is the most crucial step. Multicolor incomplete LU factorization preconditioners are compared with polynomial preconditioners. Several numerical examples from the nonsymmetric linear systems in MINIMOS are given with timing results obtained using the CM2. Comparisons with vector supercomputers are made.

1. Overview. Semiconductor device simulators compute the discrete solution of the semiconductor device partial differential equations: The Poisson equation and the carrier (electrons and holes) continuity equations. This system of nonlinear conservation laws is either solved by a damped Newton method or by a nonlinear Gauss-Seidel iteration, the so-called Gummel [5] algorithm. In the latter case, to which we restrict ourselves, each nonlinear (outer) iteration consists of the successive solution of the Poisson equation for the electrostatic potential ψ and of two carrier continuity equations for the electron and hole concentrations, n and p, respectively. Here we consider only three-dimensional nonuniform tensor product grids. For an overview see e.g. [1].
Solving the semiconductor equations means the repeated solution of large sparse linear systems. The coefficient matrices of the discrete continuity equations are nonsymmetric. Our contribution here is largely concerned with the iterative solution of those nonsymmetric linear systems on the CM2. The emphasis is placed on parallel preconditioning methods. See also [4].
Iterative methods applied to the discrete continuity equations have to cope with high condition numbers of the coefficient matrices and the enormous numerical range of the solution vector. Contrary to the Poisson equation the discrete continuity equations have to be evaluated much more accurately to guarantee the stability of the nonlinear iteration resulting in high iteration counts of the nonsymmetric solver. We use the bi-conjugate gradient squared (BiCGS) [12] method and a stabilized version of this algorithm, BiCGSTAB [14], for these nonsymmetric linear systems.

*Institute for Microelectronics, Technical University Vienna, Gusshausstrausse 27-29, A-1040 Vienna, Austria.

TABLE 1
BiCGSTAB Algorithm

Choose x_0 (e.g. $x_0 = 0$)
$r_0 = (b - \mathbf{A}x_0)$
Choose y_0 such that $\langle y_0, r_0 \rangle \neq 0$ (e.g. $y_0 = r_0$)
$p_0 = v_0 = 0$
$\rho_{-1} = 1$
$\omega_0 = 1$
$\alpha = 1$
FOR $n = 0$ STEP 1 UNTIL convergence DO
$\rho_n = \langle y_0, r_n \rangle$
$\beta = \frac{\rho_n}{\rho_{n-1}} \frac{\alpha}{\omega_n}$
$p_{n+1} = r_n + \beta (p_n + \omega_n v_n)$
$v_{n+1} = \mathbf{A}p_{n+1}$
$\alpha = \langle y_0, v_{n+1} \rangle$
$s = r_n - \alpha v_{n+1}$
$t = \mathbf{A}s$
$\omega_{n+1} = \frac{\langle t,s \rangle}{\langle t,t \rangle}$
$r_{n+1} = s - \omega_{n+1} t$
$x_{n+1} = x_n + \alpha p_{n+1} + \omega_{n+1} s$
END FOR

2. The Linear Solvers. The parallel solvers are of (bi-)conjugate gradient type. A relative error of 10^{-3} for the symmetric and 10^{-8} for the nonsymmetric systems (motivated by numerical experiments only) has been found to be sufficient.
For the Poisson equation the MIC-CG method[9] with a modification factor of $\alpha = 0.95$ is the optimal choice to the best to our knowledge. On the CM2 we use the CG in the version of Concus, Golub and O'Leary. As preconditioner the reduced system (RS) main diagonal is used. We shall denote this method by RS–CG.
In the case of locally constant carrier temperatures (see [6]) the coefficient matrices of the discrete continuity equations are diagonally similar to symmetric, positive definite matrices and thus have a positive real spectrum. In these linearized discrete systems the matrix coefficients vary rapidly resulting in a high condition number, thus yielding a large iteration counts of the linear iterative solver.
Among the number of iterative methods investigated [6] variants of the bi-conjugate gradient method such as the biconjugate gradient squared (BiCGS) method and the the BiCGSTAB method [14] (see Table 1) seem to be the optimal choice.
In particular the BiCGSTAB procedure improves one of the main problems of BiCGS, namely its erratic convergence behaviour which often yields significantly more iterations for convergence than necessary. At the same time the rapid convergence of BiCGS is maintained. BiCGSTAB needs two dotproducts more, and one vector update less.

3. Preconditioning Methods. Efficient and robust preconditioning is the most critical issue for the iterative solution of the discrete semiconductor equations.
Split incomplete LU preconditioning enables an efficient implementation of the incomplete Choleski and the incomplete LU preconditioner, a technique originally proposed in [3]. However, the ILU preconditioner of the naturally ordered unknowns is rather sequential in nature and thus unattractive on a SIMD architecture. Alternatives are polynomial preconditioners and ILU with multicolor orderings.
For multicolor ILU preconditioning the coefficient matrix is permuted according to some regular replication ('coloring') pattern of the unknowns [11]. A partitioning of the unknowns into sets of different colors resulting in a block structure of the coefficient matrix uncouples the unknowns

of the same color. The vector length decreases as the number of colors increases. At the same time the convergence speed – usually – increases. We have made experiments using the simulator NSPCG [10]. The results indicate that more than two colors are not favourable. We therefore implemented a 2-color (red-black) ILU preconditioner. Partitioning the coefficient matrix **A** into blocks

$$\mathbf{A} = \begin{pmatrix} \mathbf{D}_R & \mathbf{H}_{RB} \\ \mathbf{H}_{BR} & \mathbf{D}_B \end{pmatrix}$$

(the indices R denote the red, B the black points) the 'reduced system' is obtained by eliminating one of both sets of (e.g. the red) unknowns:

$$(\mathbf{D}_B - \mathbf{H}_{BR}\mathbf{D}_R{}^{-1}\mathbf{H}_{RB})\, x_B = b_B - \mathbf{H}_{BR}\mathbf{D}_R{}^{-1} b_R \tag{1}$$

The preconditioner is then the main diagonal of the reduced system

$$\tilde{\mathbf{D}}_B = \mathbf{D}_B - diag\,(\mathbf{H}_{BR}\mathbf{D}_R{}^{-1}\mathbf{H}_{RB}) \tag{2}$$

resulting in a symmetric diagonal scaling:

$$\tilde{\mathbf{D}}_B^{-\frac{1}{2}} (\mathbf{D}_B - \mathbf{H}_{BR}\mathbf{D}_R{}^{-1}\mathbf{H}_{RB})\, \tilde{\mathbf{D}}_B^{-\frac{1}{2}} \tilde{\mathbf{D}}_B^{\frac{1}{2}} x_B = \tilde{\mathbf{D}}_B^{-\frac{1}{2}} (b_B - \mathbf{H}_{BR}\mathbf{D}_R{}^{-1} b_R) \tag{3}$$

Thus, scaling the red unknowns by the square root of main diagonal $\mathbf{D}_R$ we may write the preconditioned system in the form

$$\left(\hat{\mathbf{D}}_B - \hat{\mathbf{H}}_{BR}\hat{\mathbf{H}}_{RB}\right) \hat{x}_B = \hat{b}_B \tag{4}$$

where hats ($\hat{}$) denote preconditioned quantities. Approximately half of the virtual processors in the Connection Machine are inactive during each stage.
Matrix polynomials as preconditioners were originally proposed in order to remove the recursions in the incomplete LU factors [2]. The construction of least squares polynomials that minimize the residual with respect to some weight function [8] is feasible due the positive real spectrum of the coefficient matrices of the uncoupled discrete device equations. The evaluation of matrix polynomials can be carried out recursively and is a fast operation especially on an SIMD architecture.
In Figure 1 we present convergence results obtained from the majority continuity equation solve of the first nonlinear iteration of device 1 (see section 5). The 2-norm of the relative euclidean solution error is plotted using the BiCGSTAB iterative procedure with various preconditioners: Point Jacobi (JA) i.e. diagonal scaling, the Reduced System (RS), Least Squares polynomials LS(i) up to degree 8, and incomplete LU (ILU) with zero fill-in.

4. Installation on the CM. MINIMOS is a standard FORTRAN 77 program. A two-dimensional simulation provides the initial data prior to the three-dimensional run. The maximum number of gridpoints in each dimension of the self-adaptive grid is 64 nodes. A massive parallelization throughout the whole MINIMOS code for application on the CM2 requires a rewrite of the largest part of the MINIMOS source code in connection machine FORTRAN (CMF) and was not the purpose of this work. Such a rewrite is very time-consuming, of little scientific value, but necessary for a production code on the Connection Machine. A compromise was obtained by leaving the largest part of the MINIMOS code, essentially all parts except the linear system solvers, in the FORTRAN 77 standard and using the Connection Machine for the solution of the linear systems only. Such an implementation achieves parallelization of the most costly computational part – the linear system solution – at relatively low programming costs. That is, the matrix assembly is done on the so-called front-end computer, and the linear system solution is achieved on the Connection Machine. The Connection Machine acts so to speak as a 'coprocessor' for linear system solving. An interface using routines from the CMF library was provided to quickly move binary data to and from the Connection Machine's processors. The matrices are stored by diagonals. For a 262144 node mesh the move of 9 diagonals (matrix+rhs+solution) per solve takes approx. 40 seconds. This time which is independent of the number of processors allocated is in most cases larger than the solution of the linear systems within the CM2. In an actual production code such an IO bottleneck must

clearly be avoided.

The linear solvers RS-CG and the RS-BiCGSTAB methods have been benchmarked. Some results are shown in Table 2: Three tensor product grids of 16^3, 32^3 and 64^3 points are chosen. The Laplace equation is solved on 8k, 16k and 32k processors using double precision arithmetic.

Using 32k processors we find a megaflop rate 2of 120 for for the $N = 64^3$ grid. Note that only half of the virtual processors are active due to the red-black mask. Obviously this speed is determined by the speed of the 7 point stencil which executes with roughly 200 megaflops. Correspondingly we obtain approximately 50 megaflops on 16k processors for the same grid. The low speed of the 7-point stencil computation is the reason why polynomial preconditioners are not (yet ?) competitive. Despite global communication the dotproduct achieves a rate of 300 megaflops. This makes polynomial preconditioners compare unfavorable. We use the RS preconditioner for our device simulation runs therefore.

FIG. 1. *Convergence Curves for the Discrete Majority Continuity Equation of Device 1*

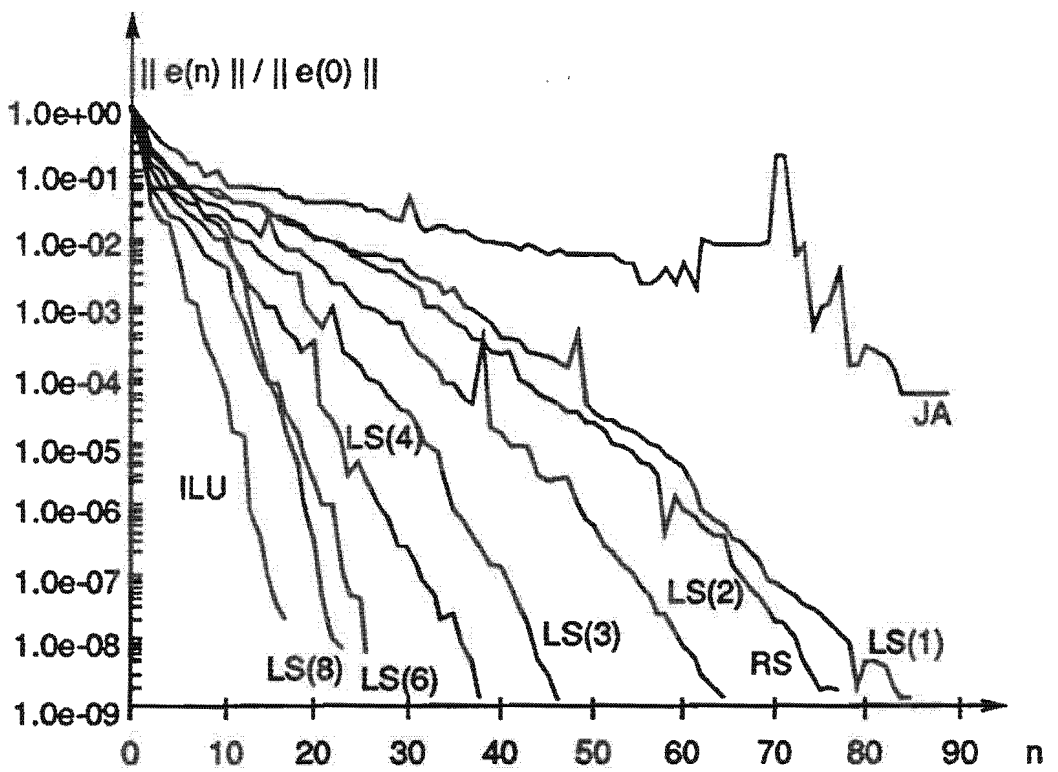

TABLE 2
RS-CG and RS-BiCGSTAB Benchmarks (msec per iteration)

Solver	RS–CG			RS-BiCGSTAB		
# Processors	8k	16k	32k	8k	16k	32k
$16 \times 16 \times 16$	10	10	10	20	20	20
$32 \times 32 \times 32$	40	20	10	50	30	20
$64 \times 64 \times 64$	190	110	60	300	160	90

TABLE 3
Performance of MINIMOS on 16k Processors

	Device 1	Device 2	Device 3
Meshpoints	12880	18144	171985
Nonlin. Iter.	5	18	26
Total CPU Time (s)	72	216	5700
Majority CC Lin. Iter.	415	5328	9880
Majority CC Time (s)	8	100	1680
Minority CC Lin. Iter.	3160	5076	22360
Minority CC Time (s)	59	97	3800
Poisson Lin. Iter.	115	540	1014
Poisson Time (s)	5	20	212

5. Performance Analysis. In this section performance measurements of MINIMOS on a 32k processor Connection Machine CM2 are given. 16k processors are used in all simulations. The reduced system conjugate gradient method is used for the Poisson equation, and the reduced system conjugate gradient squared (RS-BiCGSTAB) method is used for both carrier continuity equations. We present three device simulation runs of different complexity. Device 1 is an NMOSFET in the subthreshold operating region, device 2 a PMOSFET in the saturation region and device 3 an NMOSFET in saturation with a complex shape of the oxide-semiconductor interface thus requiring a fine mesh. The simulation includes a selfconsistent treatment of impact ionization and carrier recombination.
In Table 3 performance statistics of the three simulations are listed. The rows in Table 3 have the following meaning: Row 1 shows the grid dimensions, row 2 the number of nonlinear (Gummel) iterations. Row 3 through row 9 shows the total times and iteration counts of the linear iterative solvers for the majority and minority carrier continuity (CC) equation and the Poisson equation. The timings do *not* include front-end computation and transfer times to and from the CM2.
As has been outlined in the previous section the implementation using straightforward FORTRAN code and fieldwise data is far from optimality. The linear solvers execute with approximately 60 megaflops on 16k processors. In the case of a mean convergence degradation factor of 4 (RS vs. ILU) as e.g. in device 3 this corresponds to a 15 megaflop scalar machine using MIC-CG/ILU-BiCGS. For a detailed description of numerical results see [7].

6. Some Remarks and Conclusions. The implementation on the Connection Machine has been carried out in a relatively short time by porting the computationally most expensive code onto the Connection Machine. We have learned that the Connection Machine is easy to use and to program although re-programming of selected parts of the software is required.
Complex simulations, say, tasks exceeding 100000 unknowns execute well on the CM2 (no memory problems). However the matrix assembly on the front end computer becomes disastrous slow. Furthermore, the transfer time of the diagonally stored 7-band matrix and the right hand side to and the solution vector from the Connection Machine's processors is in most cases slower than the solution of the linear system itself. For a production code this is not acceptable. Therefore an interaction of the front-end computer with the Connection Machine's processors, which involves computation and transfer of large sets of binary data, must be avoided.
Thinking Machines Corporation is improving its FORTRAN compiler, using 'slicewise' data. Optimization of many basic linear algebra subroutines (such as multiwire nearest neighbour communication for stencil operations, overlapping of communication and computation) is under way. A speedup of CMF-coded FORTRAN code by a factor between 4 to 8 seems realistic. A one gigaflop execution speed of the linear CG and BiCGSTAB solvers is expected. Assuming a mean convergence deterioration factor of 4 corresponding to the ILU(0) preconditioned solvers, this corresponds to 250 megaflops of MIC-CG and ILU-BiCGSTAB. In a recent work [13] high performance implementations of MINIMOS on vector-computers were presented. It has been shown that megaflop rates exceeding 100 megaflops are obtainable for the MIC-CG and ILU-BiCGS solvers. E.g. the FUJITSU VP200 can execute the triangular solves of the ILU(0) preconditioner, the bottleneck of the overall computation, with 100 megaflops. We expect that a fully optimized code on the CM should thus be at

least twice as fast.

The convergence loss factors of the iterative solvers in the presented simulations are larger than three, in some cases larger than 10. A convergence degradation of more than a factor of 10 makes the question of a potential superiority of the Connection Machine or similar massively parallel computer architectures over vector-supercomputers doubtful. Part of this convergence dilemma is believed to be an insufficient 3D mesh. The ILU preconditioner can much more efficiently handle grids that deviate to much from quasi-uniformity. An improved grid generation with respect to conditioning of the linear equations may alleviate this problem and is part of current investigations.

Parallel preconditioning concerning the semiconductor equations is in its early stages. Black-box solvers such as those used in this investigation may not be the ultimate answer. Concepts that are more flexible in nature such as domain decomposition methods, are likely to yield better results on parallel computers. Further research in this field is encouraged.

The following final statements are meant to summarize what we believe a potential user of MINIMOS, e.g. a process/device engineer at a semiconductor manufacturing site, would expect from a Connection Machine implementation: At first double precision hardware is an absolute prerequisite for any implementation. This is not only true for MINIMOS but also for all process simulators, a class of software tools which are also candidates for Connection Machine implementations.

A Connection Machine system with a large number of processors is desirable. On the one hand this provides a powerful computing basis for very large three-dimensional problems, on the other hand a large number of processors may be subdivided, thus enabling more than one simulation to be executed concurrently on the Connection Machine. Concurrency of this type is essential to any integration of simulators such as MINIMOS into a technology CAD framework.

Fast transfer of binary data to and from the Connection Machine's processors is a key requirement for the use of the Connection Machine in a server-client manner. The transfer speeds for large sets of binary data between the Connection Machine and the front-end computer measured in this work are rather slow. We note that modern device/process CAD environments, are based on a computer network, which consists of high performance workstations as visualization tools and supercomputers as the device/process equation solvers. The effectiveness of such a configuration depends quite critically on a fast interconnection between the workstations and the supercomputer hardware.

REFERENCES

[1] Bank, R.E., Rose, D.J., Fichtner, W., "Numerical Methods for Semiconductor Device Simulation", *IEEE ED-30*, pp. 1031-1041, 1983

[2] Dubois, P.F., Greenbaum, A., Rodrigue, G.H., "Approximating the Inverse of a Matrix for Use in Iterative Algorithms on Vectors Vector Processors", *Computing*, Vol. 22, pp. 257-268, 1979.

[3] Eisenstat, S.C., "Efficient Implementation of a Class of Preconditioned Conjugate Gradient Methods", *SIAM J.Sci.Stat.Comput.*, Vol. 2, No. 1, Mar. 1981, pp. 1-4.

[4] Guerrieri, R., et. al., "Massively Parallel Algorithms for Three-Dimensional Device Simulation", Proceedings NUPAD 2, July 1990, pp. 35-36

[5] Gummel, H.K., "A Selfconsistent Iterative Scheme for One-dimensional Steady State Transistor Calculations", *IEEE ED-11*, pp. 455-465, 1964.

[6] Heinreichsberger, O., Selberherr, S., Stiftinger, M., Traar, K.P., "Fast Iterative Solution of Carrier Continuity Equations for 3D Device Simulation", to appear in *SIAM J.Sci.Stat.Comput.*.

[7] Heinreichsberger, O., "MINIMOS on the Connection Machine", Technical Report, February 1991, Institute for Microelectronics, Technical University Vienna, AUSTRIA.

[8] Johnson, O.G., Micchelli, A., Paul, G., "Polynomial Preconditioners for Conjugate Gradient Calculations", *SIAM J.Numer.Anal.* Vol. 20, No. 2, Apr. 1983, pp 362-376.

[9] Meijerink, H., Vorst, H., "An Iterative Solution Method for Linear Systems of which the Coefficient Matrix is a Symmetric M-Matrix", Math. Comp., 31 (1977), pp. 148-162.

[10] Oppe, T.C., Joubert, W.D., and Kincaid, D.R., "NSPCG User's Guide.", Center of Numerical Analysis, University of Texas at Austin, 1984.

[11] Poole, E.L., Ortega, J.M., "Multicolor ICCG Methods for Vector Computers", *SIAM J.Numer. Anal.*, Vol. 24, No. 6, Dec. 1987, pp. 1394-1418.

[12] Sonneveld, P., "CGS, A Fast Lanczos-Type Solver for Nonsymmetric Systems", *SIAM J.Sci.Stat.Comput.*, Vol. 10, No. 1, Jan. 1989, pp. 36-52.

[13] Traar, K.P., Mader, W., et. al., "High Performance Preconditioning on Supercomputers for the 3D Device Simulator MINIMOS.", Proceedings Supercomputing '90, pp. 224-231.

[14] Vorst, H.A. "Bi-CGSTAB: A Fast and Smoothly Converging Variant of BiCG for the Solution of Nonsymmetric Linear Systems.", to appear in *SIAM J.Sci.Stat.Comput.*.