Algorithms for Drift-Diffusion Device Simulation Using Massively Parallel Processors

E. Tomacruz, J. Sanghavi, A. Sangiovanni-Vincentelli
Department of Electrical Engineering & Computer Sciences,
University of California, Berkeley 94720

Massively parallel processor (MPP) drift-diffusion device simulators have been presented in [1,2]. In both cases, most of the CPU time was spent in solving linear systems of equations (up to 95% reported in [1]). In this paper, we present methods for decreasing the total time for the iterative linear system solver by modifying the preconditioner for the iterative solver and by improving the initial guess for the Newton loop. In addition, we offer some general considerations regarding the parameters to use for the selection of a MPP architecture for device simulation.

The steady-state drift-diffusion equations and discretization of [1] are used in this study. Our contributions for improving the preconditioning algorithm for a full Newton outer loop with a conjugate gradient squared inner loop are:

1. Extension of the partitioned natural ordering of [1] to the MIMD MPP CM-5 architecture. This ordering compared favorably with other well known techniques such as the red-black ordering and the natural ordering when implemented on a SIMD CM-2. The CM-5 mesh structure is divided into rectangular blocks each called a subdomain. Each subdomain is mapped to a processor and the dimensions of the subdomain are powers of 2. The dimensions in each axis are equal or almost equal in order to form cubic subdomains. This minimizes the total surface area which in turn minimizes the data length of communications between processors. A simple row ordering is used to map the subdomains to the CM-5 processors since the fat tree connections allow minimal penalty for communications between arbitrary processors [3].

2. Evaluation of a three color ordering scheme and of nested dissection. These evaluations showed that while the three color scheme improved the quality of the preconditioner, the CPU time required did not make it competitive overall with the simpler red-black ordering. The nested dissection ordering, where the basic blocks of decomposition are ordered according to a red-black scheme, was also shown not to be effective. It can be deduced that better results are obtained when the discarded norm of the fill-ins is minimized instead of their number.

3. Introduction of a new ordering scheme tailored for the CM-5: the block partitioned natural ordering. To allow each processor of the CM-5 to execute in parallel, each subdomain mapped into each processor should be disconnected from other subdomains while doing forward and backward substitution. Using the idea of not having the same cut points for the forward and backward substitution proposed in [1], a new preconditioner called the block partitioned natural ordering preconditioner is proposed. This preconditioner still cuts the links at the boundary of subdomains for forward substitution. However, the cut planes for the backward substitution is moved by an offset of one which is illustrated by Figure 1. Natural ordering backward substitution is done consecutively from set 1 to set 4. Set 2 is composed of two planes of nodes, set 3 is composed of three lines of nodes, and set 4 has one node. Doing simple subdomain partitioning for backward substitution would have disconnected the set 4 node from its three neighbor subdomains by processing it first. This partitioning gave poor results. The offset of one allows information to travel from one subdomain to another during the preconditioning of the linear matrix.

4. Inclusion of fill-ins within each CM-5 subdomain. The architecture and larger memory of the CM-5 make it possible to accommodate several levels of fill-ins within each subdomain while doing incomplete LU decomposition, forward substitution, and backward substitution. Allowing fill-ins only in the incomplete LU decomposition did not improve the linear solver. A significant reduction in the total number of inner loop iterations is observed when fill-ins are also allowed in the forward and backward substitution process.

Results for a bipolar transistor described in [1] with V_{be}=0.8 and V_{ce}=1.0 are shown in Figures 3 and 4. PNO, NO, and BPNO signify partitioned natural ordering, natural ordering, and block partitioned natural ordering respectively. The number attached to BPNO indicates the fill-in levels allowed. 64 processors with no vector units are used for CM-5 simulations and 8k processors with floating point accelerators are used for CM-2 simulations. Fill-ins decreased the total inner loop iterations but not the CPU time. BPNO0 is two times faster than PNO-CM5 for the 32k mesh and produces the lowest CPU time for the CM-5. It is also more robust since PNO does not converge for the 64k mesh. PNO still gives the best performance for the CM-2.

The Newton algorithm is known to perform best when a good initial estimate of the solution is given. The best initial guess is usually obtained by a projection from two previous solutions whose bias conditions differ only at one contact to a new applied bias at that contact. The initial guess for the first two solutions may be obtained by the initialization described in [1]. This can be accelerated by a multigrid initial guess which does not require any specific knowledge of the device and the region of operation upon which it is simulated. It should be noted that voltage sweeps do not necessarily need to start with zero bias. Measuring threshold voltage or device breakdown for example only involves the simulation of a certain segment of the IV curves. Hence, the first two bias points may significantly affect the total CPU time of voltage segment simulations.

The scheme is based on two coarse grid mesh structures intertwined as shown in Figure 2. These coarse grids are constructed from 4 sets of discretization nodes. Coarse grid 1 is defined as the union of sets 1 and 2, and coarse grid 2 is defined as the union of sets 3 and 4. Set 1 is defined to be nodes with even coordinate values in all three grid axes, and set 3 is defined to be nodes with odd coordinate values in all three grid axes. Set 2 is defined to be the nodes connecting the nodes of set 1, and set 4 is the nodes connecting the nodes of set 3. If the solution of the equations on one of the coarse grids previously defined is carried out while the other coarse grid mesh structure is used as a boundary condition, the nodes in sets 2 and 4 will have only two active neighbors, thus making possible the static elimination of the variables associated with the nodes themselves. This ultimately allows the use of a smaller grid mesh structure composed solely of set 1 or 3. In addition, the elimination of set 2 or 4 decreases the number of variables which decreases the search space of the linear solver and hence, reduces the number of linear solver iterations.

We propose to use the multigrid discretization to define an approximate decoupled Newton scheme to produce a good initial guess for the full Newton algorithm. The algorithms consist of the following steps: First, do a simulation of set 1 nodes. Using set 1 as a boundary condition, an initial guess for set 2 nodes is computed by doing three extended simulations. Each extended simulation uses the actual fine grid discretization for one axis and the set 1 discretization for the remaining two axes. Using sets 1 and 2 as boundary conditions, an initial guess is calculated for sets 3 and 4. As a final step, sets 3 and 4 are used as boundary conditions to improve the initial guess for sets 1 and 2.

Simulations with a multigrid initial guess are done for a 31x31x15 mosfet, a 63x63x31 mosfet, and a 63x63x31 bipolar transistor described in [1] with varying bias conditions using a CM-2. A factor of at least 2 in CPU time improvement is observed compared to simulations with initial guess generated by [1]. It is also observed that as the bias conditions of the devices become harder to solve, the multigrid initial guess gives a better relative performance. Similar results are expected for the CM-5.

More memory for each CM-5 processor over each CM-2 processor produce a significant improvement in the convergence behavior since more coupling between grid points give better preconditioners. This decrease in inner loop iterations along with faster CM-5 processor elements give better performance relative to CM-2 type architectures as the problem size gets larger for the proposed algorithms and discretization technique. Several performance metrics for MPPs discussed in [4] are significant for the implemented CM-5 device simulator. The simulator has an 80 to 20 percent computation to communication ratio which is important in determining which MPP architecture to use and/or which MPP architectural and software aspects need to be improved. The CM-5 algorithm does not take advantage of overlapping communication and can be executed in SIMD mode. These two architectural considerations may become important when nonuniform grids are used. The bisection bandwidth is important since adjacent subdomains which are arbitrarily mapped to different processors need to communicate with each other. Latency time is unimportant because each communication call usually involves transferring hundreds of floating point numbers. Finally, a user specified mapping that takes advantage of certain regular properties in the algorithm and the architecture should give comparable if not better performance over MPP architectures that mimic shared-memory.

[1] D. Webber, E. Tomacruz, R. Guerrieri, T. Toyabe, A. Sangiovanni-Vincentelli, *IEEE Trans. on CAD,* Vol. 10, pp. 1201-1209, 1991.

[2] K. Wu, G. Chin, R. Dutton, *IEEE Trans. on CAD,* Vol. 10, pp. 1132-1140, 1991.

[3] Z. Bozkus, S. Ranka, G. Fox, *IEEE 4th Symposium on the Frontiers of Massively Parallel Computation,* pp. 100-107, 1992.

[4] G. Zorpette, *IEEE Spectrum,* pp. 28-33, September 1992.

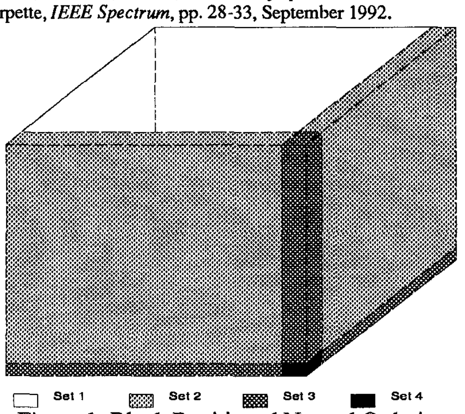

Figure 1: Block Partitioned Natural Ordering

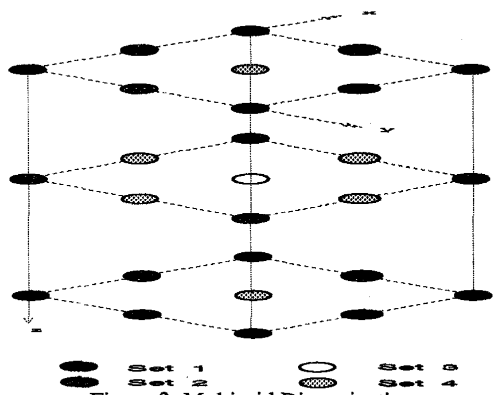

Figure 2: Multigrid Discretization

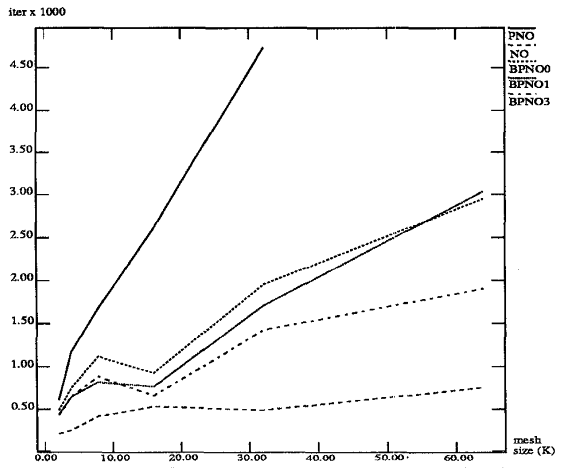

Figure 3: Inner Loop (iter) vs. Mesh Size (K)

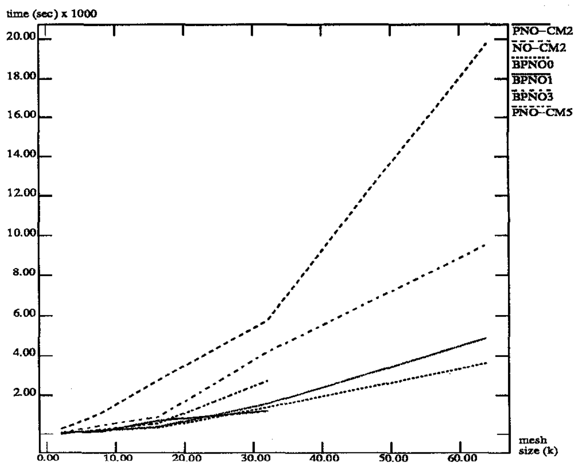

Figure 4: CPU Time (sec) vs. Mesh Size (K)