THREE DIMENSIONAL MONTE CARLO DEVICE SIMULATION WITH PARALLEL MULTIGRID SOLVER

C. K. SANDALCI and Ç. K. KOÇ
Electrical & Computer Engineering
Oregon State University
Corvallis, OR 97330, USA

S. M. GOODNICK
Electrical Engineering
Arizona State University
Tempe, AZ 85287, USA

ABSTRACT
We present the results in embedding a multigrid solver for Poisson's equation into the parallel 3D Monte Carlo device simulator, PMC-3D. First we have implemented the sequential multigrid solver, and embedded it into the Monte Carlo code which previously was using the sequential successive overrelaxation (SOR) solver. Depending on the convergence threshold, we have obtained significant speedups ranging from 5 to 15 on a single HP 712/80 workstation. We have also implemented the parallel multigrid solver by extending the partitioning algorithm and the interprocessor communication routines of the SOR solver in order to service multiple grids. The Monte Carlo code with the parallel multigrid Poisson solver is 3 to 9 times faster than the Monte Carlo code with the parallel SOR code, based on timing results on a 32-node nCUBE multiprocessor.

Keywords: Semiconductor device simulation, Monte Carlo methods, Multigrid solvers, Poission's equation, parallel computation.

Short title: 3D Parallel Monte Carlo.

1. Introduction. Semiconductor device simulation is an important aspect of the computer aided design (CAD) of integrated circuits. As semiconductor device dimensions continue to shrink in ultra-large scale integration technology, there is an increasing need for full, three-dimensional (3D) device models to accurately represent the physical characteristics of the device. Further, as dimensions shrink and the internal electric fields increase, approximate solutions to the semiconductor transport equation (the Boltzmann equation when quantum effects are negligible) based on low order moment methods (e.g. the drift-diffusion model) are no longer applicable. Solution of the Boltzmann equation using Monte Carlo methods is currently the most widespread technique used in device simulation at this level of modeling [1]. In the Monte Carlo method, the motion of charge carriers (electrons and holes) is assumed to be given by classical trajectories interrupted by random, instantaneous, scattering events which change the energy and momentum of the particles. The classical trajectories are calculated according to the instantaneous forces acting on the particles over a small time step. In a device simulation, these forces are determined by the electric fields obtained by solving Poisson's equation on a mesh over the device domain. The random scattering events are generated stochastically using a random number generator and the quantum mechanical scattering probabilities for all possible mechanisms in the semiconductor. In a Monte Carlo device simulation, the solution of the particle motion (referred to here as the Monte Carlo phase) is synchronized with the solution of Poisson's equation so as to provide an accurate representation of the time dependent evolution of the fields in the semiconductor, which in turn accelerate the carriers over each time step. It is necessary to solve Poisson's equation at various time intervals, and therefore the above algorithm is basically a time-domain solution of the transport and field equations in the device.

The computational burden of using such Monte Carlo techniques is quite high, particularly when combined with the simultaneous solution of Poisson's equation in 3D. Alternate particle methods for solving the Boltzmann equation using lattice-gas cellular-automaton have demonstrated considerable speedup compared to the Monte Carlo technique [2]. However, this technique does not alleviate the computational burden of solving Poisson's equation, which in 3D may become the principal bottleneck in the calculation.

Parallel or multiprocessor computers provide some relief to the computational requirements of Monte Carlo device simulation. We have previously developed a parallel 3D Monte Carlo device simulator, PMC-3D [3], which was implemented on the distributed-memory nCUBE multiprocessor. In this algorithm, a subspace decomposition of the 3D device domain was performed, in which the particles and mesh nodes were distributed in a load-balanced way among the individual processors. During each time step, the particle motion and field calculation is performed locally, and the results communicated to neighboring processors at the end of the time step. In order to parallelize the solution of Poisson's equation in this initial implementation, an iterative successive over relaxation (SOR) method with a red-black ordering scheme was used. We have obtained good efficiencies using this algorithm, up to 70 % with 512 processors. Our subsequent analysis of the code has revealed the fact that nearly 60–90 % of the computation time is spent in the the Poisson solver for simulating real 3D structures [4].

Significant speedup of 2D Poisson-Monte Carlo algorithm has been reported by Saraniti et al using multigrid methods [5]. In the multigrid method discussed in Section 3, the convergence of the Gauss-Seidel iteration (which is the basis of the SOR method), is accelerated through the use of coarser grids on which the residual is solved. In their serial implementation, speedups of 10-20 times were reported compared to the SOR method [5].

In this paper, we describe a parallel implementation of the multigrid Poisson solver in the 3D Monte Carlo device simulator PMC-3D. Section 2 briefly describes the previous parallel SOR implementation while Section 3 reviews the multigrid method itself. Section 4 describes the implementation of this algorithm including its parallelization while Section 5 describes the results. First we compare the sequential multigrid Monte Carlo code to the sequential SOR Monte Carlo code. Depending on the convergence threshold, we have obtained significant speedups ranging from 5 to 15 when PMC-3D code is executed on a single HP 712/80 workstation. Furthermore, the parallel multigrid Monte Carlo implemented on a 32-processor nCUBE is faster between 3 to 9 times than the parallel SOR Monte Carlo code.

2. The Multigrid Method. A Monte Carlo device simulator requires the solution of Poisson's equation to obtain the spatial distribution of the potential and electric fields which accelerate particles during the Monte Carlo particle phase. In this section, we discuss the basic aspects of the multigrid method, and then explain the details of the implementation The primary emphasis will be on the three dimensional Poisson's equation which is formulated as

$$\frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} + \frac{\partial^2\phi}{\partial z^2} = \frac{\rho}{\epsilon_s} \ ,$$

and its finite-difference discretization

$$\frac{u_{x-1yz} - 2u_{xyz} + u_{x+1yz}}{h_x^2} + \frac{u_{xy-1z} - 2u_{xyz} + u_{xy+1z}}{h_y^2} + \frac{u_{xyz-1} - 2u_{xyz} + u_{xyz+1}}{h_z^2} = -\frac{\rho_{xyz}}{\epsilon_s} \ .$$

Here, h_x, h_y, and h_z are the grid spacings in the x, y, an z axes respectively. The plus and minus signs in the subscript denote the different directions.

The multigrid technique is a well-established approach for solving ordinary and partial differential equations. Its main advantage over other iterative methods like the SOR is that it is immune to increasing grid point numbers and/or more accurate convergence thresholds [6, 7, 12, 9]. Here we describe the main idea behind the multigrid approach, taking the three dimensional Poisson's equation as an example. The Poisson's equation can be expressed as follows

$$L\mathbf{u} = \mathbf{f} \ ,$$

where L represents the ∇^2 operator, $\mathbf{u}$ is the potential distribution, and $\mathbf{f}$ is the normalized charge distribution, $\rho(x, y, z)/\epsilon_s$. Let $\mathbf{v}$ denote the approximation to $\mathbf{u}$, and $\mathbf{e}$ denote the corresponding error, where $\mathbf{e} = \mathbf{u} - \mathbf{v}$. In this case, the residual $\mathbf{r}$ is defined as

$$\mathbf{r} = \mathbf{f} - L\mathbf{v} \ ,$$

where $L\mathbf{v}$ is the approximation to the forcing function $\mathbf{f}$. It is easy to show that the error $\mathbf{e}$ obeys the so-called the residual equation

$$L\mathbf{e} = \mathbf{r} \ .$$

Let $L_n\mathbf{u}_n = \mathbf{f}_n$ denote the finite difference discretization of the Poisson's equation on the grid, Ω_n and the next coarser grid be Ω_{n-1}. The simplest multigrid approach is the two level coarse grid correction. In this scheme, the residual $\mathbf{r}$ is first transferred to the next coarser grid as

$$\mathbf{r}_{n-1} = \mathbf{I}_n^{n-1}\mathbf{r}_n \ ,$$

where $\mathbf{I}_n^{n-1}$ is the residual weighting or restriction which is a fine to coarse transfer operator. Then the residual equation on the coarse level

$$L_{n-1}\mathbf{e}_{n-1} = \mathbf{I}_n^{n-1}\mathbf{r}_n \ ,$$

is solved exactly, either by means of an iterative method such as SOR, or directly. L_{n-1} is some coarse grid approximation to the dense grid Laplacian, L_n, which corresponds to the same finite difference discretization of the problem on the coarser grid. After the residual equation is solved on the coarse level, the error is interpolated to the dense grid. This error component is then added as a correction to $\mathbf{v}_n$ as

$$\mathbf{v}_n' \leftarrow \mathbf{v}_n + \mathbf{I}_{n-1}^n L_{n-1}^{-1}\mathbf{I}_n^{n-1}\mathbf{r}_n \ .$$

The advantage of this scheme comes from the error smoothing effect of the relaxation operators [10, 11]. In the Fourier domain, the low frequency components of the error vector are slightly reduced while the high frequency components practically vanish in a few relaxation sweeps. On the coarse grid, however, some of these low frequency components overlap with high frequency components due to aliasing. The same relaxation scheme can reduce these overlapped components on the coarse grid. A simple two-level coarse grid correction cycle can be described as follows:

1. Pre-smoothing: $\mathbf{v}_n \leftarrow S_n^{v_1}\mathbf{v}_n$.
2. Calculate the residual: $\mathbf{r}_n = \mathbf{f}_n - L\mathbf{v}_n$.
3. Restriction: $\mathbf{f}_{n-1} \leftarrow \mathbf{I}_n^{n-1}\mathbf{r}_n$.
4. Solve exactly on Ω_{n-1}: $\mathbf{u}_{n-1} = L_{n-1}^{-1}\mathbf{f}_{n-1}$.
5. Interpolation: $\mathbf{e}_n \leftarrow \mathbf{I}_{n-1}^n\mathbf{u}_{n-1}$.
6. Correction: $\mathbf{v'}_n \leftarrow \mathbf{v}_n + \mathbf{e}_n$.
7. Post-smoothing: $\mathbf{v'}_n \leftarrow S_n^{v_2}\mathbf{v'}_n$.

Here S_n^k denotes k relaxation sweeps of an appropriate relaxation scheme. The details about the interpolation, restriction and smoothing operators will be discussed in the next part of this section. The equation in step 4 has the same form as the original equation, $L\mathbf{u} = \mathbf{f}$. Applying the entire procedure recursively γ times in step 4, one can produce different multigrid cycles, e.g., the V-Cycle for $\gamma = 1$ or the W cycle for $\gamma = 2$, as illustrated in Fig. 1. Using W cycles with a pointwise Gauss-Seidel relaxation scheme and a homogeneous grid with uniform spacing gives the best performance upgrade in a reasonable development time.

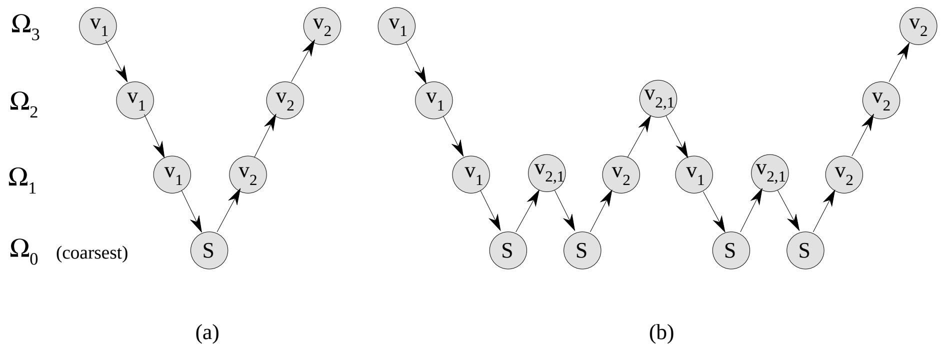

FIG. 1. *Two common multigrid cycles are* (a) *V Cycle ($\gamma = 1$) and* (b) *W Cycle ($\gamma = 2$). Here v_1 denotes pre-smoothing and v_2 denotes post-smoothing. Also S is the exact solution operator, $\searrow$ is the fine to coarse grid restriction operator, and $\nearrow$ is the coarse to fine grid prolongation operator.*

3. Implementation. In this section, we discuss the implementation details of the multigrid Poisson solver. The coarsening scheme, intergrid transfer operators, relaxation scheme, discretization and the parallelization of the method are explained in the following parts.

3.1. Coarsening. For the multigrid approach, the choice of the grid set is crucial. The first task is to create a hierarchical set of grids ranging from the densest Ω_n to the coarsest possible Ω_k. The coarsening factor we used is 1/2, which implies that the grid spacing of Ω_{n-1} is twice as big as the grid spacing of Ω_n. Fig. 2 illustrates the two dimensional representation of the multiprocessor coarsening scheme. Determining the coarsest possible level is another important aspect. As long as the boundary conditions of the original grid can be represented on a coarser grid, coarsening is allowed. Dirichlet boundary conditions need to be mapped to all grids with at least one boundary point per contact. Let the grid point at (x_1, y_1, z_1) belong to an electrical contact with the potential value ϕ_a. Then the boundary value on the densest grid is the contact potential ϕ_a. On the coarser levels, we are trying to approximate the error on this potential value. The potential at the contact is fixed and known exactly, and thus, the corresponding error on the coarser grids must be zero, i.e.

$$\phi^n_{x_1,y_1,z_1} = \begin{cases} \phi_a & n = 0 \quad \text{(the densest level)} \\ 0 & n \neq 0 \quad \text{(on all other levels)} \end{cases}$$

The Neumann boundaries are treated the same way on the entire grid set, and their mapping is not as crucial as the Dirichlet boundaries [5]. The multigrid method does not have any restrictions concerning the total number of grid-points. However, choosing the number of points of the form 2^k+1 for all three directions (but not necessarily with equal k values) would simplify the restriction and the prolongation operators and improve the convergence ratio of the Poisson Solver.

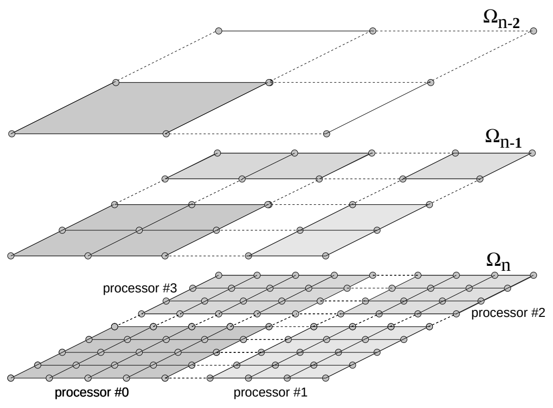

FIG. 2. *Two dimensional representation of the multiprocessor coarsening scheme. Here Ω_n is the densest, Ω_{n-1} is the next coarser, and Ω_{n-2} is the coarsest grid.*

3.2. Restriction and Prolongation. Another important component of the multigrid method is the restriction and prolongation operators. After generating the hierarchical grid set, the next step is designing the tools for residual transfers from coarser to finer grid and the opposite way for the error.

The prolongation operator we used is a modified version of the nine point prolongation used in the two dimensional case. The three cases for the prolongation operation are shown in Fig. 3a. The arrows denote the contributing coarse grid points, where the attached numbers are the corresponding weighting factors.

The restriction operator is a little more difficult to implement. There are two different useful approaches, namely the full weighting and the half weighting restriction [12, 9]. In our experience, a full weighting residual transfer operator is necessary for a stable solution. In Fig. 3b, the dense grid points that take part in the regular full weighting scheme are listed with the corresponding weighting factors. Although there are 27 points to be considered, the nature of the red/black ordered Gauss-Seidel relaxation scheme allows us to concentrate on 13 of those points as the residual values for the last updated color are always zero.

3.3. Relaxation Method. The main goal of the relaxation scheme is to reduce the high frequency components of the error on any given grid. There can be several suitable relaxation schemes for a specific problem depending on the boundary conditions and/or coarsening method. In this implementation, we chose to use a pointwise Gauss-Seidel relaxation scheme.

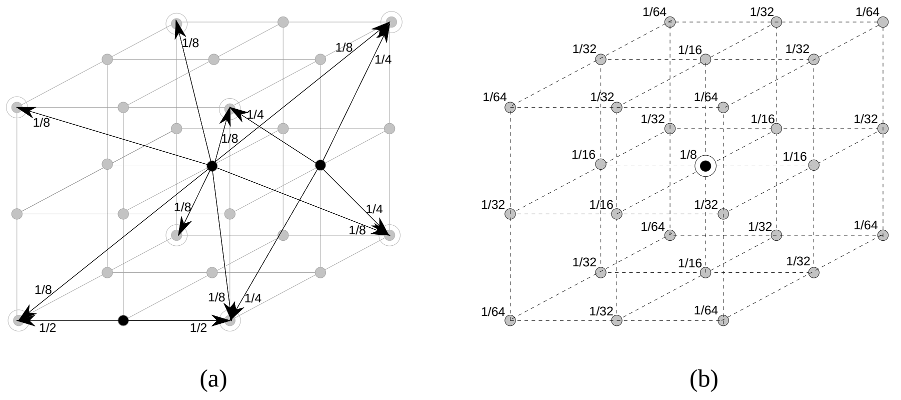

FIG. 3. *The intergrid transfer operators.* a) *Restriction: A 27-point full weighting scheme is used. The number in front of each grid point denotes its weight in this operation.* b) *Prolongation: The arrows denote the coarse grid points to be used for interpolating the dense grid point. The numbers attached to the arrows denote the contribution of the specific coarse grid point.*

The efficiency of a relaxation scheme can be measured by the smoothing factor [10, 11]. For a cubic grid with $N \times N \times N$ grid points with periodic boundary conditions, the Fourier transform of the error $\mathbf{e}$ is given by

$$\mathbf{e}_{x,y,z} = \sum_{r,s,t=-N/2+1}^{N/2} c(\theta_r, \theta_s, \theta_t) \exp\left[i(\theta_r x + \theta_s y + \theta_t z)\right] ,$$

where $\theta_r = r\pi/N$, $\theta_s = s\pi/N$, $\theta_t = t\pi/N$, and c is the magnitude of the frequency component of the error for a given frequency. The amplification factor of the $\theta_{\{r,s,t\}}$ component due to one relaxation is

$$\mu(\theta) = \left| \frac{\overline{c}(\theta)}{c(\theta)} \right| ,$$

where $\theta = (\theta_r, \theta_s, \theta_t)$, and $\overline{c}$ represent the frequency components of the error after the relaxation sweep. Finally the smoothing factor is defined by

$$\mu = \max_{\rho\pi \leq |\theta| \leq \pi} \mu(\theta) ,$$

where ρ is the grid coarsening factor. A double coarsening scheme implies $\rho = 1/2$. The pointwise Gauss-Seidel relaxation over a cubic grid has a typical smoothing factor $\mu \simeq 1/2$ [12]. This implies that the high frequency components of the error are reduced by almost an order of magnitude, in three relaxation sweeps. This smoothing rate is achieved only for the non-degenerate case where the grid spacings in all three dimensions are the same.

The reason for the poor smoothing effect in the case of nonuniform and inhomogeneous grids comes form the fact that a pointwise relaxation scheme has a smoothing effect only with respect to the direction that has the smallest grid spacing. Thus, for a decent smoothing effect, according to the various configurations of the grid spacings, line and/or plane relaxations are required, which are difficult to implement in parallel. As the multigrid solver is designed to be a replacement for the former SOR solver, we chose to use a pointwise red-black ordered Gauss-Seidel relaxation scheme and restricted the grids to be homogeneous and uniformly spaced along all three dimensions.

4. Parallelization. Several parallel implementations of the multigrid method has been reported in the literature [12, 13, 14]. Our parallelization of the multigrid code is essentially the same as the former SOR implementation. The partitioning and the communication routines are extended to service the hierarchical grids, hence the communication pattern is preserved.

In the former SOR implementation as well as in our parallel multigrid solver, the device is partitioned using a recursive bisection algorithm as illustrated in Fig. 4. The domain is split into two parts that represent roughly equal amount of work, and each part is recursively divided until the desired number of subgrids are obtained. The subgrids are assigned to processors using a Gray code mapping. Thus, communication occurs only between pairs of processors that are physically adjacent to one another.

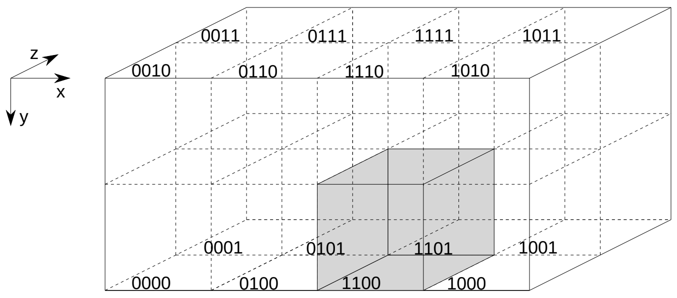

FIG. 4. *The geometrical partitioning of the semiconductor device domain onto a hypercube of 16 processors. The processors are labeled using binary numbers.*

The two dimensional representation of the SOR communication pattern is shown in Fig. 5. Since the Gauss-Seidel relaxation operator is simply the SOR with $\omega = 1$, the communication pattern of the smoothing operator remains unchanged [3, 4]. There are three main events in the multigrid code where communication among adjacent processors needs to occur before the operation is performed.

Each relaxation sweep for both smoothing and exact solution (SOR) operations consist of two half sweeps corresponding to the two orderings of the grid points. During these operations, each processor updates potential values of the grid points belonging to the subgrid mapped to its memory. Before each half sweep, the processors need to communicate via message passing with their neighboring processors. This way the potentials of the oppositely colored grid points external to the processor's subgrid are obtained.

After the smoothing operation is performed, the residual values are calculated. Due to red-black ordering, the residual values of the last updated grid set is always zero.

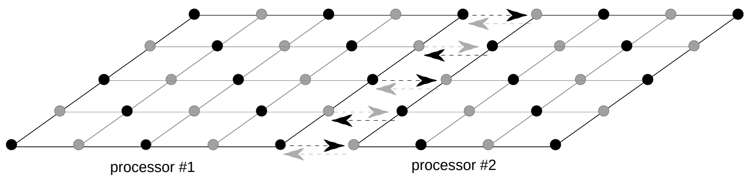

FIG. 5. *The 2-D representation for the red-black ordered SOR solver and Gauss-Seidel relaxation. The light and dark arrows represent the communication attempts before relaxing the red and black ordered points respectively.*

Before the residual restriction is performed, each processor again communicates with its neighboring processors to obtain the non-zero residual values of the grid points external to its subgrid. This way a correct restriction to the coarser levels is achieved.

The same situation is valid for the prolongation operator as well. The prolongation operation is performed after either a post-smoothing or an exact solution operation. In our implementation, these two operations, although different in functionality, are very similar. The exact solution operation is nothing but the former SOR solver applied to the coarsest level. Before the prolongation is performed, each processor communicates with its neighboring processors to obtain the updated potentials of the grid points external to its subgrid. Then the prolongation operation is performed and the error is interpolated to the finer levels.

Another issue in developing a parallel semiconductor device simulation package is distribution and load balancing. As noted earlier, a montecarlo device simulation can be divided into two phases. Even with the speedups presented in this paper, the solution of the Poisson's equation still is the dominating computational part of the package. Thus, load balancing in the multigrid solver which is easily achieved by recursive bisection algorithm, is necessary and justified. The interested reader is referred to [3] for issues concerning different approaches for load balancing during the montecarlo phase.

The PMC-3D simulation package with the SOR solver exhibits a high level of scalability [3]. Although the overall communication load of the multigrid solver is larger than the one of the SOR solver, the parallel code is still scalable, but the slope of the speedup versus problem size curve is slightly decreased. Possible approaches to decrease the communication load of the MG solver will be discussed in the next section.

5. Results and Discussion. In this section we present the results of our experiments in simulating the MESFET device structure shown in Fig. 6. We executed the PMC-3D code with both the SOR and the multigrid solver in order to compare their timings.

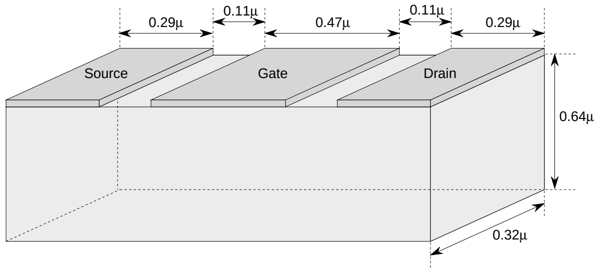

FIG. 6. *The MESFET structure used as a model problem for the simulations.*

We executed the PMC-3D code for a number of time steps of the Poisson/Monte Carlo solver to simulate an actual MESFET device run. The MESFET was started from an initial charge neutral state, and the applied voltage turned on abruptly at $t = 0$. Thus, the first time step requires a significantly greater number of iteration cycles for the Poisson solver to converge. Subsequent time steps require fewer iteration cycles as the initial guess is provided from the solution to the previous timestep. To compare the effectiveness of the multigrid solver, we recorded the total simulation time and the time spent in solving the Poisson's equation using both the SOR and multigrid solvers. Table 1 gives the timings in seconds when PMC-3D code is executed on a single HP 712/80 workstation. The simulation is run for 100 time steps with convergence thresholds for the potential ranging from 10^{-3} down to 10^{-12} on a $129 \times 65 \times 33$ homogeneous grid with uniform grid spacings. A total of 32,000 particles was used in the simulation. As seen in Table 1, the multigird Poisson solver is about 7–16 times faster than the SOR solver depending on the convergence threshold. For smaller convergence thresholds, the number of iterations of the SOR solver becomes quite large, whereas, as discussed in the previous sections, the multigrid converges much more rapidly due to the error smoothing on the coarser meshes.

	PMC-3D with SOR		PMC-3D with MG		Speedup	
Threshold	Poisson	Total	Poisson	Total	Poisson	Total
10^{-3}	$14,879.82$	$15,595.59$	$2,145.84$	$3,117.65$	6.93	5.00
10^{-6}	$76,208.28$	$77,029.05$	$5,664.77$	$6,576.83$	13.45	11.71
10^{-9}	$153,118.90$	$153,952.04$	$9,779.32$	$10,728.29$	15.65	14.35
10^{-12}	$225,867.00$	$226,735.04$	$14,160.49$	$15,124.02$	15.95	14.99

TABLE 1. *The timings of the PMC-3D with SOR and MG solvers on a single HP 712/80 workstation.*

As can be seen in Table 1, the total time of execution and the Poisson execution time are fairly close for the chosen number of grid points and particle number. Even for a convergence threshold of 10^{-3}, the Poisson solver uses 95% of the execution time. Thus, the overall speedup of the total simulation is 5–15 times, which is close to the speedup of the Poisson solver itself.

Table 2 shows the timing results of parallel SOR and multigrid PMC-3D codes on a 32-processor nCUBE multiprocessor. The PMC-3D code is again run for 100 time steps on the same grid with 20,000 particles. As can be seen from Table 2, the parallel multigrid PMC-3D code is 3–9 times faster than the parallel SOR PMC-3D. In this case, the multigrid solver is found to be 5–9 times faster than the SOR solver. The difference in the speedup values between the serial and the parallel cases seems to arise from the fact that the communication load for the multigrid solver is higher than that of SOR. The amount of data transferred among adjacent processors decreases with increasing grid-spacing in the multigrid method, however, the number of communication attempts and the total number of iterations are generally higher than those in the SOR solver.

	PMC-3D with SOR		PMC-3D with MG		Speedup	
Threshold	Poisson	Total	Poisson	Total	Poisson	Total
10^{-3}	2,917.611	3,340.020	596.367	1,121.035	4.89	3.05
10^{-6}	15,093.156	15,515.990	2,064.199	2,589.482	7.31	5.99
10^{-9}	31,167.143	31,653.002	3,486.319	4,011.031	8.94	7.89
10^{-12}	45,597.321	46,083.452	4,927.825	5,453.801	9.25	8.45

TABLE 2. *The timings of the PMC-3D with SOR and MG solvers on the 32-node nCUBE multiprocessor.*

As mentioned in section 2, the computation time of the multigrid solver increases only linearly with respect to the decrease in the convergence threshold. However, the computation time of the SOR solver tends to grow exponentially. This effect can be seen in Fig. 7, in which we plot the computation time as a function of the convergence threshold. Therefore, the speedup improves for the multigrid versus SOR as the convergence threshold is decreased.

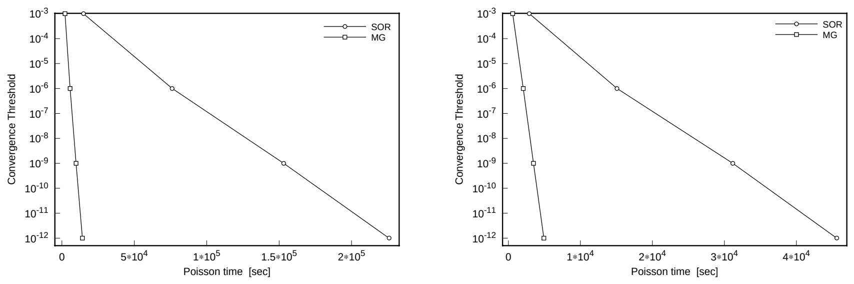

FIG. 7. *The computation time versus convergence threshold for the serial code running on a HP712/80 and the parallel code running on an nCUBE with 32 nodes.*

In conclusion, we have presented the results of embedding a multigrid solver into our PMC-3D code in place of the SOR solver for solving Poisson's equation. We have obtained speedups between 6 to 15 for the serial code and 4 to 10 for parallel code, depending on convergence threshold. The simulations were performed on a $129 \times 65 \times 32$ homogeneous grid with uniform grid spacings in order to simulate a MESFET structure whose exact dimensions are shown in Fig. 6.

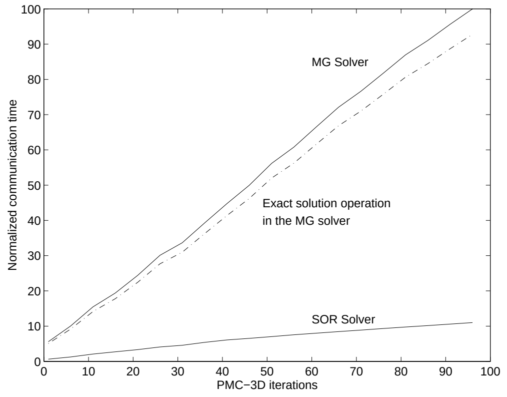

FIG. 8. *Normalized communication times for both SOR and MG solvers.*

The normalized communication overheads for both SOR and MG solvers can be seen in Fig. 8. The dashed line indicates that the communication overhead inflicted by the exact solution operator in the coarsest grid is the dominating part. This explains the speedup gap between the parallel and the serial versions of the multigrid solver, which can be alleviated by replacing the parallel SOR exact solution operator with either an analytical solver or a serial SOR solver. In the near future, in addition to this improvement, we plan to use ideas from [15] to extend the 3D multigrid solver further to handle non-uniform grid spacings.

6. Acknowledgement. This research is supported in part by the National Science Foundation under grant ECS–9312240. An earlier version of this paper was presented at *The Eight SIAM Conference on Parallel Processing for Scientific Computing*, in Minneapolis, Minnesota, March 14-17, 1997. The authors wold like to thank Dr. Marco Saraniti (Arizona State University) for helpful discussions in relation to this work.

REFERENCES

[1] C. JACOBONI AND P. LUGLI, *The Monte Carlo Method for Semiconductor Device Simulation*, Springer-Verlag, Vienna, Austria, 1989.

[2] K. KOMETER, G. ZANDLER, AND P. VOGL, *Lattice-gas cellular-automaton method for semiclassical transport in semiconductors*, Physics Reviews B, 46 (Jul. 1992), pp. 1382–1394.

[3] U. A. RANAWAKE, C. HUSTER, P. M. LENDERS, AND S. M. GOODNICK, *PMC-3D: A parallel three-dimensional Monte Carlo semiconductor device simulator*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 13-6 (1994), pp. 712–724.

[4] S. S. PENNATHUR AND S. M. GOODNICK, *Monte Carlo investigation of three-dimensional effects in sub-micron GaAs MESFETs*, Inst. Phys. Conf. Ser., No 141, Chapter 7, 1995.

[5] M. SARANITI, A. REIN, G. ZANDLER, P. VOGL, AND P. LUGLI, *An efficient multigrid Poisson solver for device simulations*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 15-2 (1996), pp. 141–150.

[6] A. BRANDT, *Rigorous quantitative analysis of multigrid, I: Constants coefficients two-level cycle with L_2-norm*, SIAM Journal on Numerical Analysis, 31-6 (1994), pp. 1695–1735.

[7] K. STÜBEN AND U. TROTTENBERG, *Multigrid methods: Fundamental algorithms, model problem analysis and applications*, in Multigrid Methods, Proceedings of the Conference, W. Hackbusch and U. Trottenberg, eds., Lecture Notes in Mathematics, Number: 960, Köln-Porz, November 23–27, 1981, Springer-Verlag, Berlin, pp. 1–176.

[8] A. BRANDT, *Guide to multigrid development*, in Multigrid Methods, Proceedings of the Conference, W. Hackbusch and U. Trottenberg, eds., Lecture Notes in Mathematics, Number: 960, Köln-Porz, November 23–27, 1981, Springer-Verlag, Berlin, pp. 220–312.

[9] W. HACKBUSCH, *Multi-Grid Methods and Applications*, Springer-Verlag, Berlin, 1985.

[10] J. KUO AND C. LEVY, *Two-color Fourier analysis of the multigrid method with red-black Gauss-Seidel smoothing*, Applied Mathematics and Computation, 20 (1989) pp. 69–87.

[11] I. YAVNEH, *Multigrid smoothing factors for red-black Gauss-Seidel relaxation applied to a class of elliptic operators*, SIAM Journal on Numerical Analysis, 32-4 (1995) pp. 1126–1138.

[12] A. BRANDT, *Multigrid solvers on parallel computers*, in Elliptic Problem Solvers, M. H. Schultz, ed., New York, Academic Press, 1981, pp. 39–84.

[13] O. A. MCBRYAN, P. O. FREDERICSON, J. LINDEN, A. SCHÜLLER, K. STÜBEN, C. A. THOLE, AND U. TROTTENBERG, *Multigrid methods on parallel computers - A survey of recent developments*, IMPACT of Computing in Science and Engineering, 3 (1991), pp. 1–75.

[14] L. R. MATHESON AND R. E. TARJAN, *A critical analysis of multigrid methods on massively parallel computers*, Technical Report CWI Tract 103, Center for Mathematics and Computer Science, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands, 1993.

[15] C. A. THOLE AND U. TROTTENBERG, *Basic smoothing procedures for the multigrid treatment of elliptic 3D operators*, Applied Mathematics and Computation, 19 (1986), pp. 333–345.