SIMULATION OF SEMICONDUCTOR DEVICES AND PROCESSES Vol. 4
Edited by W. Fichtner, D. Aemmer - Zurich (Switzerland) September 12-14, 1991 - Hartung-Gorre

Massively Parallel Computation for Three-Dimensional Monte Carlo Semiconductor Device Simulation

Henry Sheng, Roberto Guerrieri †
and Alberto Sangiovanni-Vincentelli

Department of Electrical Engineering and Computer Sciences
University of California, Berkeley, CA 94720, U.S.A.

† Dipartimento di Elettronica e Informatica
Università di Bologna, Italy

Abstract

This work presents a study of the applicability of a massively parallel computing paradigm to Monte Carlo techniques for device simulation. A unique mapping of Monte Carlo to SIMD fine-grained parallelism has been developed, decoupling the problem into separate computational domains. For MOSFET simulation, this novel mapping allows estimated speeds of over 200,000 scatterings processed per second on a 65,536 processor Connection Machine, nearly a factor of six over the fastest known to date.

1 Introduction

Three-dimensional Monte Carlo simulation is desirable as device technology is pushed into the deep submicron regime. However, its computational complexity is so high today that its practical use is out of the question. Attempts at coarse-grained parallelism and vectorization have been performed previously [1],[2],[3]. For the simulation of MOSFET's, speeds of 15,000-30,000 scattering events processed per second on a single processor CRAY-XMP are reported [1]. Though good for two-dimensional simulation, the computational speeds attained are not satisfactory due to the complexity of three-dimensional Monte Carlo.

We propose the use of massive data-level parallelism to solve this problem. Specifically, we present an approach, implemented on the Connection Machine[1], that significantly improves the CPU times required by three-dimensional Monte Carlo simulation on other architectures.

2 Computational Model

The Connection Machine [4] is a massively parallel SIMD computing system. The CM is based a *data directed* computational model. In this model, the processing units and the memory are associated together in one unit, rather than as separate entities. Thus, the potential bandwidth is increased significantly. A fully-configured system contains 65,536 processors. These processors can be allocated into *Virtual Processor sets*, VP sets, where multiple "virtual" processors can

[1]The Connection Machine is a registered trademark of Thinking Machines Corporation

VPR	Neighbor	Gen.	Gen., 2 VP sets
1	0.52 ms	3.17 ms	3.23 ms
4	1.34 ms	18.63 ms	18.75 ms
32	11.76 ms	70.99 ms	93.19 ms

Table 1: Communications time (32-bit integer) for different configurations, using 1024 processors: Near-neighbor, general, and general across VP sets. Randomized patterns are used for the general communications.

be mapped onto a single physical processor. This amounts to splitting the physical resources of a single processor among k virtual processors, thus obtaining a k-fold increase of the number of processors, (but each processor will be k times slower and memory will be reduced by a factor of k). Also, multiple VP sets can be implemented, where each VP set can be composed of a different number of virtual processors (the Virtual Processor Ratio, or VPR).

The CM communications architecture is configured in a hypercube network [10]. There are two mechanisms of data routing in the Connection Machine: generalized and near-neighbor. In generalized communications, the data is routed along the edges of the hypercube. For an n-dimensional cube, no two points are more than distance n apart (for a full CM, n=12). Conflicts in routing are resolved through the hardware *router* [4]. In near-neighbor communications, data is transferred between neighbors. Since all processors communicate in the same direction, there are no routing conflicts. Furthermore, the distance to the destination is minimum, not requiring any intermediate routing of data. In Table 1, a comparison among different communication relays is reported. Here, the overhead associated with communications across VP sets is small.

3 Description of the Simulator

Our simulator addresses the problem of static-field Monte Carlo. We have included the scattering mechanisms used in [1], while the electric field is provided by a 3D drift-diffusion simulator running on the CM [5].

3.1 Algorithmic Mapping

In a parallel computing environment, the execution time stems from two sources: computational cost and communications cost. In a massively parallel paradigm, the latter cost becomes significant, if not dominant. In fact, high data collision rates are produced from concurrent communications if the messages are not to physical neighbors. It is apparent that an efficient mapping, minimizing the communication complexity, is an important issue in finding an efficient parallel algorithm.

The difficulty in mapping Monte Carlo to massive parallelism lies in the nature of its computations. Unlike its drift-diffusion counterpart, the formulation of Monte Carlo inherently suggests multiple disjoint data representations.

The problem of charge transport suggests a particle-based representation to resolve the mechanics of flight for each simulated particle, where each particle is assigned to a processor. However, the calculation of the electric field suggests a spatially- based representation, where the device structure is physically discretized into a three-dimensional mesh structure, and each grid point is assigned to a different processor. Choosing one of these representations to solve one problem inherently forces the other one to be mapped in a very unnatural and inefficient way.

We solved this problem by exploiting the flexible nature of the CM communication network. The problem is decoupled into two disjoint domains - one particle-based and one spatially-based - allowing the use of the most natural configuration for each problem. Through the use of multiple VP sets, the parallel architecture can be structured in multiple configurations. The simulator can then dynamically switch between these configurations.

In this approach, interprocessor communications is minimized. In the "particle domain", each particle is associated with a virtual processor. Since individual flight calculations are independent of one another, there is no interprocessor communications. Here, the carrier flight characteristics are calculated. In the "spatial domain", the device is mapped onto a 3D grid, where only near-neighbor communications are required. Here, the particle locations are tracked, and the incident forces and the relevant Monte Carlo statistics are updated. Thus, the only overhead is in transferring data between domains, when the update of the incident field and of the statistical quantities is required. As presented earlier, the overhead of communications between the two domains is small, since this amounts to communications between VP sets (Table 1).

3.2 Computation of the flight time

The usual technique used to compute the flight time is based on an improved self-scattering approach [7]. While easy to implement, this technique suffers the problem of finding a reasonable definition of the self-scattering term, which should be physically correct and at the same time computationally efficient. In fact, an efficient definition of the self-scattering term requires a good knowledge of the highest energy reached by any particle at each location. However, since this is a result of the simulation, difficult trade-offs are required [11], [12]. An estimate of Γ, the scattering probability must be made. If this is too small, the flight must be repeated, increasing Γ each time, until it is large enough. Overestimation of Γ will result in excessively short flight times, leading to computational inefficiency.

In our simulator we have implemented two different approaches. The standard self-scattering approach has been implemented for sake of performance comparison with other existing codes. To guarantee the consistency of incident forces to the spatial position of the particle, a fictitious scattering event occurs when the flight time is sufficiently long as to allow a particle flight to proceed into the domain of another grid point. An alternative approach under investigation is the numerical solution of integral equation defining the flight time, avoiding the difficulties associated with self-scattering.

4 Computational Results

The Monte Carlo simulator has been implemented on a 1024 processor CM2 and its performances have been evaluated. From Figure 1, it is seen that the communications overhead scales with the number of VP's. This is because of the increase in collisions when routing data. The time-multiplexing of processors should reflect a linear tradeoff of VP's with iteration time (verified empirically in Figure 1). This can be modeled by:

$$Totaltime = m \cdot VPR + b \qquad (1)$$

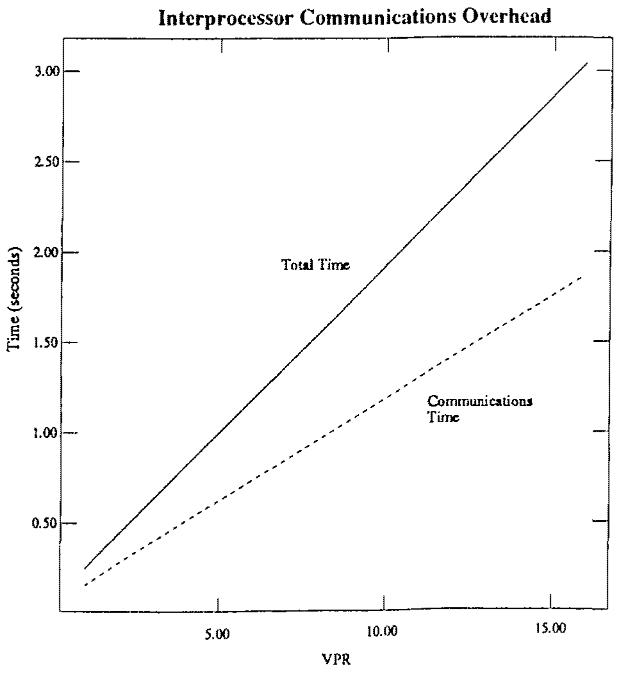

Figure 1: Total time and communications time per scattering iteration

where b is a constant having a value which can be determined fitting the experimental curves for a specific task. From this, the total scatterings per second is:

$$\frac{Scatterings}{second} = \frac{(VPR) \cdot (physicalprocessors)}{m \cdot (VPR) + b} \tag{2}$$

For $m \cdot (VPR) \ll b$, the scatterings per second is approximately linear with the VPR; while for $m \cdot (VPR) \gg b$, the scatterings per second becomes constant regardless of the VPR. This behavior is empirically seen in figure 2.

Theoretical calculations for maximum scattering processing rate for 512 and 1024 processors yielded 3048 and 5500 scatterings per second, correlating well with the empirical observations. It is seen that doubling of the number of processors does not double the speed. This is due to the increase in routing congestion (in communicating between domains) commensurate with the increased number of particles.

It is clear that a significant cost comes from the communications overhead. In some cases, this can be as high as 60% of the total cost. This is mainly due to the collisions incurred in the routing of data, particularly severe at the beginning of the simulation, when all the particles are in the vicinity of the ohmic contacts. However, as the simulation progresses, the particles spread out in the device, leading to faster routing. This behavior can be empirically observed (c.f. Figure 3). Within only a short amount of time, the communications cost decreased by 20-25%. Processing rates of 5,392 scatterings/second were achieved with a Connection Machine with 1024 processors. Using the theoretical model developed, taking into account the increase in routing traffic, a full Connection Machine with 65,536 processors will exhibit computation rates of over 200,000 scatterings/sec. Computational experiments are now under way to measure this rate on a full Connection Machine.

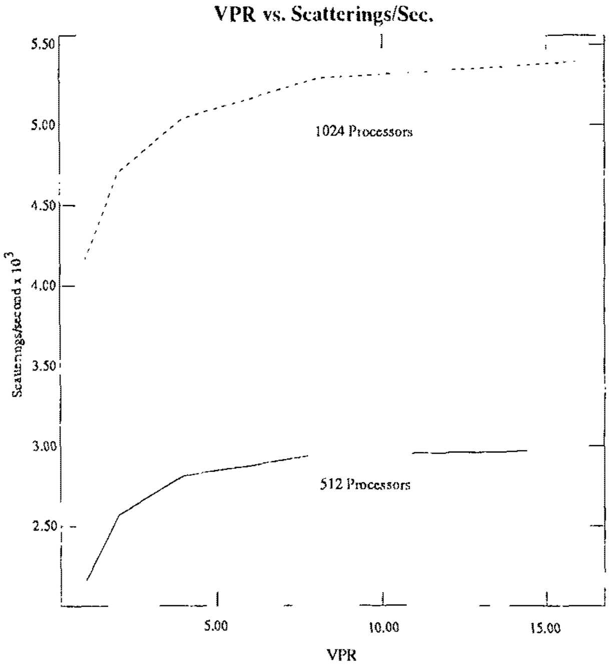

Figure 2: Scattering processing rate on the CM

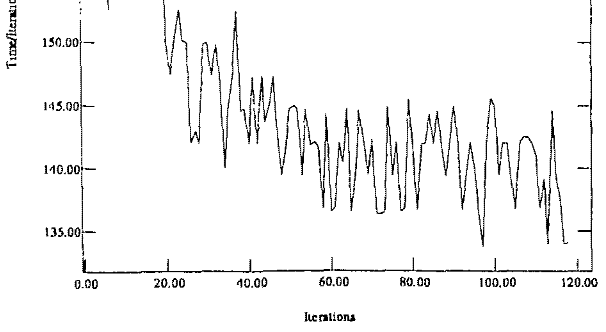

Figure 3: The cost of communications decreases as the simulation progresses.

5 Conclusions

A three-dimensional static-field Monte Carlo simulator has been developed to study the suitability of massive fine-grained parallelism for charge transport calculations. By exploiting the architectural flexibility of the CM, efficient mappings were constructed for each problem of relevance, allowing an estimate of a six-fold increase over scattering processing rates known to date.

An alternative approach for flight time computation, currently under investigation, is based on the numerical solution of the integral equation defining the flight time. This computation requires an additional evaluation of the scattering probability, but avoids the problem of the definition of the self-scattering factor. This approach is appealing on the CM since the additional computation is completely parallel.

References

[1] F. Venturi, et. al. *IEEE Transactions on Computer-Aided Design*, April, 1989.

[2] W. R. Martin and F. B. Brown. *The International Journal of Supercomputer Applications*, Volume 1, Number 2.

[3] D. Cheng, et. al. *IEEE Transactions on Computer-Aided Design*, September, 1988.

[4] D. Hillis. *The Connection Machine*, MIT Press, Cambridge, MA, 1985.

[5] D. Webber, E. Tomacruz, R. Guerrieri, T. Toyabe, A. Sangiovanni-Vincentelli, Proc. of NUPAD 1990.

[6] J. Hennessy and D. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan-Kaufmann, 1990.

[7] C. Jacoboni and P. Lugli. *The Monte Carlo Method for Semiconductor Device Simulation*. Springer-Verlag, 1989.

[8] D. Douglas, et. al. *The Architecture of the CM-2 Data Processor*, Thinking Machines Corporation Technical Report HA88-1.

[9] D. Hillis and G. Steele. *Communications of the ACM*, December, 1986.

[10] C. Stanfill. *Communications Architecture in the Connection Machine System*, Thinking Machines Corporation Technical Report HA87-3.

[11] E. Sangiorgi, B. Ricco, and F. Venturi, *IEEE Transactions on Computer-Aided Design*, February, 1988.

[12] C. Moglestue, *IEEE Transactions on Computer-Aided Design*, April, 1986.