Three-Dimensional Monte Carlo Device Simulation for Massively Parallel Architectures

Henry Sheng, Roberto Guerrieri [†]
and Alberto Sangiovanni-Vincentelli

Technical Report ERL-95-53
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

The applicability of a massively parallel processing (MPP) paradigm to the simulation of charge transport in semiconductor devices through the Monte Carlo method is investigated. A unique mapping of Monte Carlo simulation to a data-parallel software model has been developed in which the problem is decoupled into multiple computational domains, thereby increasing the locality of computation. The target machine for the Monte Carlo simulator is the Connection Machine CM-2, a massively parallel SIMD architecture. Enhancement of convergence rates for Monte Carlo estimators are achieved through the development of methods for efficient carrier flight time generation and importance sampling. The computational efficiencies of the algorithms are analyzed, and applications of the simulator are shown.

1 Introduction

The push for higher integration densities in semiconductor devices and the consequent reduction of minimum feature sizes have placed tremendous demands on the capabilities of simulators. These tools play a key role in the development of new designs, and accomplish three main goals: they streamline the development cycle by helping engineers guide the refinement of designs, they help to ensure the proper operation of a design once it has been finished, and they eliminate the need for costly experimental approaches for design. These goals are particularly relevant in semiconductor device design, where the investment in development effort and cost can be daunting. Worse yet, the required investment is steadily growing because of the significance of phenomena that become increasingly complex as device sizes approach the deep submicron regime. CAD tools for device simulation can help mitigate this complexity and considerably reduce the investment required for new devices. These tools become essential for the effective design of submicron devices.

Device simulation relies on the solution of the Boltzmann Transport Equation (BTE), coupled with Poisson's equation [1]. The Monte Carlo method [2, 3, 4] offers a statistical solution to the BTE and is one of the most accurate forms of simulation available. This method studies the aggregate system through the detailed motion of individual particles. The Monte Carlo method is often used to obtain accurate solutions of device behavior where macroscopic approaches such as drift-diffusion simulation are inadequate. The disparity in accuracy between these approaches becomes increasingly pronounced as device technology is pushed into the deep-submicron regime, where higher-order effects, such as non-local or ballistic transport behavior, are critical. Unfortunately, the Monte Carlo approach is plagued by exhaustive CPU requirements. In the absence of variance-reducing techniques, the convergence of the estimators is inversely proportional to the square root of the number of particles simulated [2]. For semiconductor device design, this may lead to inordinate run-times which are unacceptable for practical engineering use. For instance, results related to a self-consistent Monte Carlo simulation of a two-dimensional silicon MOSFET required several days of CPU time on an IBM 3090/600E with vector processing capabilities [5]. Other attempts at coarse-grained parallelism and vectorization [6, 7, 8, 9] have achieved promising speedups, but are focused on the two-dimensional Monte Carlo problem. Analysis in three dimensions is particularly relevant for the feature sizes of interest for Monte Carlo. Unfortunately, the extension of Monte Carlo simulation to three dimensions demands even greater computational requirements.

With such great demand for computational power, it is natural to turn to massively parallel architectures for solving simulation problems. The Monte Carlo method holds great promise for exploiting the raw computational power of these architectures, since it solves problems statistically through a large number of independent particle flights. In our previous work, we have proposed the use of massive data-level parallelism to solve the three-dimensional Monte Carlo problem in [10], on the Connection Machine CM-2. Alternative node-based implementations have been proposed in [11], where performances on a distributed memory nCUBE multicomputer and a shared memory Ardent Titan II graphics multiprocessor are investigated.

While the effectiveness of these node-based parallel Monte Carlo simulators rely on load balancing algorithms (which are in turn dependent on the target device), a data-parallel algorithmic mapping avoids the problems of task partitioning, since it is inherently load balanced. In this paper, we present our data-parallel approach, developed on the Connection Machine CM-2, for three-dimensional Monte Carlo simulation. In addition to the raw computational gains made through the use of massively parallel architectures, we present algorithms, which exploit data-parallelism, for the efficient statistical convergence of Monte Carlo estimators through methods for carrier flight time generation and importance sampling.

This paper is organized as follows: Section 2 describes the Monte Carlo method and its mapping to data parallelism; Section 3 discusses flight time algorithms for SIMD machines. in Section 4, variance reduction in rare regions is discussed; Section 5 contains computational results and applications of the simulator; and finally, Section 6 discusses some conclusions from this work.

2 Description of the Simulator

2.1 The Monte Carlo Model

Our simulator addresses the problem of single-particle, static-field Monte Carlo for electron transport [4, 12]. Formally, the solution to the stationary Boltzmann Transport Equation [1] is sought (equation 1), compliant with Poisson's Equation (equation 2):

$$0 = \frac{\partial f}{\partial t} = -\mathbf{v} \cdot \nabla_{\mathbf{r}} f - \frac{1}{\hbar}\mathbf{F} \cdot \nabla_{\mathbf{k}} f + \left.\frac{\partial f}{\partial t}\right|_{coll} \tag{1}$$

$$\nabla \cdot (\epsilon \nabla \varphi) = \rho \tag{2}$$

Here f, $\mathbf{r}$, and $\mathbf{k}$ are the distribution function, position vector, and wave vector, respectively, while φ is the potential and ρ is the net charge concentration. The last term of the BTE is the collision term, and is responsible for scattering and generation/recombination of carriers. The Monte Carlo solution for f is generated from the iterative simulation of a single charge carrier within the device, whose physics is governed by the BTE. An alternate method is the direct simulation of an entire carrier population which is synchronized in time, called the Ensemble Monte Carlo method [4]. In this work, we focus on the stationary problem and develop a single-particle parallel algorithm. With the single-particle method, the carrier is treated in a periodic manner, where it is reinjected into the device when it exits through an ohmic contact. By simulating a sufficient amount of time, enough sampling is performed on the carrier behavior so that reliable Monte Carlo estimates of the distribution function can be formed.

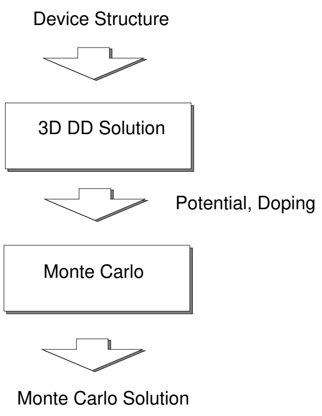

Figure 1: Monte Carlo as a post-processor to Drift-Diffusion

The carrier injections involved in the simulation are statistically independent of each other. Hence, by ergodicity, the simulation of n injections of a single particle into a device (with each injection taking place after the particle exits the device) is equivalent to the concurrent simulation of n particles which are injected simultaneously. Alternatively, this can be thought of as the collection of n independent single-particle Monte Carlo simulations.

The representation of the band structure is assumed to be a single spherical non-parabolic band, parameterized by α and β (we choose 0.5 and 1, respectively), which relates the energy E to wave vector $\mathbf{k}$:

$$E(\beta + \alpha E) = \frac{(\hbar \mathbf{k})^2}{2m^*} \tag{3}$$

In this representation, ellipsoidal behavior can be accounted for through the Herring-Vogt transformation [4]. We have included the scattering mechanisms used in [10, 13]: elastic acoustic phonon scattering [4], optical phonon emission and absorption with deformation potential interaction [4], Brooks and Herring [14] ionized impurity scattering, Tang and Hess impact ionization [15], and surface roughness scattering [16]. For impact ionization, the effects of secondary pair generation are neglected. In addition to computing flight mechanics using the exact analytical effective mass, the simulator can optionally update effective mass values only when an electron enters a control volume [13]. The simulator is used as a post-processor to an existing solution (figure 1), where the electric fields in the device are provided by a three-dimensional drift-diffusion simulator running on the Connection Machine CM-2 [17].

2.2 Algorithmic Mapping

2.2.1 Computational Model

The target machine for our simulator is the Connection Machine CM-2 [18], a massively parallel Single Instruction stream, Multiple Data stream (SIMD) [19] computer architecture. It can contain a maximum of 64K processors, and communication among these processors is accommodated through a hypercube network [20]. The processors can be allocated into *Virtual Processor* (VP) sets, where many "virtual" processors can be mapped onto a single physical processor. The ratio of virtual processors to physical processors is referred to as the Virtual Processor Ratio (VPR). The use of VP sets amounts to splitting the physical resources of a single processor among k virtual processors, thus obtaining approximately a k-fold increase of the number of processors, at a cost of a factor of k in speed and local memory space for each virtual processor. These VP's are connected in a topology, called the *virtual machine architecture.* This topology is consequently mapped onto the underlying physical machine. Multiple VP sets can be implemented, where each VP set can be composed of a different number of virtual processors and a different topology. This architecture employs a

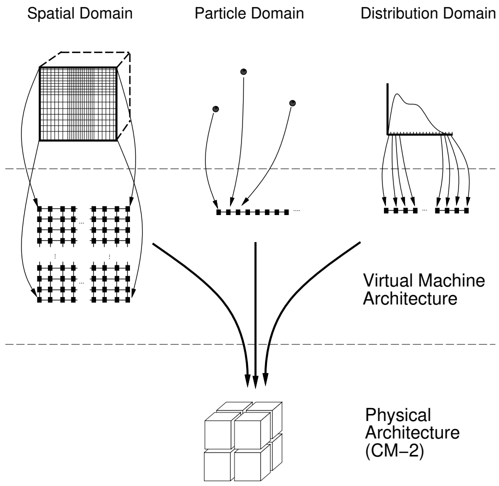

Figure 2: The computational domains are mapped to multiple VP sets, which in turn, are mapped to the physical machine architecture.

data-parallel software paradigm. In this paradigm, the parallelism comes from simultaneous operations across large sets of data, rather than from multiple threads of control [21].

2.2.2 Data Layout

Monte Carlo device simulation is difficult to map to data-parallelism because it inherently suggests multiple, disjoint data representations. Because of the tight relationship between algorithm and data layout in a data-parallel paradigm, the choice of data representation has a substantial impact on performance. 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 storage of the electric fields, doping concentrations, and spatially-dependent statistical information extracted from the simulation suggest 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. Storage of distribution functions generated from the simulation suggests yet another data layout, where the bins used in generating histograms are distributed in an array of processors.

In parallel machines, this multi-faceted nature of the data is typically handled through the partitioning of the spatial device structure among the processors [8, 11]. Each processor is then responsible for simulating the charge carriers which reside in its corresponding spatial partition. Unfortunately, the effectiveness of this spatial partitioning of tasks relies on load balancing among the processors, which is typically addressed through either dynamic load balancing algorithms or through reliance on the statistical nature of the run-times.

In this work, an alternative approach was taken which solves the data layout problem by exploiting the flexible nature of the CM-2 virtual machine architecture. The problem is decoupled into disjoint domains (particle-based, spatially-based, and distribution-based), allowing the use of the most natural configuration for each problem (figure 2). Through the use of multiple VP sets, the parallel architecture can be structured in multiple configurations. In each configuration, each processor is assigned to a fine-grained element of the corresponding domain, avoiding the need for load balancing algorithms. The simulator dynamically switches between these configurations to perform computations with the most appropriate data layout. This approach is appealing for a linearized Boltzmann formulation, where there are no direct carrier-carrier interactions, since the carrier flights are computationally independent.

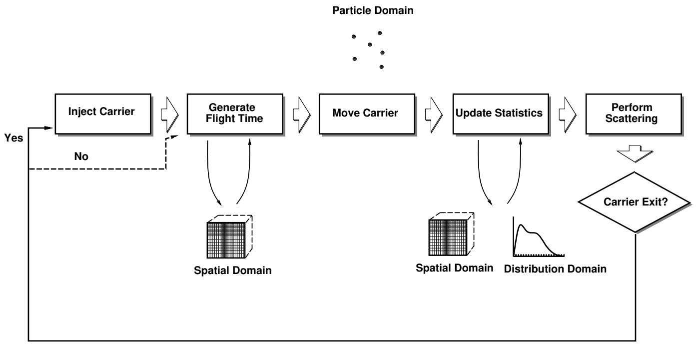

Figure 3: Main loop for the Monte Carlo algorithm

2.2.3 Parallel Algorithm

The Monte Carlo algorithm is composed of two parts. First, the preprocessing consists of constructing the VP domains, along with their associated data structures, and creating the parameters for flight time generation. The specific parameter generation algorithm is dependent on the choice of flight time algorithms, and is discussed in the next section. These parameters are dependent on the local device structure, since the carrier scattering probabilities are related to the local values, such as dopant densities. As a result, this step is performed with the architecture configured in the spatial domain. The second part of the algorithm is the main loop (figure 3), which processes the carrier flights. The main loop involves simulating the flight of particles and is therefore primarily performed with the parallel architecture configured in the particle domain, where each virtual processor is assigned a particle. The computational tasks for each particle are performed in instruction-level synchrony. These tasks consist of:

(i) Carrier injection: The particle is injected at the ohmic contacts according to a hemi-maxwellian distribution at 300^oK [22]. In this step, no interactions among processors are required, so that the injections are completely parallel.

(ii) Free Flight Generation: The methods used for the generating carrier free flight durations are discussed in Section 3. Depending on the algorithm used for flight time generation, this step requires from one to four single-precision floating point coefficients to be retrieved from the spatial domain, at the location corresponding to the current grid position of the carrier. In addition, the carrier flight processing in the subsequent algorithmic steps requires the impurity concentration, electric field data, and carrier concentration data, which are also communicated in this step of the algorithm. As a result, from 192 to 288 total bits per particle are communicated across VP domains. These communications references may be considered global distributed memory accesses, where the global data addresses are the spatial grid locations of the carriers.

These references may demand a substantial amount of communications [10]. The potential congestion is reduced through a single-celled software caching mechanism for storage and retrieval of spatial or distribution-related data. In this scheme, data is cached, or accumulated, in the particle domain (in the corresponding particle-based data structure) until the carrier enters a new control volume. For instance, the electric field is updated only when a carrier enters a new spatial cell. The use of software caching in this step of the simulation is effective. Simulations of gridded bulk Si material ($N_d = 10^{16} cm^{-3}$, $\mathcal{E} = 10kV/cm$) showed a 41% reduction of communications cost in this step.

(iii) Carrier State: In this step, the pre-scattering state of the particle $(\mathbf{r}, \mathbf{k})$ is updated after free flight. As in the first step, the carrier state update operation is completely parallel. This step involves the analytical integration of the carrier acceleration (to obtain wave vector) and velocity (to obtain position):

$$\mathbf{k}_{final} = \mathbf{k}_{initial} - \int_0^{t_f} \frac{q\mathcal{E}}{\hbar} dt \tag{4}$$

$$\mathbf{r}_{final} = \mathbf{r}_{initial} + \int_0^{t_f} \frac{\hbar \mathbf{k}(t)}{(1 + 2\alpha E(t))m^*} dt \tag{5}$$

Here, $\mathcal{E}$ is the electric field (under a piecewise constant discretization), $E(t)$ is the time-dependent carrier energy and t_f is the carrier flight time.

(iv) Statistics Update: Prior to carrier scattering, the particle state is sampled for updating the Monte Carlo statistics [4], and the residence time data is sent to the appropriate VP domain for statistics. This step reveals an advantage that parallel machines enjoy over vector architectures. While statistics update is not vectorizable [6], parallelization was achieved through the multiple domain mapping scheme. The accumulation of statistics is performed through a single communications step, in which the routing hardware performs concurrent computation operations that "reduce" the data, and sends it to the appropriate computational domain (spatial domain for spatially-related distributions such as electron concentrations, distribution domain for single-dimensional functions such as energy distributions). If multiple data packets have identical destinations, they are combined together during data collisions (for instance, when many carriers send residence time information to the same histogram bin for estimating the energy distribution). The cost of statistics computation is therefore equivalent to the communications cost of sending data across VP sets. Here, each VP in the particle domain sends a single 32 bit packet for each Monte Carlo statistic. Again, software caching is performed to reduce the cost of the communications. For instance, residence time information is accumulated until the particle enters a new spatial cell - at which time the totals are communicated to the corresponding spatial domain processor. For this step, software caching reduced the communications cost by 52% in gridded bulk Si simulations.

(v) Carrier Scattering: This step, which is completely parallel, randomly selects the scattering mechanism responsible for the carrier flight termination. This selection is made according to the distribution generated by the scattering models (according to the pre-scattering carrier state). The carrier wave vector, which may be impacted by scattering, is updated in this step.

After carrier scattering, the processors return to step (i). If a carrier does not exit the device through an ohmic contact in this free flight, the associated processor remains idle during the next iteration of step (i). Overall, for a bulk simulation of Si material, the software caching mechanism in steps (ii) and (iv) reduced the total communications cost by 43%, yielding a 5% overall improvement in simulator speed.

3 Computation of the Flight Time

In Monte Carlo, an essential part of simulating carrier flights is the generation of the free flight times - the amount of time, during which a semi-classically treated carrier will move, before it suffers a collision. Although the percentage of CPU time required by flight time routines may be as small as 5%, the impact on simulation performance is much greater, because of the introduction of artificial scattering events which restrict the free flight times of particles. The probability, $\mathcal{P}(t)$, that the free flight duration of a carrier is terminated during some dt around t is [4]:

$$\mathcal{P}(t)dt = P(t) \exp^{-\int_0^t P(t')dt'} dt \tag{6}$$

Stochastic free flight times may be generated through the inversion of the cumulative distribution function associated with equation 6 [4]:

$$\begin{aligned} 1 - r &= \int_0^t P(t') \exp^{-\int_0^{t'} P(t'')dt''} dt' && (7) \\ &= 1 - \exp^{-\int_0^t P(t'')dt''} && (8) \end{aligned}$$

$$\int_0^t P(t'')dt'' = -\ln r \tag{9}$$

Here, $P(t)$ is the instantaneous carrier scattering probability and r is a uniformly distributed random number on $(0, 1)$.

This integral equation must be inverted to find the time associated with the random quantity on the right hand side. However, the probabilities involved may require a great deal of computation. Since a large number of free flights may be processed during a simulation, speed is a critical issue. As a result, algorithms must be used to transform the problem into tractable forms which can be solved more easily [4, 23, 24]. These algorithms are typically based on the introduction of a fictitious "self-scattering" mechanism [24] whose probability is constructed so that integral inversion from equation 9 can be efficiently computed. Unfortunately, the algorithms that are commonly used may not be well-suited for data parallel machines.

In this work, a number of approaches have been implemented. The standard self-scattering approach, in which a stepwise constant function is used in place of $P(t)$, has been implemented for sake of performance comparison [10]. A generalization of this method, which expands the definition of the self-scattering function to include arbitrary families of functions, has also been developed and leads to more efficient flight time processing [25, 26]. Similar methods are proposed in [27, 28]. Another alternative approach is the analytical solution of an approximation to equation 9, avoiding the difficulties associated with generating "good" functions for self-scattering.

3.1 Γ-based Approaches

The usual techniques used to compute the flight time define self-scattering probabilities so that the total probabilities are constant or piecewise constant functions [4, 13, 23, 29, 30], Γ, which can be used in equation 9, under the condition that $\Gamma \geq P(E(t)), \forall E(t)$. In this manner, the inversion of the resulting integral can be computed efficiently as:

$$\int_0^t P(t')dt' = \int_0^t \Gamma dt' = -\ln r \tag{10}$$

$$t = -\frac{1}{\Gamma} \ln r \tag{11}$$

While easy to implement, these techniques suffer from the problem of finding a reasonable definition of the self-scattering term, which should be physically correct and at the same time computationally efficient. Furthermore, they either involve iterative methods (Iterative-Γ technique [23, 29, 31]), or many fictitious scattering events (Variable-Γ technique [13, 23]). In both cases, performance in SIMD machines is compromised because a large fraction of the processors may lie idle during conditional execution steps [25, 26] (up to 70% of the processors, in extreme cases, can be idle for approximately 38% of the main loop).

3.2 Algorithms for SIMD Architectures

The limitations of the Γ-based approaches suggest that other, more efficient, flight time algorithms be investigated for SIMD architectures. Processing efficiency can be used as a metric to evaluate the performance of these algorithms:

$$\begin{aligned} Processing\ Efficiency &= \frac{T_{simulated}}{T_{CPU}} \\ &= \frac{T_{simulated}}{CPU_{flight\ time} + CPU_{other}} \end{aligned} \tag{12}$$

Here, $T_{simulated}$ is the simulated time and T_{CPU} is the total CPU time, which can be decomposed into the CPU time for flight time generation, and the CPU time required for all other operations. The overall processing efficiency of the simulator can clearly be improved if $T_{simulated}$ is increased. This can be achieved through two approaches. First, by using massively parallel architectures, the number of carrier flights simulated can be increased. Second, each carrier flight can be made longer, subject to physical correctness. Traditional self-scattering algorithms artificially reduce the carrier flight times for the sake of computational efficiency. If longer carrier flights can be generated with little added cost in $CPU_{flight\ time}$, the overall processing efficiency can be improved.

One way longer carrier flight times may be generated is by reducing the effects of the piecewise application of a flight time technique. Flight time algorithms are typically applied in a piecewise manner to the energy (or **k**-space) domain of interest. This allows the development of total probability functions which more tightly bound the real scattering probabilities, leading to fewer self-scatterings and longer free-flight times. However, the piecewise nature of these algorithms requires a fictitious scattering mechanism which causes the termination of free flights during the transition of carriers across energy subregions. Though the self-scattering probabilities can be decreased with the use of more piecewise energy segments, a tradeoff exists with the number of fictitious scattering events induced by the boundaries between these energy segments. The number of energy-induced false scatterings has substantial implications in a massively parallel SIMD context because of the inherent need for instruction-level synchronization. In this paradigm, processors are latent during conditional execution stages if the condition is not satisfied, rather than immediate branching. This synchronous behavior implies that each false scattering, as with a true scattering, requires a full computation of carrier flight, update of statistics, and scattering angle. As a result, the effects of false scatterings are more pronounced in a massively parallel paradigm than in the sequential case, since many processors may lie idle during these operations. The flight time algorithms developed in this paper seek to create scattering functions which span larger energy regions with fewer segments than traditional variable-Γ schemes (to reduce energy-induced false scatterings), while reducing or entirely eliminating self-scatterings.

3.2.1 Generalized Self-Scattering

A generalized self-scattering scheme is proposed in [25, 26], where arbitrary probabilities $S_G(E)$ can be used instead of Γ, if $S_G(E)$ has the bounding property: $S_G(E) \geq P(E), \forall E$. Instead of only constant functions, the definition of $S_G(E)$ includes wider classes of functions, provided that the ability to analytically integrate and invert the resulting probability equation (equation 9) is preserved. This lends more flexibility to choose functions which form tighter bounds on the true probability (which leads to longer and more statistically significant carrier flights, as well as fewer self-scatterings) and whose piecewise segments span larger energy regions (which leads to fewer energy-induced false scatterings). A proper choice for $S_G(E)$ can be derived from the representation of the band structure, described by the $E - \mathbf{k}$ relationship in equation 3. From this equation, under the assumption of a piecewise constant spatial discretization of the electric field, we can define ζ:

$$\zeta^2(t) = \beta^2 + \frac{2\alpha}{m}(\hbar\mathbf{k} + \mathbf{F}t)^2 = (2\alpha E + \beta)^2 \tag{13}$$

With this definition of ζ we can define a particular bounding function, $\tilde{S_G}$:

$$\tilde{S_G}(\zeta(E)) = \tilde{S_G}(\zeta) = p_0 + p_1\zeta^2 \tag{14}$$

Here, the choice of p_0 and p_1 is subject to the bounding constraints that $\tilde{S_G}(E) \geq P(E)$, and that $\tilde{S_G}(E) - P(E)$ is small. The choice of $p_1 = 0$ recovers the case of the variable-Γ scheme. With the above definition of $\tilde{S_G}$, the bounding function defines a probability which is quadratic in time. As a result, the flight probability can be integrated and inverted analytically. Our particular choice of fitting constants results from linear regression, over each piecewise energy segment, on the analytical probability. After this regression, the constant term, p_0, is increased to ensure a non-negative self-scattering probability.

The inversion of the resulting flight integral involves the solution of a cubic equation. The amount of computation required for this step is substantially reduced by selecting p_0 and p_1 such that $\tilde{S_G}$ is non-negative for all energies (a sufficient, but not necessary, condition). In this manner, there is only one real solution to the integral inversion, thus avoiding the need to generate and choose from among three solutions.

3.2.2 Analytical Flight Time Integration

A different approach towards flight time generation is based on the direct solution of the flight integral (equation 9). The piecewise segments associated with this technique also span larger energy regions than the variable-Γ technique and result in substantially fewer energy-induced false scatterings. In addition, self-scattering is avoided entirely, in an effort to maximize the simulated carrier flight times while maintaining physical correctness. This approach avoids the need to iteratively develop functions for self-scattering probabilities. The results obtained for this algorithm required no interaction in defining scattering probabilities. The algorithm is composed of four main steps:

Energy/Time Transformation: The first step of the analytical flight time algorithm is the transformation between carrier energy and time domains. Probabilities for scattering are generally expressed as a function of energy or momentum. Equation 9 can be posed in terms of $P(E)$ rather than $P(t)$:

$$\int_0^t P(t')dt' = \int_{E_{initial}}^{E_{final}} P(E)\left(\frac{dt}{dE}\right)dE \tag{15}$$

Unfortunately, $P(E)$ remains quite complex, causing difficulties in analytically integrating the equation.

Surrogate Functions: An alternative for solving equation 15 directly is the solution of an approximation to the instantaneous scattering probability function. The $P(E)$ function is approximated with a suitable set of surrogate functions, as found through the piecewise application of linear regression. Because of the simpler form of the resulting functions, the integration can be performed analytically to find the corresponding primitive. For our examples, a four-dimensional basis was used ($\mathbf{G} \equiv \{\mathbf{1}, \sqrt{\mathbf{E}}, \mathbf{E}, \mathbf{E}^2\}$). The preprocessing parameter generation for this flight time algorithm consists of defining the bounds for the piecewise segments of S_A, and finding the corresponding coefficients of these basis functions.

Analytical Integration and Inversion: The generation of stochastic flight times is performed through the solution of equation 9, with the $P(E)$ term replaced by the appropriate surrogate functions, $S_A(E)$:

$$\Lambda = \int_{E_{initial}}^{E_{final}} P(E)\left(\frac{dt}{dE}\right)dE \tag{16}$$

$$\approx \int_{E_{initial}}^{E_{final}} S_A(E)\left(\frac{dt}{dE}\right)dE \tag{17}$$

We define a $F(E)$ as the primitive of the resulting integral:

$$F(E) = \int S_A(E)\left(\frac{dt}{dE}\right)dE \tag{18}$$

$$\Lambda \approx F(E_{final}) - F(E_{initial}) = -\log r \tag{19}$$

As a result, the generation of free flight is reduced to the solution for E_{final} in equation 19, followed by the subsequent energy to time transformation.

Turning Point Linearization: A fundamental assumption underlying the validity of the time/energy transformation is strict monotonicity. Carrier energy turning points violate this assumption and must be resolved numerically. Two ramifications result: First, the analytical integration must be performed in two steps. This leads to inefficiency in simulation and amounts to the introduction of a false scattering mechanism, which we call "linear boundary" scattering. Second, the $\frac{dt}{dE}$ term approaches infinity, leading to numerical problems in the integration.

These numerical problems are overcome through the introduction of an energy region in which the instantaneous scattering probability is assumed to be constant with time. At turning points, the instantaneous scattering probability changes very slowly with time and this approximation is accurate, given appropriate bounds for its use.

The definition of appropriate energy bounds for the constant probability approximation can be obtained from the first-order Taylor expansion of the scattering probability, P, about the carrier energy minimum, E_{min}:

$$P_0 = P(E_{min}) \tag{20}$$

$$P(E) \approx P_0 + \frac{dP}{dE}\Delta E \tag{21}$$

Here, P_0 is the instantaneous scattering probability of the carrier at E_{min}. The maximum tolerable percentage error in the instantaneous scattering probability, δ, is defined as the difference in probabilities between the minimum energy point and the boundary of the region, and can be approximated with equation 21:

$$\begin{aligned} \delta &= \left|\frac{\Delta P}{P_0}\right| \\ &= \left|\frac{P(E) - P_0}{P_0}\right| \\ &\approx \left|\frac{\frac{dP}{dE}\Delta E}{P_0}\right| \end{aligned} \tag{22}$$

This approximation can be inverted to find a bound on the size of the constant approximation region:

$$|\Delta E| \leq \left|\frac{\delta P_0}{\frac{dP}{dE}\Big|_{E_{min}}}\right| \tag{23}$$

where ΔE is the distance from the energy minimum to the boundary of the constant approximation region for the given tolerance. During free flight, a constant scattering probability is assumed when the carrier is within $\pm\Delta E$ of E_{min}.

4 Weighted Monte Carlo

The Monte Carlo method is primarily used to solve the BTE in areas where macroscopic methods are deficient - for instance, in the simulation of hot-carrier phenomena. Typically, these phenomena have significant effects on device behavior, but occur very rarely. As a result, direct simulation of the physical system will converge very slowly in these rare regions. Without variance-reducing techniques, Monte Carlo is not effective in estimating device behavior in the presence of rare events. This may lead to exorbitant convergence times. Particle multiplication is a common technique for reducing the variance of Monte Carlo estimates [32]. Unfortunately, this technique involves the simulation of a nondeterministic particle population. In a data-parallel context, this is difficult to achieve in a straightforward and efficient manner because of its requirement for the dynamic allocation of processors.

In recent years, a Weighted Monte Carlo technique [33, 34] has been developed which is based on importance sampling in classical Monte Carlo [2]. This approach is especially appealing in a parallelized or vectorized environment, since the method conserves particle populations. Weighted Monte Carlo allows the use of arbitrary probability distributions to provide the underlying physics.

The typical evaluation of crude Monte Carlo integration generates an estimator through collecting values, f, associated with discrete events which occur with probability p:

$$\begin{aligned} < f > &= \int_{-\infty}^{\infty} f(x)p(x)dx \\ &\approx \sum f(\hat{x})p(\hat{x}) \end{aligned} \tag{24}$$

A transformation is required to allow arbitrary probabilities, $p'(x)$, to control the physics of the simulation:

$$\begin{aligned} < f > &= \int_{-\infty}^{\infty} f(x)p(x)dx \\ &= \int_{-\infty}^{\infty} f(x)\left(\frac{p(x)}{p'(x)}\right)p'(x)dx \\ &\approx \sum f(\hat{x})\left(\frac{p(\hat{x})}{p'(\hat{x})}\right)p'(\hat{x}) \end{aligned} \tag{25}$$

This transformation is resolved for statistically through the use of the weighting function, $\frac{p(x)}{p'(x)}$. Consequently, the expectations of the integrations will be identical, leading to an unbiased estimate. With a proper choice of new probabilities, the variance can be reduced for the estimation of rare events. Particles can be "pushed" into regions where $p(x)$ is small. As a result, convergence of Monte Carlo estimators in these regions can be improved, because of the ability to observe more samples. In this paper, statistics for high-energy carriers are enhanced by using weights for preferential scattering directions (so that the carriers may gain more energy from the electric fields) and for dampening optical phonon emission probabilities. The distributions for carrier scattering angles are modified by increasing the probabilities for the preferred scattering directions. For instance, to push a carrier in the positive "y" direction, the probability for the final **k**-state may be modified according to:

$$P_{wmc}(\mathbf{k}(x,y,z)) = \tag{26}$$
$$\begin{cases} 0.85P_{crude}(\mathbf{k}(x,y,z)) & y < 0 \\ P_{crude}(\mathbf{k}(x,y,z)) + 0.15P_{crude}(\mathbf{k}(x,-y,z)) & y > 0 \end{cases}$$

where P_{wmc} and P_{crude} are the probability distributions for final carrier wave vectors using weighted and crude monte carlo techniques, respectively. The probabilities for optical phonon emission are modified by:

$$P'_{wmc} = \begin{cases} P'_{crude}\exp(-\frac{E}{\frac{1}{2}V_t}) & E \leq 0.45 \\ P'_{crude}\exp(-\frac{\frac{1}{100}(E-0.4)}{\frac{1}{2}V_t}) & E > 0.45 \end{cases} \tag{27}$$

Here, P'_{wmc} is the optical phonon emission probability using the Weighted Monte Carlo technique, and P'_{crude} is the unperturbed probability for optical phonon emission, for crude Monte Carlo. In this equation, $V_t = \frac{kT}{q}$, where k is Boltzmann's constant, T is the carrier temperature, and q is the electronic charge.

5 Computational Results

5.1 Scaling

An important computational statistic is the relation between virtual processor scaling (scaling the number of virtual processors while maintaining a constant number of physical processors) and processing speed, where the latter metric can be defined by the CPU time needed for all virtual processors to simulate one free flight iteration (figure 4). In the results for scaling, a constant conductivity effective mass was assumed during these free flights. With higher VP ratios, and hence, more particles in the simulation, the amount of congestion in the routing network increases. Overall, the time-multiplexing of processors yields a linear tradeoff between the VP ratio and processing speed:

$$Total\ Time = m \cdot VPR + b \tag{28}$$

Here, m is the slope of execution time to VP ratio, and b is an empirical constant. An interesting point is that doubling the VP ratio does not double the CPU time. This occurs because communications need not be performed if both the

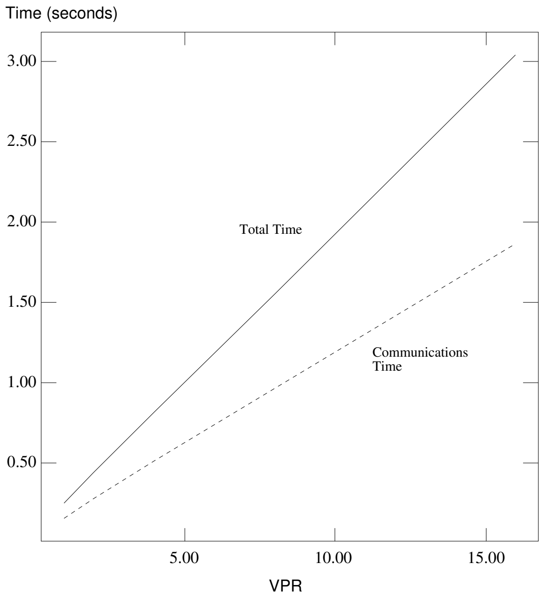

Figure 4: Total time and communications time per scattering iteration for a bulk simulation using variable-Γ.

source and destination virtual processors reside on the same physical processor. In addition, high VP ratios amortize the cost of instruction broadcast and decode, which need to be performed only once per physical processor.

From equation 28, the total number of scatterings processed per second is:

$$\begin{aligned} \frac{Scatterings}{second} &= \frac{Total\ Scatterings}{Total\ Time} \\ &= \frac{(VPR)\cdot(physical\ processors)}{m\cdot(VPR)+b} \end{aligned} \tag{29}$$

For $m \cdot (VPR) \ll b$, the scatterings processed per second is approximately linear with the VPR; while for $m \cdot (VPR) \gg b$, the scatterings processed per second exhibits asymptotic behavior, as seen in figure 5.

Theoretical calculations of maximum scattering processing rates for 512 and 1024 processors yielded 3048 and 5500 scatterings per second, correlating well with the actual performance. Clearly, doubling the number of physical processors does not double the speed. This is due to the increase in routing congestion (in communicating among domains) commensurate with the increased number of particles. These results indicate that the maximization of VP ratios, subject to memory constraints, will maximize the processing speed. The physical models used in this simulator for scattering mechanisms and flight kinematics were identical to that from a Cray-XMP vector implementation [6]. This included the assumption of constant conductivity effective mass during free flight, which simplifies the processing of the carrier kinematics. Processing rates of 5,392 scatterings/second were achieved with a Connection Machine with 1024 processors, while a larger Connection Machine with 32K processors achieved computation rates of over 200,000 scatterings/sec. This is a factor of between six to ten times vector performance, which was reported to be between 15,000 to 30,000 processed scattering events per CPU second for typical MOSFET applications.

Unfortunately, the constant effective mass approximation yields inaccuracies when used in conjunction with some Weighted Monte Carlo functions. The effective mass of carriers change as a function of energy, which, in the representation of the band structure from equation 3, results in:

$$m^* = \frac{\partial E}{\partial k} = m_0(\beta + 2\alpha E) \tag{30}$$

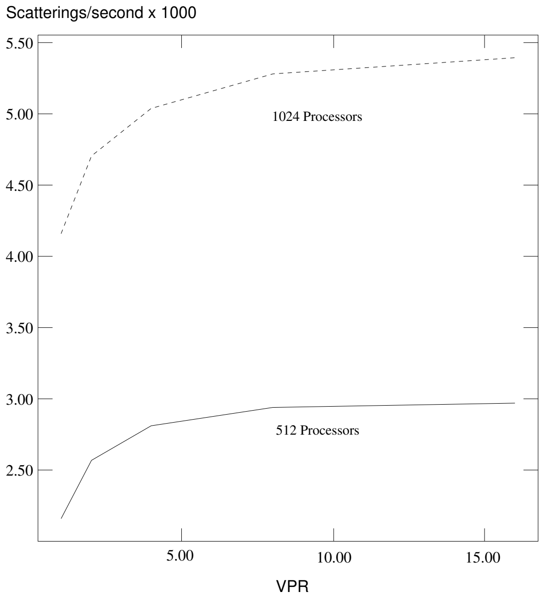

Figure 5: Scattering processing rates on the CM-2

where m_0 is the scalar electron mass in free space. With Weighted Monte Carlo, carriers may often gain a large amount of energy during the course of a free flight, as a result of abnormally long carrier flights or scatterings in preferred directions which may align the carrier flight path with the electric field. In these cases, the approximation of a constant electron effective mass fails, and the equations of motion must be integrated with the analytical conductivity effective mass from equation 30. The use of the analytic effective mass during flight leads to roughly a 2x increase in computation time for calculating the flight trajectories and detecting carrier exit from the current control volume.

5.2 Flight Time Algorithms

5.2.1 Gridless Simulation

A typical application of Monte Carlo is the simulation of gridless bulk silicon for the study of carrier behavior and extraction of macroscopic parameters, such as carrier mobilities. Processing efficiencies were therefore measured for simulations involving a gridless homogeneous slab with a dopant level $10^{16} cm^{-3}$ as a performance benchmark, running on a CM-2 with 1024 processors. Results from the generalized self-scattering and analytical flight time integration algorithms were compared with the results from three optimized choices of energy segment size for the variable-Γ technique (figure 6). Two energy patches were used for analytical flight time integration (with the boundary at the optical phonon threshhold, 0.052 eV), while four energy patches were required with generalized self-scattering (with boundaries at 0.052 eV, 0.175 eV, and 0.45 eV). The strategy from [13] was used for determining good choices for the energy segment sizes in the variable-Γ technique. The coarsest choice of segments among the three variable-Γ implementations (VG I, where the segments each span 0.029 eV) shows the best performance under high electric fields because of fewer energy-induced false scatterings, while the use of finer segments (VG II and VG III, where the segments each span 0.025 eV and 0.022 eV, respectively) yields better fits to the analytical probabilities and better performances at lower electric fields. Under high fields, the performances of all flight time techniques degrade because of both the short free flight times of high-energy carriers, as well as an increase in the number of energy-induced false scatterings.

Whereas optimality with the variable-Γ technique typically requires 15-20 segments [13], substantially fewer (2-5)

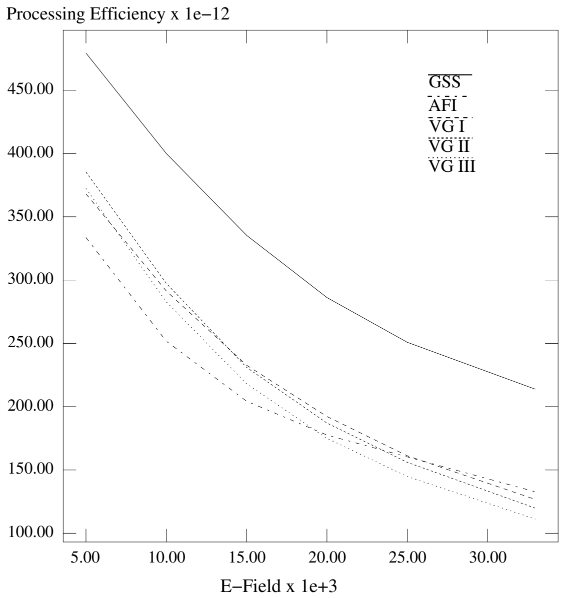

Figure 6: Performance of methods in gridless bulk silicon

segments are typically necessary for the generalized technique and the analytical flight time integration method, leading to fewer energy-induced false scatterings. The benefits from this are substantial, particularly in a SIMD paradigm, and are visible when scaling the applied field (figure 6). Here, the speedup of the generalized method improves from 37% for a 10 kV/cm field to a 69% gain in the case of a 33 kV/cm field. Likewise, while the performance of the analytical method is worse than the variable-Γ implementations for low fields, it yields a 5% improvement over variable-Γ for a 33kV/cm field. In the high field region, energy-induced false scatterings account for 58% of the carrier flights in the case of a variable-Γ approach, compared to 22% for the generalized method and 12% for the analytical integration approach.

5.2.2 Gridded Simulation

Although the simulation of a gridless device reveals the full improvement in efficiency obtainable through new algorithms for the study of bulk materials, the performance for gridded structures must be analyzed to estimate the computational speedup for more complex devices. The main benefit derived from the new flight time algorithms is longer carrier flight times, and hence, more statistical significance per simulated flight. In gridded structures, however, flight times are artificially truncated when carriers cross grid boundaries, offsetting the benefits of the new algorithms. The pathological case occurs when spatial false scattering becomes the dominant mechanism, resulting in little performance difference between the two self-scattering based flight time algorithms and degrading the performance of the analytical integration approach.

The same homogeneous device was simulated under 10 kV/cm, 20 kV/cm, and 33 kV/cm electric fields with different choices of grid spacings to determine the degree to which the grids limit the computational advantage of the new algorithms (the speedups of the generalized self-scattiring method over the variable-Γ algorithm is shown in figure 7, while the speedups of the analytical flight time integration algorithm over variable-Γ is shown in figure 8). For the generalized self-scattering method, the processing efficiency is more sensitive to grid spacings in the presence of low or moderate electric fields, because of longer average flight times. Long carrier flight times increase the probability of a grid boundary crossing, leading to more grid-induced false scatterings. Table 1 shows the frequency of the different fictitious scattering mechanisms. The speedup of the generalized algorithm for a 10 kV electric field falls from 37% for the gridless case to half of that value in the finely discretized case. Alternatively, with higher fields, average carrier energies are higher, and

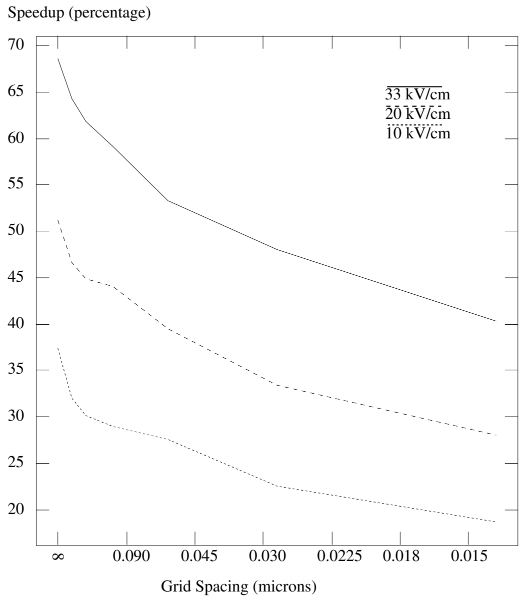

Figure 7: Effects of grid spacing on the speedup of Generalized Self-Scattering over optimized Variable-Γ.

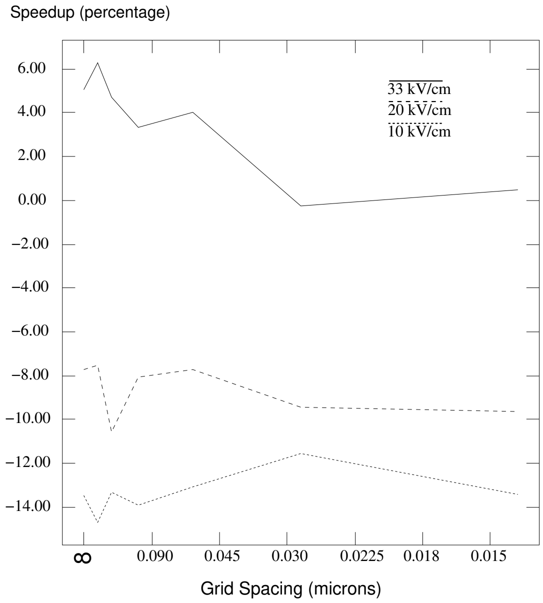

Figure 8: Effects of grid spacing on speedup of Analytical Flight Time Integration over optimized Variable-Γ.

Table 1: Fictitious scattering frequencies in a simulation of bulk silicon ($N_d = 10^{16} cm^{-3}$, E=10kV/cm) with varying grid size.

	Algorithm					
	Var-Γ		Gen. Γ		AFT	
Grid Size	0.25μ	0.01μ	0.25μ	0.01μ	0.25μ	0.01μ
Self	10.7%	6.3%	4.0%	1.9%	N/A	N/A
Energy	31.6%	15.2%	12.6%	5.2%	9.9%	4.5%
Grid	5.3%	48.8%	7.6%	57.5%	6.5%	53.1%
Linear	N/A	N/A	N/A	N/A	13.9%	6.5%
Total	47.6%	70.2%	24.2%	64.6%	30.3%	64.1%

Table 2: Processing Efficiency on Devices

Device	Var-Γ	Generalized	Analytical
$n^+nn^+ - A(1K)$	8.21e-11	9.35e-11	6.90e-11
$n^+nn^+ - B(1K)$	5.72e-11	6.45e-11	5.34e-11
$n^+nn^+ - A(16K)$	1.09e-9	1.23e-9	8.98e-10
$n^+nn^+ - A(32K)$	2.05e-9	2.42e-9	1.88e-9
nMOSFET (16K)	5.086e-10	5.566e-10	4.535e-10

average flight times are shorter. This leads to lower probabilities for grid-induced false scatterings. At high fields, the generalized scheme yields significant improvement over the variable-Γ approach, even for very fine grids. In the case of a 33 kV/cm applied field, the speedup falls from 69% for the gridless case to 40% in the finely discretized case. Hence, this regime is not dominated by grid-induced false scattering and is therefore still far from the pathological case mentioned previously. Typically, low field regions are discretized with coarse grids while high field regions are discretized with fine grids. The worst case performance is therefore rarely seen in practice.

The performance of the analytical flight time integration algorithm is competitive with that of the optimized variable-Γ algorithms. While it suffers a disadvantage in low-field regions where electron scattering probabilities do not change rapidly, this method enjoys a performance advantage in high-field regions (figure 8), where more statistics are generally needed - for instance, in the estimation of the tail of the energy distribution function. With this algorithm, the effects of grid spacings on simulator performance are smaller than with the generalized self-scattering approach. In Table 1, the frequency of the different types of fictitious scattering events is shown. For all flight time algorithms, the fraction of spatial grid scatterings increase as the grid spacing is reduced. In the case of generalized self-scattering, these spatial grid scatterings replace mostly real scatterings, so that the real scatterings are reduced by 40.39%. However, the analytical flight time and variable-Γ methods produce a large fraction of fictitous scatterings which are not related to the grid. For these algorithms, as the grid spacing is reduced, spatial grid scatterings replace significant fractions of both real and other fictitious scatterings. The effects of other fictitious scatterings therefore mitigate the effects of grid-size scaling on performance. Consequently, although the Generalized Self Scattering method yields the smallest fraction of false scatterings for a 0.25μ grid spacing (24.24%), the Analytical Method exhibits the smallest fraction for a 0.01μ spacing (64.07%).

5.2.3 Devices

Applications of the simulator to n^+nn^+ (figure 9) and n-MOSFET structures (figure 10) were performed to determine the speeds of the flight time algorithms in more complex devices (Table 2). The n^+nn^+ device is a psuedo-1D device

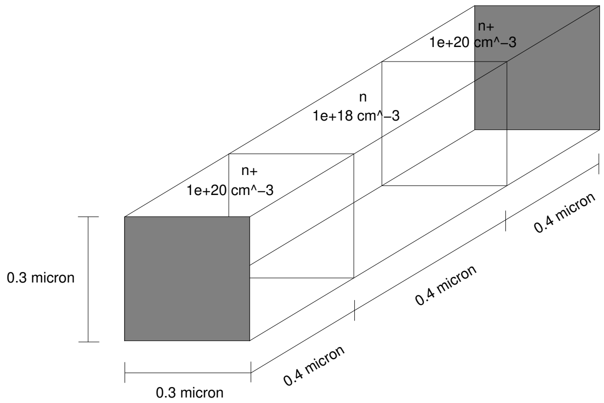

Figure 9: Device structure of the 1.2 micron, 1-D n^+-n-n^+ structure

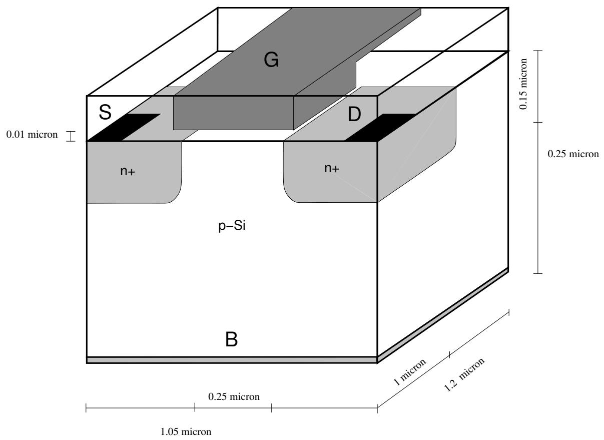

Figure 10: Device structure of the 3D MOSFET

Table 3: Simulator Routines

Max Time	Calculates maximum time spent in control volume by a particle
Scatter	Calculates scattering mechanism responsible for flight termination
Injection	Performs injection of particles into ohmic contacts
Commun./Stat	Time spent in communications and statistics collection
Flight Time	Calculates free flight time of particles
Kinematics	Performs kinematics calculations for particles

Table 4: Simulator execution profile on n^+nn^+ structure

Subroutines	Percentage CPU Time		
	Var-Γ	Gen. Γ	AFT
Max Time	20.98%	21.22%	16.36%
Scatter	8.91%	9.14%	8.52%
Injection	6.36%	6.66%	4.92%
Commun./Stats	20.06%	16.69%	12.39%
Flight Time	4.34%	6.84%	26.09%
Kinematics	39.35%	39.45%	31.72%

$(0.3\mu \times 0.3\mu \times 1.2\mu)$ doped at concentrations of $n = 10^{18}cm^{-3}$ and $n^+ = 10^{20}cm^{-3}$, where the lightly doped region is 0.5μ long. This device was simulated with both coarse (n^+nn^+-A, average grid spacing 0.0375μ) and fine grid discretizations (n^+nn^+-B, average grid spacing 0.009μ) to evaluate the effects of grids. A bias of 4V was applied across the device. In scaling the grid spacing, the analytical flight integration method was the least susceptible to change, (exhibiting a 22.6% change in processing efficiency for the n^+nn^+ device), while the generalized self-scattering was the most affected (a 31.02% change in processing efficiency for the same device).

The scalability of processing efficiency with machine size can be seen in the application of the simulator to the n^+nn^+ device for Connection Machines with 1K, 16K, and 32K processing elements in Table 2. The 32K machine yields speedups of 25.0, 25.9, and 27.2 times over the 1K machine for the variable-Γ, generalized, and analytical flight time methods respectively.

The simulator was applied to the submicron n-MOSFET structure in figure 10 with a machine size of 16K processors. The source and drain regions are doped at $N_d = 10^{20}cm^{-3}$, and the bulk region is doped at $N_a = 10^{16}cm^{-3}$. In this device, the source and drain regions extend 1μ into the device, in the z direction. The effective channel length is 0.25μ, where $\Delta L = 0.22\mu$. The biases applied to the device were $V_{gb} = V_{gs} = 3.0V$ and $V_{ds} = 3.5V$. On average, the performance results were roughly half of the n^+nn^+ device. The cause of the performance degradation may be best seen in the comparison of simulator profiles between n^+nn^+, and MOS simulations (in Tables 4 and 5, respectively). The tables show the differences in relative CPU costs among the routines in the main loop. First, the time required to calculate the maximum flight time before a carrier exits a spatial cell is larger in the case of this MOSFET. In these simulations, the exact analytical mass is used, so that the inversion of the motion equations to find maximum flight time requires an iterative method. In contrast to the one-dimensional nature of the n^+nn^+ device, the MOSFET is three-dimensional.

Table 5: Simulator profiles for MOS simulation

Subroutines	Percentage CPU Time		
	Var-Γ	Gen. Γ	AFT
Max Time	35.59%	44.33%	29.12%
Scatter	7.02%	6.59%	7.01%
Injection	4.87%	2.61%	4.07%
Commun./Stats	23.16%	13.54%	15.99%
Flight Time	3.51%	4.74%	22.16%
Kinematics	25.85%	28.19%	21.65%

With multi-dimensional fields, the iterative methods require longer convergence times.

6 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. The problems associated with the data layouts required by Monte Carlo were overcome by exploiting the architectural flexibility of the CM-2. Efficient mappings were constructed, allowing over a six-fold increase in scattering processing rates over vector supercomputing.

In addition, alternative approaches for flight time computation were investigated. These approaches seek to improve simulator performance through diminishing the effects of artificial scattering mechanisms. Generalized self-scattering is appealing because of the potential for increased performance with little additional effort. The conversion of variable-Γ methods to the generalized self-scattering scheme consists of the generation of coefficients for $\tilde{S_G}$ and the replacement of the integral inversion and probability integration code. Aside from the coefficient generation, which is performed once, the flight time code needs only to be changed from linear to cubic equations. The analytical flight time algorithm provides performance which is competitive with variable-Γ approaches while not requiring interaction to optimize performance. The results presented here were obtained without experimentation to determine optimal energy segment lengths and $S_A(E)$ values.

The particular flight time algorithm employed is dependent on the nature of the underlying scattering probabilities. From the results, the Generalized method yields substantial speedups over the other algorithms. However, when the true scattering probabilities are strongly irregular, a large number of self-scatterings may occur. In this case, the analytic flight time integration method may be more efficient. Also, with more complex scattering models, we expect the performance of the analytic method to improve relative to the other algorithms, because, unlike the other flight time methods, the CPU time for analytic flight time generation is not dominated by CPU_{other} in equation 12.

Finally, although variable population algorithms are problematic in a data-parallel paradigm, the problem of variance reduction was overcome through the use of Weighted Monte Carlo techniques.

References

[1] S. Selbeherr. *Analysis and Simulation of Semiconductor Devices.* Springer-Verlag, Wien, 1984.

[2] J.M. Hammersley and D.C. Handscomb. *Monte Carlo Methods.* Chapman and Hall, London, 1983.

[3] T. Kurosawa. Monte Carlo Calculation of Hot-Electron Problems. In *Proceedings of the International Conference on the Physics of Semiconductors*, pages 424–426, 1966.

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

[5] M. Fischetti and S. Laux. Monte Carlo Simulation of Submicron Si MOSFET. In *Proceedings of SISDEP '88*, 1988.

[6] F. Venturi, K. Smith, E. Sangiorgi, and B. Riccó. A General Purpose Device Simulator Coupling Poisson and Monte Carlo Transport with Applications to Deep Submicron MOSFET's. *IEEE Transactions on CAD*, Vol. 8, pages 360–369, April 1989.

[7] D. Cheng, C. Hwang, and R. Dutton. Pisces-MC: A Multiwindow, Multimethod 2-D Device Simulator. *IEEE Transactions on CAD*, Volume 7, pages 1017–1026, September 1988.

[8] S. Sugino, C. Yao, and R. Dutton. Parallelization of Monte Carlo Analysis on Hypercube Multiprocessors and on a Networked EWS System. In *Proceedings of SISDEP '91*, pages 275–284, 1991.

[9] A. Hiroki, S. Odanaka, and A. Goda. Massively Parallel Computation for Monte Carlo Device Simulation. In *Proceedings of VPAD '93*, pages 18–19, 1993.

[10] H. Sheng, R. Guerrieri, and A.L. Sangiovanni-Vincentelli. Massively Parallel Computation for Three-Dimensional Monte Carlo Device Simulation. In *Proceedings of SISDEP '91*, pages 285–290, 1991.

[11] U. Ranawake, C. Huster, P. Lenders, and S. Goodnick. PMC-3D: A Parallel Three-Dimensional Monte Carlo Semiconductor Device Simulator. *IEEE Transactions on CAD*, Vol. 13, pages 712–724, June 1994.

[12] P. Lugli. The Monte Carlo Method for Semiconductor Device and Process Modeling. *IEEE Transactions on CAD*, Vol. 9, pages 1164–1176, November 1990.

[13] E. Sangiorgi, B. Ricco, and F. Venturi. MOS2: An Efficient Monte Carlo Simulator for MOS Devices. *IEEE Transactions on CAD*, Vol. 7, pages 259–271, February 1988.

[14] H. Brooks and C. Herring. Scattering by Ionized Impurities in Semiconductors. *Phys. Rev.*, Vol. 83, page 879, 1951.

[15] J. Tang and K. Hess. Impact Ionization of Hot Electrons in Silicon (Steady State). *J. Appl. Physics*, Vol. 54, pages 5139–5144, September 1983.

[16] C. Hao, J. Zimmerman, M. Charef, R. Fauquembergue, and E. Consant. Monte Carlo Study of Two Dimensional Electron Gas Transport in Si-MOS Devices. *Solid State Electronics*, Vol. 28, pages 733–740, 1985.

[17] D. Webber, E. Tomacruz, R. Guerrieri, T. Toyabe, and A.L. Sangiovanni-Vincentelli. A Massively Parallel Algorithm for Three-Dimensional Device Simulation. *IEEE Transactions on CAD*, Vol. 10, pages 1201–1209, September 1991.

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

[19] M.J. Flynn. Very High Speed Computing Systems. *Proc. IEEE*, Vol. 54, pages 1901–1909, December 1966.

[20] C. Stanfill. Communications Architecture in the Connection Machine System. Technical report, Thinking Machines Corporation, 1987.

[21] D. Hillis and Jr. G. Steele. Data Parallel Algorithms. *Communications of the ACM*, Vol. 29, pages 1170–1183, December 1986.

[22] M. Lundstrom. *Fundamentals of Carrier Transport.* Addison Wesley, Reading, MA, 1990.

[23] R. Yorsten. Free-Flight Time Generation in the Monte Carlo Simulation of Carrier Transport in Semiconductors. *Journal of Computational Physics*, Vol. 64, pages 177–194, May 1986.

[24] H. Rees. Calculation of Distribution Functions by Exploiting the Stability of the Steady State. *J. Phys. Chem. Solids*, Vol. 30, pages 643–655, 1969.

[25] H. Sheng, R. Guerrieri, and A.L. Sangiovanni-Vincentelli. A Generalized Self-Scattering Technique for Monte Carlo Simulation Suitable for SIMD Architectures. In *NASECODE X*, pages 24–25, 1994.

[26] H. Sheng, R. Guerrieri, and A.L. Sangiovanni-Vincentelli. A Generalized Self-Scattering Technique for Monte Carlo Simulation Suitable for SIMD Architectures. *COMPEL*, Vol. 13, pages 661-669, December 1994.

[27] M. Cheng and E. Kunhardt. Electron Energy Distributions, Transport Parameters, and Rate Coefficients in GaAs. *J. Applied Physics*, Vol. 63, pages 2322–2330, April 1988.

[28] H. Kosina and S. Selberherr. A Hybrid Device Simulator that Combines Monte Carlo and Drift-Diffusion Analysis. *IEEE Transactions on CAD*, Vol. 13, pages 201–210, February 1994.

[29] R.W. Hockney and J.W. Eastwood. *Computer Simulation Using Particles.* McGraw Hill, New York, 1981.

[30] C. Moglestue. A Self-Consistent Monte Carlo Particle Model to Analyze Semiconductor Microcomponents of Any Geometry. *IEEE Transactions on CAD*, Vol. CAD-5, pages 326–345, April 1986.

[31] C. Jacoboni and L. Reggiani. The Monte Carlo Method for the Solution of Charge Transport in Semiconductors with Applications to Covalent Materials. *Reviews of Modern Physics*, Vol. 55, pages 645–705, July 1983.

[32] A. Phillips and P. Price. Monte Carlo Calculations of Hot-Electron Energy Tails. *Appl. Phys. Lett.*, Vol. 30, pages 528–530, May 1977.

[33] C. Jacoboni, P. Poli, and L. Rota. A New Monte Carlo Technique for the Solution of the Boltzmann Transport Equation. *Solid State Electronics*, Vol. 31, pages 523–526, March-April 1988.

[34] L. Rota, C. Jacoboni, and P. Poli. Weighted Ensemble Monte Carlo. *Solid State Electronics*, Vol. 32, pages 1417–1421, December 1989.