Available online at www.sciencedirect.com

ScienceDirect

Journal of Computational Physics 227 (2007) 1387–1410

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

Convective scheme solution of the Boltzmann transport equation for nanoscale semiconductor devices

D.A. Fixel [a,*], W.N.G. Hitchon [b]

[a] *Electrical and Microsystem Modeling, Sandia National Laboratories, P.O. Box 5800, MS 0316, Albuquerque, NM 87185-0316, USA*
[b] *Department of Electrical and Computer Engineering, University of Wisconsin-Madison, Madison, WI 53706, USA*

Received 5 April 2007; received in revised form 27 August 2007; accepted 6 September 2007
Available online 19 September 2007

Abstract

A model for the simulation of the electron energy distribution in nanoscale metal–oxide–semiconductor field-effect transistor (MOSFET) devices, using a kinetic simulation technique, is implemented. The convective scheme (CS), a method of characteristics, is an accurate method of solving the Boltzmann transport equation, a nonlinear integrodifferential equation, for the distribution of electrons in a MOSFET device. The method is used to find probabilities for use in an iterative scheme which iterates to find collision rates in cells. The CS is also a novel approach to 2D semiconductor device simulation. The CS has been extended to handle boundary conditions in 2D as well as to calculation of polygon overlap for polygons of more than three sides. Electron energy distributions in the channel of a MOSFET are presented.
© 2007 Elsevier Inc. All rights reserved.

Keywords: Boltzmann transport equation; Convective scheme; MOSFET; Semiconductor device simulation

1. Introduction

In a short-channel metal–oxide–semiconductor field-effect transistor (MOSFET), one has to contend with high electric fields at the drain end of the device, which can lead to hot carrier effects and device degradation. These arise due to several reasons, including the fact that power-supply voltages have not scaled proportionately because of compatibility issues with previous technologies, physical limits on the threshold voltage and noise margins, as well as manufacturability constraints. In order to explore hot carrier effects, models are needed to determine the energy distribution of carriers in the device.

In this paper, a kinetic transport model is implemented that is applicable to environments in which the mean free path is comparable to the device dimensions. In semiconductor devices, such as the MOSFET, as device dimensions are scaled down to the nanometer regime, the electron mean free path becomes comparable to the device dimensions. Characteristically, in conventional device modeling, simulations of carrier behavior are based on the drift–diffusion current equation, the continuity equation, and Poisson's equation.

[*] Corresponding author. Tel.: +1 505 284 3029; fax: +1 505 284 5451.
E-mail addresses: dafixel@sandia.gov (D.A. Fixel), hitchon@engr.wisc.edu (W.N.G. Hitchon).

In cases where the distribution function of particles is desired, the direct solution of the Boltzmann equation has been done by Monte Carlo techniques and a number of iterative schemes. In exploring the tail of the electron distribution, the particles that need to be observed represent a smaller portion of the total electron distribution, i.e., the rare occurrence. When we attempt to track the particular particles in question, and because the carrier mean free path becomes comparable to the device dimensions, the approach of employing the continuity equation is no longer appropriate. The fluid equation approach needs to be replaced by a kinetic description of the particles' behavior. Also, in researching hot carrier effects, one desires to obtain the distribution in energy of electrons, which requires a kinetic simulation technique. In this work we investigate the distribution of electrons using three velocity dimensions and two space dimensions. Of particular importance is the tail of the distribution and resolution of the same.

In order to find the electron distribution, it is necessary to solve the Boltzmann transport equation, which is given by [1]

$$\frac{\partial f}{\partial t}+\mathbf{v}\cdot\nabla f+\frac{\mathrm{d}\mathbf{k}}{\mathrm{d}t}\cdot\frac{\partial f}{\partial\mathbf{k}}=\left.\frac{\partial f}{\partial t}\right|_{\mathrm{sc}}, \tag{1}$$

where the second term on the left-hand side of the equation represents the rate of change of the distribution due to the particles' velocity, and the third term on the left-hand side represents the rate of change of the distribution due to the particles experiencing external forces. The term on the right-hand side of the equation is the collision term, i.e., the effect of collisions in altering the distribution function.

In later sections, we find the scattering rate in a phase space cell from a set of probabilities which are in turn found from the CS solution of the Boltzmann equation. This is essentially equivalent to finding a Green's function for a differential equation by considering the equation with a delta-function source. These probabilities are concerned with the manner in which particles move from one cell to another on the mesh and with subsequent scattering events of the particles. The scattering frequency in each cell depends on the kinetic energy of the particles; the scattering mechanisms used are explained in Appendix C. As the particles are scattered, they get mapped to the original mesh in accordance with the rules of polygon overlap. Calculation of polygon overlap is an integral part of the CS and it is important to accurately calculate it for accurate mapping of particles to the mesh. In this work, we have implemented the capability to calculate overlap between triangles and quadrilaterals and polygons of three or more sides. In most cases, overlap with polygons of more than five sides is not required, although the code was written to handle polygons of six sides. Details of these calculations can be found in Appendix A.

Many have undertaken the task of solving the Boltzmann Transport Equation. Abramo et al. performed an extensive study of various methods for solving the BTE, the focus of their study being the consequences that the various methods have on the high energy tail of the distribution. They noted that the tail controls such effects as impact ionization and electron injection from silicon into the oxide [2]. The Legendre polynomial technique has been around for some time, in which the distribution is expanded by a series of polynomials. The main limitation to this approach is the truncation of the series to a finite number of terms. Paige [3] published a numerical review on the subject in 1964, and Jorgensen and Meyer did an early transport study on silicon using the Legendre polynomial approach [4]. Higher orders of the polynomials as well as numerical discretization for the solution have been used by the Bologna group for more than twenty years (see, for example [5]).

Kurosawa reduced the Boltzmann equation to a Fokker–Planck equation in four-dimensional space (three spatial coordinates and one energy) [6], and Levinson equivalently demonstrated the same approach [7]. The approach regards the motion of an electron in energy space as a Brownian motion, and the ensemble of electrons is represented by a diffusion equation in energy space. Budd [8] used the path variable formulation or Chambers path integral approach to treat the Boltzmann transport equation. In this method, the Boltzmann transport equation is transformed to a coordinate system determined by the collision-free particle trajectories and an integral equation is formulated to obtain the energy distribution function. In order to find the hot carrier distribution function, one must obtain a solution to the spatially dependent nonlinear Boltzmann transport equation. The benefit of employing an iterative solution of the BTE is that it takes a nonlinear problem and breaks it down into manageable pieces. Beginning with the foundation laid by Budd, Rees [9] developed an iterative scheme in which the nth iterate of the distribution is substituted into the right-hand side of the transport equation and, subsequently, the $n + 1$th iterate is generated as the causal solution of the resulting inhomogeneous equation.

Monte Carlo simulations are one of the most widely used types of kinetic simulation. The particles are tracked by integrating the equations of motion over time. Kurosawa pioneered the Monte Carlo method of semiconductor device simulation [10]. The Modena group under Jacoboni [11] as well as Fawcett, Boardman, and Swain quickly implemented the new approach in their work [12]. The idea in many cases is to simulate the motion of one electron in momentum space as it undergoes a large number of scattering processes and to remember the time that the electron spends in each element of momentum space during its trajectory. The motion of the electron consists of a series of ballistic motions, each of which is followed by a scattering event. Although the approach began as a single particle approach, it was rapidly extended to the full many-body approach by the Modena group [13]. The approach was heavily used and expanded to include exchange by the Tempe group in the mid-1980s [14]. The single-particle full-band Monte Carlo approach was introduced by Shichijo and Hess [15]. Implementing a band structure using the empirical pseudopotential method, they further elucidated high-field transport and impact ionization in GaAs. Fischetti and Laux [16] undertook an extensive investigation into electron transport within small semiconductor devices using Monte Carlo analysis. Their technique improved upon the "state-of-the-art" treatment of high-energy carrier dynamics. The semiconductor is modeled beyond the effective-mass approximation by using the band structure obtained from empirical pseudopotential calculations. The electron–phonon, electron–impurity, and electron–electron scattering rates were computed in a way consistent with the full band structure of the solid, accounting for density of states and matrix-element effects more accurately than transport formulations had done prior to their study. The inclusion of the full band structure had the effect of reducing the amount of velocity overshoot via electron transfer to upper conduction valleys, particularly at large biases and low temperatures. In addition, Fischetti and Laux popularized the single-particle full-band Monte Carlo approach through the DAMOCLES code, which is currently being used around the world by several groups. Jakumeit and Ravaioli presented a local iterative Monte Carlo procedure, or LIMC, to reduce the computation time required for simulation of rare events in the device [17]. The method combines information from the local particle trajectories, obtained by use of an MC procedure, with an iterative method.

An approach that bears a striking resemblance to the approach used in this work is one that is termed the cellular-automaton (CA) method [18]. Transition probability matrices are used for both ballistic motion and cell-to-cell scattering. The cellular automaton is a discrete dynamical system which evolves in discrete time steps. It is defined at the nodes of a lattice, whereas the CS is defined inside each cell (triangle or quadrilateral) of the spatial mesh. For the CA approach, each site is characterized by a finite number of Boolean states which are either empty or filled. Transport is apparently more local (cell to neighboring cell) than in the CS, which is designed to use the largest spatial displacement possible, given the mean free path. There is a set of transition rules that are local in the sense that they only act on a given site and its nearest neighbors. In the CS, particles may transition more than one cell away from where they had their scatter in the previous timestep. Also, the CA method calculates the geometrical overlap in **k**-space, in which the normalized overlap factors play the role of effective scattering probabilities. In the CS, we use the calculation of polygon overlap in physical space in order to determine the spatial cell in which the particles being mapped have their next scatter. Saraniti and Goodnick [19] embellished the CA approach by extending it to a fullband Cellular Automaton/Monte Carlo approach. This hybrid algorithm helps to alleviate some memory problems of the CA while sustaining the speed and accuracy of the method. The Cellular Monte Carlo (CMC) approach was introduced as a faster alternative to the traditional Ensemble Monte Carlo (EMC) approach for the fullband simulation of charge transport in semiconductors. All possible transitions between cells of the discretized momentum space are precomputed and stored in lookup tables. In the case of cellular Monte Carlo (CMC), the coarseness of the discretization grid in the high energy region of the Brillouin zone results in numerical heating, or spurious diffusion of carriers toward the zone edge. Similarly, in the CS, numerical diffusion of carriers in the spatial cell can be an issue when solving the BTE on the mesh. This is an issue in the simulation since continuous quantities are represented discretely on a mesh, giving rise to an uncertainty in the exact location of the particles [20]. In either case the results could be improved by implementing a finer discretization, although one needs to trade off accuracy with computational expense.

In a case where one desires to accurately resolve the tail of the distribution (the electron distribution, for example), a more accurate method for solution of the Boltzmann transport equation is needed. One of the drawbacks of using a finite-difference scheme is summed up by the Courant–Friedrichs–Levy (CFL) criterion. It states that the fastest moving particles only travel a small fraction of the mesh spacing in a timestep [21]. In this work the electric field is unchanged throughout the simulation. By using the field from SGFramework, we are avoiding the issue with self-consistency (which slows down all simulations of charged particles in a self-consistent E-field) by setting the time step at a fraction of the dielectric relaxation time or the plasma period. This is discussed at length elsewhere including our own work where we have worked on strategies to deal with this [22,23]. Others have implemented the frozen-field approach, with the plausible assumption that the details of the distribution arising from nonlocal and hot-electron effects do not significantly influence the electric field [24]. It has previously been demonstrated that differences in electrostatic potential profiles as obtained from self-consistent and nonself-consistent Monte Carlo simulations are negligible. Jungemann et al. [25] obtained good agreement with experiment for substrate currents simulated using the nonself-consistent Monte Carlo method.

In Section 2 the convective scheme is described as well as an overview of the kinetic simulation based on the scattering rates, which is a specific implementation of the CS. Section 3 reports some results from the simulation, and Section 4 gives the conclusions.

2. Transport model at long mean free paths

This section commences with an introduction to the CS and then discusses the specific implementation in more detail. As a consequence of moving particles around the mesh, remapping rules are required, and these are presented next. Then the mesh structure and mesh locator are covered; lastly, the boundary conditions used in the simulation are explored. A point to be noted is that as the dimensions of interest approach the thermal deBroglie wavelength, we also approach the limits of validity of the semiclassical method of device simulation. At room temperature, the thermal deBroglie wavelength, λ_{DB} is equal to

$$\lambda_{\mathrm{DB}} = \left(\frac{2\pi\hbar^2}{mk_{\mathrm{B}}T}\right)^{1/2} = 8.4\ \mathrm{nm}, \tag{2}$$

where $\hbar$ is the reduced Planck's constant, m is the conductivity effective mass, and k_{B} is Boltzmann's constant. The 35-nm channel-length device is thus approaching this limit and the results should be viewed with the appropriate caution.

In this work the CS is used to find a set of probabilities for use in an iterative scheme which iterates in order to find collision rates in cells. The CS tracks a group of particles which have just had a collision in a given initial cell, in order to determine where they have their next collision. Depending on their velocity after the collision, they are divided into different groups. Each group is followed as they move "ballistically" and when they have their next collision, their location and energy are recorded. The fractions of the particles going to each final phase space cell then yield the probabilities needed by the iterative calculation of the rates.

There are two segments involved in moving the distribution of electrons. There is the ballistic move which is then followed by evaluation of the collision operator. During the ballistic move, the Boltzmann Transport Equation is reduced to the Vlasov equation, or

$$\frac{\mathrm{d}f}{\mathrm{d}t} = \frac{\partial f}{\partial t} + \mathbf{v}\cdot\nabla f + \frac{\mathrm{d}\mathbf{k}}{\mathrm{d}t}\cdot\frac{\partial f}{\partial \mathbf{k}} = 0. \tag{3}$$

The Vlasov equation is integrated along the characteristic curves, which are given by $\mathrm{d}\mathbf{x}/\mathrm{d}t = \mathbf{v}$ and $\hbar\mathrm{d}\mathbf{k}/\mathrm{d}t = \mathbf{F}$. Phase space cells, containing a density of electrons, follow the trajectory of the characteristic curves. This implements Liouville's theorem, which states that along the trajectory of any phase point the probability density in the neighborhood of the point remains constant in time [26]. Note that this approach is physically equivalent to the ensemble Monte Carlo method. Collisions are included at the end of a timestep. This method of characteristics, the CS, operates as follows [20]. The electron (or hole) distribution function $f(x, \mathbf{v})$ is advanced in time according to a propagator $p(x, \mathbf{v}, x'', \mathbf{v}'', \Delta t)$ such that

$$f(x,\mathbf{v},t+\Delta t) = \int f(x'',\mathbf{v},t)\cdot p(x,\mathbf{v},x'',\mathbf{v}'',\Delta t)\,\mathrm{d}x''\,\mathrm{d}\mathbf{v}''. \tag{4}$$

The propagator, or Green's function, $p(x, \mathbf{v}, x'', \mathbf{v}'', \Delta t)$, determines what fraction of particles move from cell $(x'', \mathbf{v}'')$ to cell $(x, \mathbf{v})$ in the time step Δt. The propagators allow Boltzmann's equation to be solved by successive convolutions of the "old" distribution function with the propagator.

In this work, the primary variable which is calculated, in each cell of the phase space mesh, is the scattering rate in the cell, denoted R. R is found from a set of probabilities, in an iterative calculation of the steady state values of R. The iterative scheme is outlined below. The CS is used to find the probability of moving from one cell where the particles had their last scatter to another cell where they have their next scatter. One reason that this is an accurate approach is that in this method each step taken by the particles corresponds to the distance between successive collisions. This step is large enough that the numerical diffusion in the simulation is reduced and the overall calculation proceeds faster. The advantage of this approach is most pronounced when the angular distribution after scattering is isotropic, which is the case here. Beginning with the scattering rates in the cells, and by moving the cells along their trajectories according to the CS, the simulation finds the steady state rate of scattering from each initial cell into each other cell of the mesh.

As mentioned previously, the version of the CS used in this work commences with the scattering rates in cells, moving the cells along the characteristic curves. The CS is actually used to find the scattering rate in cells, because the CS allows you to find the probability of scattering in any given cell, provided that the particle has just had a scatter in any other specified cell. That probability is used in an iterative scheme, which is described below in more detail. The way in which the method calculates the scattering rates is by finding several sets of probabilities. The probabilities indicate how many of the particles which scatter in a given cell of the mesh have their next scatter in each cell of the mesh [22]. Suppose the probability of scattering next in cell i having just scattered in cell i' is denoted $P^i_{i'}$. $R_{i'}$ is the rate of scattering in cell i'. A fraction of these scatters equal to $P^i_{i'}$ leads to scatters in i. Thus, in steady state the scattering rate $R_{i'}$ in cell i' contributes a rate of scattering equal to $P^i_{i'}R_{i'}$ in cell i. With other cells on the mesh contributing to this, and an additional source $S_{i'}$, due to generation of particles in cell i', the total scattering rate in cell i is

$$R_i = \sum_{i'} P^i_{i'}(R_{i'} + S_{i'}). \tag{5}$$

An example would be if 10^{13} particles per second scatter in cell 1 and if $P^2_1 = 0.01$, indicating that a fraction 0.01 of these scatter next in cell 2, then in steady state there are 10^{11} scatters per second in cell 2 contributed from cell 1. Fig. 1 demonstrates an example of calculating the scattering rates in a cell. In order to solve the Boltzmann transport equation, it is sufficient to find the collision rates R. Any other information needed derives from this quantity. The density in a cell $n(c)$, is

$$n(c) = \sum_E \frac{R(c,E)\lambda(c,E)}{v(E)\gamma(c)}, \tag{6}$$

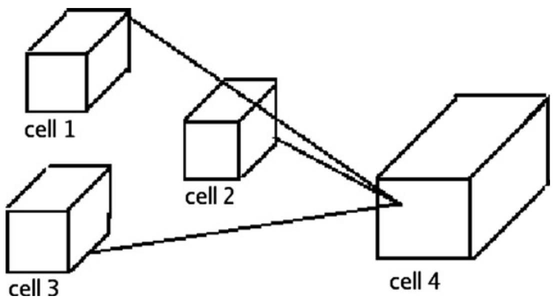

Fig. 1. Calculation of the total scattering rate in a cell. The scattering at rate R_1 in cell 1 contributes a scattering rate $P^4_1R_1$ in cell 4, as P^4_1 is the probability that a particle which just scattered in cell 1 will scatter next in cell 4. Similarly, $P^4_2R_2$ is the scattering rate contributed by cell 2 and $P^4_3R_3$ is the scattering rate contributed by cell 3.

where $n(c)$ is the density of the cell c on the spatial mesh, $R(c, E)$ is the collision rate of particles in cell c at energy E, $\lambda(c, E)$ is the mean free path, $v(E)$ is the magnitude of the velocity (or speed), and $\gamma(c)$ is the volume of cell c. The distribution f is found from R as follows:

$$f(c,E) = \frac{R(c,E)}{\nu(c,E)\gamma(c)}, \tag{7}$$

where $f(c, E)$ is the distribution in cell c at energy E and $\nu(c, E)$ is the scattering frequency in cell c at energy E. Since R is the scattering rate in each cell of the mesh, f is the distribution of particles which scattered in each cell. This distribution (and equivalently R) contain a complete specification of the full distribution. In order to find the full distribution in any cell, it is necessary to calculate the distribution of all the particles passing through that cell, which can be done using R. This method of obtaining the collision rate of particles has been implemented in previous work [27,28]. It is referred to as the transition probability matrix (TPM) method, wherein the CS finds the probabilities using the knowledge of where particles go as they move from the cell where they had their last collision. A more detailed overview of the TPM is given in the next section.

2.1. Overview of kinetic model

The CS updates the distribution each iteration by calculating successive scattering rates. Here we describe more specifically how the CS computes the probabilities of particles moving from one cell to a subsequent cell and the probabilities of moving from one energy to another due to collisions in that spatial cell. Using the TPM, or propagator, the subsequent iteration takes the distribution of scattered particles and advances it to the next cells where scattering takes place [27]. The quantity of interest is $R(c, E)$, the collision rate of particles in cell c at energy E. There are two portions to the TPM method. There is a ballistic, or collisionless portion, followed by the collision operator phase.

The first transition probability matrix computes the number of particles per second that collide in cell c at energy E'',

$$R(c,E'') = \sum_{c'} R(c',E')T_{\text{bal}}(c,E'':c',E'), \tag{8}$$

where $R(c, E'')$ is the number rate of particles that collide in cell c at energy E'', $R(c', E')$ is the number rate of particles that collided in cell c' and were redistributed with energy E' in the previous iteration. The way that the rate $R(c', E')$ is iterated on the mesh occurs as follows: consider a spatial cell c' and a group of electrons at energy E'. This group of electrons scatters at a rate $R(c', E')$ and has an angular distribution $f(\theta, \phi, c', E')$. $f(\theta, \phi, c', E')$ is the probability that an electron with energy E' that last scattered in cell c' is moving in a direction within the range ϕ to $\phi + \Delta\phi$ and θ to $\theta + \Delta\theta$. Also, $T_{\text{bal}}(c, E'' : c', E')$ is the probability that a particle having started in cell c' at energy E' will have its next collision in cell c at energy $E'' = E' - q\Delta\Phi$, where E' is the kinetic energy, $\Delta\Phi$ is the change in potential, and q is the magnitude of the charge on an electron. The sum is over all mesh cells c' at energy E'. In the current work the scattering is isotropic, so that the factor $f(\theta, \phi, c', E')$ is just proportional to solid angle. For this reason we do not stress that factor and write R as a function of E and c only. If it does depend on the angle, i.e., if the scattering becomes anisotropic, the method is the same but more storage is needed.

The second transition probability matrix redistributes the particles after a collision,

$$R(c,E) = \sum_{E''} R(c,E'')T_{\text{col}}(E:E''), \tag{9}$$

where $T_{\text{col}}(E : E'')$ is the probability that a particle, having previously collided in cell c at energy E'' will be redistributed with energy E within the same spatial cell c. Eq. (9) refers to particles which had energy E'' before collision and end up with energy E after the collision. They stay in the same spatial cell during the collision, so the spatial label c is unchanged. The rate of particles colliding in cell c with initial energy E'' is $R(c, E'')$. The fraction of these which have final energy E is $T_{\text{col}}(E : E'')$. Thus particles coming in with energy E'' contribute a rate of particles arriving at energy E in cell c of $R(c, E) = R(c, E'')T_{\text{col}}(E : E'')$. The total $R(c, E)$ must be found by summing over all initial energies E''. More details of the CS and the transition probability method are given in the references.

The motion in space from cell to cell described by Tbal following what might be referred to as "sacks" or blobs of phase space fluid that commence from and have the same shape as the physical space cell. A good way of explaining how the phase space fluid moves is the "water bag model" [29,30]. The water bag model brings out the fact that the distribution function is incompressible, or constant along a trajectory. Phase space is divided up into a number of these "water bags" within each of which the distribution function is also constant in time. The shape of the "water bag", identified by its boundary, changes in time as the bag moves. Even though the shape of the bag changes, the volume is fixed because of the constancy of the distribution and conservation of particles. These move and evolve over time, which leads to a need to know how these moving shapes overlap other cells at some time after they were created. This is the topic of the next section and is discussed in more detail.

2.2. Remapping particles

In this section we demonstrate how the particles are allocated to cells (in this work the scattering rate R of particles in each cell is allocated to the "final" cell) after each iteration of the TPM. In a given spatial location and energy, we launch Mc moving cells going in directions defined by the angular mesh in velocity space which has Mc points on that mesh. We follow these cells to see into which other cells of the mesh the particles will be scattered next. The propagator advances the particles in time, and at the end of a timestep a fraction of the particles are mapped back to the fixed mesh in accordance with the appropriate amount of overlap. The CS method uses phase space cells which move during the time step Δt. The faces of the cells move in accordance with the initial velocity and the local electric field. A certain fraction f of the particles in the moving cell collide, after the time step $\mathrm{d}t$, where $f = v\,\mathrm{d}t$ provided $f \ll 1$. This fraction of the particles which were in the moving cell at the start of the time step are put back in the overlapped cells, and with an appropriate energy so as to conserve energy.

Fig. 2 depicts the remapping of particles to the spatial mesh. The appropriate portion of particles is mapped to the fixed mesh based on the area of overlap between the moved cell and the mesh cell. For the sake of illustration, a uniform mesh is used to demonstrate how particles get mapped back. However, the mesh used in the simulation is an irregular mesh. Particles are mapped to the mesh cells A, B, C, and D in the figure according to the following rules, where N is the number of particles scattered out of the moving cell in the given timestep:

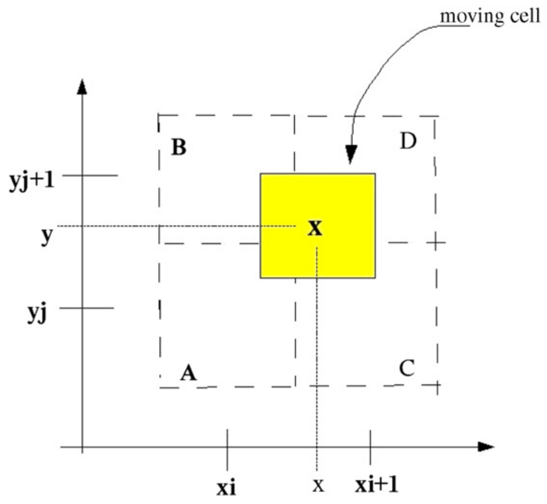

Fig. 2. Spatial remapping of particles after moving a cell. The fraction of the particles that get mapped to a spatial cell is determined by the area of overlap between the moving cell and the mesh cell. The moving cell passes through the fixed cells as shown, during multiple timesteps. Particles which are scattered out of the moving cell, for a single time step when the moving cell is in the position shown, are distributed as shown.

$$N_{\mathrm{A}} = N\left(1 - \frac{x - x_i}{\Delta x}\right)\left(1 - \frac{y - y_j}{\Delta y}\right), \tag{10}$$

$$N_{\mathrm{B}} = N\left(1 - \frac{x - x_i}{\Delta x}\right)\left(\frac{y - y_j}{\Delta y}\right), \tag{11}$$

$$N_{\mathrm{C}} = N\left(\frac{x - x_i}{\Delta x}\right)\left(1 - \frac{y - y_j}{\Delta y}\right), \tag{12}$$

$$N_{\mathrm{D}} = N\left(\frac{x - x_i}{\Delta x}\right)\left(\frac{y - y_j}{\Delta y}\right), \tag{13}$$

where Δx and Δy are the cell sizes in the x and y directions, respectively.

The mesh in velocity space is designed to reduce numerical diffusion. As mentioned previously, three dimensions in velocity are used to accurately resolve the collisions. The coordinate system consists of (v, μ, ϕ); v is the speed, μ is the cosine of the angle (θ) between the speed v and the positive z-axis, and ϕ is the angle in the xy-plane made by a counterclockwise sweep from the positive x-axis. Given a fixed cell k with electrons at energy E', the particles removed at the end of a timestep are mapped to the appropriate phase-space cell as follows: the moving cell is traveling in a direction within the range ϕ to $\phi + \Delta\phi$ and θ to $\theta + \Delta\theta$, where $\theta = 0$ is directed along the z-axis, and ϕ is measured in the xy-plane counterclockwise from the x-axis. The actual velocity of the particles in the cell will most likely lie at intermediate values of ϕ, θ, and v. In other words, the particles will be partitioned to the appropriate velocity cell based on fractions ϕ_1, ϕ_2, θ_1, θ_2, v_1 and v_2, which amounts to mapping to up to eight different velocity cells in one spatial cell for a given mapping.

In the next section, the mesh used in the simulation is discussed. This is a spatial mesh of a nanoscale MOSFET that is extracted from the 2D device simulation tool SGFramework [22]. The results of the 2D device simulation are used as input for the kinetic simulation.

2.3. Mesh structure and mesh locator

The system for the simulation is a nanoscale MOSFET. A two-dimensional (xy) mesh is used for the spatial cells, and three dimensions in velocity space are used to properly treat collisions. This section describes first the structure of the mesh used and the mesh locator used in the simulation of the CS. There is the spatial mesh from which moving cells are launched, and there is a base mesh which overlays the spatial mesh. The base mesh aids in determining which part of the spatial mesh a given cell has reached.

2.3.1. Spatial mesh

The spatial mesh used for the simulation is an irregular mesh extracted from the 2-D device simulation tool SGFramework [22]. The mesh consists of triangles and quadrilaterals of various sizes, and both the potential and electron concentration at each of the nodes is used in the initial guess for the CS. As mentioned previously, three dimensions in velocity are used to accurately resolve the collisions. In Fig. 3, only the device portion of the mesh is shown (the actual MOS transistor). This figure does not include the right and left boundary regions. The mesh locator is an “artificial” mesh, superimposed on the actual mesh. It is used to locate a specific region of the mesh with which to check for overlap with a moving cell. In order to implement the mesh locator, the simulation region is broken up into a grid by use of a base mesh.

2.3.2. Base mesh

The base mesh is a uniform rectangular mesh that is superimposed on the SGF mesh, allowing the mesh to be broken up into numerous regions. The simulation region was broken up into a 10×10 base grid. Each cell of the SGF mesh is associated with one or more of these regions of the base mesh. The point is that it is not efficient to check for overlap with every single cell of the fixed mesh. Then when a cell gets mapped back to the original mesh, it is only necessary to check for overlap with the cells associated with that region of the mesh locator. A linked list is associated with each region of the base mesh. The code traverses the list to ascertain which cells have an area of overlap with the moving cell. When a moving cell is determined to have overlap with one of the cells in the list of fixed cells, the particles get mapped back to that fixed cell at the appropriate

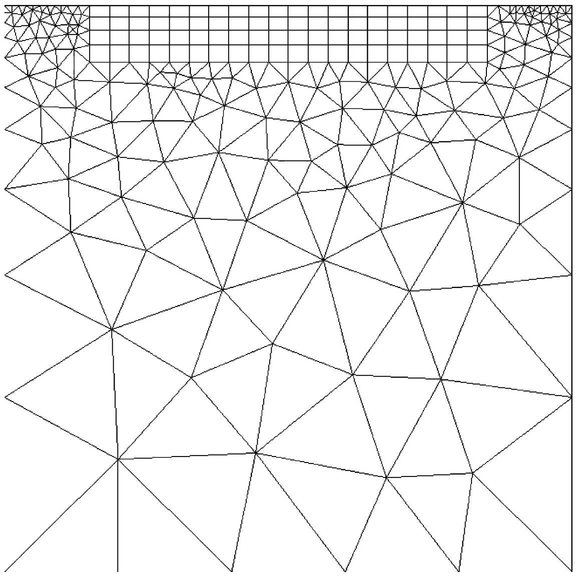

Fig. 3. Portion of mesh representing the simulation region.

values of velocity. In Fig. 4, an example of what one region of the base mesh entails is shown. The base mesh consists of 100 of these regions.

One of the more challenging aspects of this work is the handling of boundary conditions, i.e., reflecting boundaries, truncating boundaries, and corners. Special treatment is given to phase space cells that encounter one of these boundaries, and this is explored in the next section.

2.4. Boundary conditions

One of the challenges facing the implementation of the CS for the MOSFET device is the handling of boundary conditions. In previous work conducted for plasma simulation, complex boundaries were handled for an unstructured, triangular mesh [31] and for a mesh composed of small rectangular cells [32]. In this work,

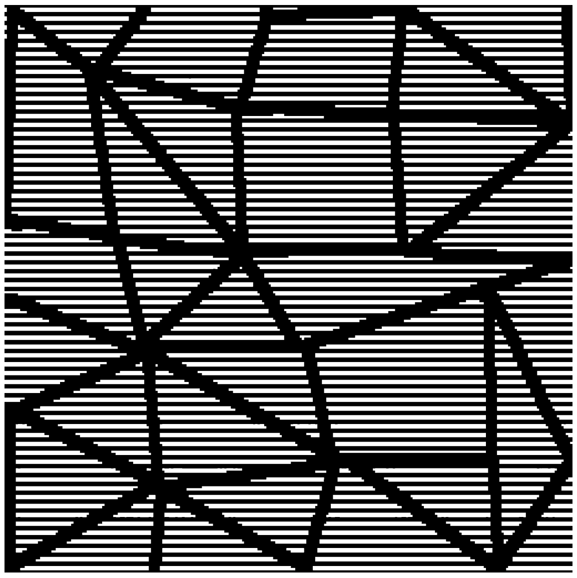

Fig. 4. Sample base mesh region.

an unstructured mesh composed of triangles and quadrilaterals was implemented with an external electric field applied to the semiconductor device. Additionally, the CS implemented in this work has been extended to handle boundary conditions in 2D. The mapping of cells of arbitrary triangular and quadrilateral shapes, onto other such shapes, is one of the major technical issues addressed here.

Specular reflection is used at the two horizontal boundaries. The horizontal boundaries represent the top of the device, or the silicon–silicon dioxide interface, and the bottom is inside the bulk of the device, below the channel. A cell approaching the boundary at an angle ϕ will rebound at an angle $2\pi - \phi$. Fig. B.1 is a simple depiction of a cell reflecting off the upper boundary. The two vertical boundaries allow cells to enter or leave the simulation region. The simulation region is the region in which particles are launched from and remapped to the original fixed cells of the mesh. The simulation region is bordered on the two vertical boundaries by a boundary layer, from which cells can enter or leave. For more specific details on the implementation of the boundary conditions, please see Appendix B.

2.5. Collisions

Types of scattering in the simulation include ionized impurity scattering, acoustic phonon scattering, optical phonon scattering (deformation potential scattering), and impact ionization. The collision frequencies used are energy-dependent. The collision frequency is used to calculate the timestep for a given energy shell. For each energy shell (or speed on the phase space mesh) there is a sheet of moving cells which is launched at the beginning of an iteration. Sheets are launched from the original mesh for different values of velocity and hence, energy. A sheet represents a cohesive unit, in which the corners (or nodes) of the sheet are moved with the same timestep for a given value of energy. The concept of the sheet was introduced in a previous implementation of the CS [31]. The timestep for a given energy shell is calculated based on the total scattering frequency for that particular case, according to

$$v\Delta t = 0.2. \tag{14}$$

In the case of ionized impurity scattering, there is a density dependence for the ionized impurity scattering frequency. In order to determine a single timestep for a given sheet, a constant density is assumed across the spatial mesh for purposes of calculating the timestep. The true density is used in the collision frequency implemented in the code, however. Details on the calculation of collision frequencies are given in Appendix C. In the upcoming section, results from the MOSFET simulations are presented.

3. Simulation results

In the MOSFET (metal-oxide-semiconductor-field-effect-transistor) a high electric field develops at the drain end of the device, especially in the case of short-channel MOSFETs. In this case some of the carriers gain sufficient energy to overcome the silicon–silicon dioxide barrier and make it into the gate dielectric, creating gate leakage current as well as interface states at the silicon–silicon dioxide interface. The gate leakage current is generally greatest when the drain bias approximately equals the gate bias [33] and leads to an undesirable increase in power consumption, since the gate current is characteristically undesirable. There is an application to EPROM (erasable programmable read only memories) and flash EEPROM (electrically erasable programmable read only memories) devices in which the carriers injected into the gate are used as the writing mechanism, but even then one wants to control the amount of carriers entering the gate [34]. This is the reason for the high-stress condition for the potentials applied to the gate and drain of the MOSFET.

In the case of an NMOS transistor, there is a positive potential applied on the gate and drain, with the source grounded. Two theories address the value of energy that an electron can attain. One is called the lucky-electron model, and this theory assumes that the maximum energy an electron can acquire as it drifts along the channel is limited to the maximum potential difference along the surface (this includes the source–drain bias plus the barrier height between the source and inversion layer) [35]. The second theory, called the quasi-thermal-equilibrium approach, says that the electron energy distribution function is determined from the electric field and does not depend directly on the external voltage applied [36]. In this theory, if the electric field is sufficiently high in the channel, a certain fraction of electrons will be able to initiate impact ionization or to surmount the oxide barrier regardless of the applied potential.

In the transport of cells, the quantity of discretization on the mesh is the velocity. The velocity as a function of energy is calculated from the expression for kinetic energy, or

$$E = \frac{1}{2}mv^2. \tag{15}$$

The mean-free path is calculated according to

$$\lambda(E) = \frac{v(E)}{\nu(E)}. \tag{16}$$

Table 1 shows values of the electron mean free path at the values of applied voltage used in this work. The first column indicates the applied potential, the second column gives the electron velocity for the corresponding energy, and the third column gives the mean free path. The mean free path is an important parameter in that it represents the mean distance between successive scattering events. The higher the energy, the shorter the mean free path. Electrons at higher energy are more likely to undergo a collision, and thus are not quite as ‘‘lucky’’ as electrons at lower energy.

Each plot exhibits two peaks. One is a peak at zero energy which is due to electrons at room temperature. The peak at low energy corresponds to carriers that are in equilibrium with the lattice. The second peak is due to the hot electrons and has a peak slightly below the value of energy that corresponds to the potential experienced by the electrons. For example, if there is a potential of 0.9 V applied between source and drain, then an electron that traverses the entire channel can acquire up to 0.9 eV of energy. An electron that travels halfway from source to drain would be able to acquire roughly 0.45 eV. Beyond this is the thermal tail; the slope of the tail is proportional to $1/T_L$, where T_L corresponds to the lattice temperature [37]. The tail can be attributed to electron–phonon interactions. In the event that electron–electron interactions would be included in the simulation, these too would contribute to the tail of the distribution.

We discuss first results from simulation of a 35-nm channel length transistor and second results from a 70-nm channel length transistor. Note that the length of the transistor device refers to the length in the x-direction of the simulation region (the portion of the mesh over which the CS is performed). The 35-nm channel length transistor has a device length of 50 nm, and the 70-nm channel length transistor has a device length of 100 nm in the simulation region. The electron energy distribution is presented. This is found for a given spatial cell on the mesh by integrating over all values of θ and ϕ for a given value of energy on the velocity mesh. The upcoming section presents results from a 35-nm channel length transistor device, and the following section presents results from a channel 70-nm length transistor device. A MOSFET is depicted in Fig. 5. The source and drain regions are formed with n-type material, and the substrate, or bulk is formed with p-type material.

3.1. 35-nm channel length transistor

The MOSFET transistor used is an NMOS transistor (p-type substrate with n-type source and drain regions). The channel length is 35 nm and the device length, including source and drain regions, is 50 nm. The substrate doping concentration is $8.0 \times 10^{17}/\text{cm}^3$ and the source and drain doping concentration is $8.0 \times 10^{19}/\text{cm}^3$. The minority carriers in the device are electrons, and during device operation the strong inversion in the channel of the device gives rise to a high density of electrons. A potential of 0.9 V was applied on the gate and drain of the MOSFET, and the potential, electron and hole densities were obtained from SGFramework and used as input for the kinetic simulation. The reason for applying 0.9 V on the gate and

Table 1
Electron mean free paths

Potential (V)	$v(E)$ (m/s)	Mean free path (nm)
0.9	1.103×10^6	10.2
1.5	1.425×10^6	8.2

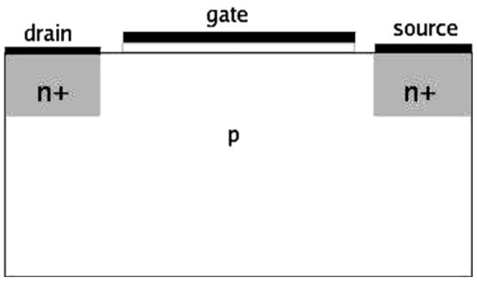

Fig. 5. NMOS field-effect transistor used in the device simulations.

drain is that device reliability experiments are often performed under these conditions in order to "stress" the transistor device and to observe hot carrier effects [33].

The total scattering frequency as a function of energy for an energy of 0.9 eV is 1.08×10^{14}/s. This value is the sum of the acoustic phonon scattering (2.024×10^{13}/s), intervalley phonon scattering (emission, 7.88×10^{13}/s), intervalley phonon scattering (absorption, 7.46×10^{12}/s), and ionized impurity scattering (1.35×10^{12}/s) frequencies. Fig. 6 is a plot of the electron energy distribution in the same cell. Again, this is the case with 0.9 V applied to the gate and the drain. The distribution is found from the scattering rate and is calculated by dividing the rate by the scattering frequency in the phase space cell and by the volume in the phase space cell. The scattering frequency is also an energy-dependent parameter. The spike on the tail in this case appears to be caused by a local voltage difference of about 0.7–0.8 V which occurs close to the channel. Such a difference would allow electrons to accelerate to the cell where the distribution is shown and acquire about 0.7 V. That flux would be balanced, in equilibrium, by a diffusive flux in the opposite direction. Fig. 7 is a plot of the electron energy distribution in the cell located approximately halfway between source and drain, but in this case, 0.6 V is applied to the gate and 0.9 V is applied to the drain for comparison to the previous case. As shown in the plots, the numbers in the tail of the distribution are greater in the case of 0.9 V applied on the gate contact than they are for the case of 0.6 V applied on the gate. Ang et al. [42] showed experimentally that the worst-case stress conditions occur at $V_g \simeq V_d$. We see in the results that as the gate voltage becomes equal to the drain voltage, the amounts of electrons in the tail increase, which in turn implies an increase in the hot-carrier damage that can occur. Ang et al. [42] found conclusive experimental evidence that the worst-case stress condition is caused by the increased injection of the "high-energy tail" electrons into the gate oxide under such conditions. To see the effect of varying the gate voltage while holding the drain voltage constant at 0.9 V, consider Fig. 8. The mesh for this case has 922 nodes and 1111 elements. Runs take

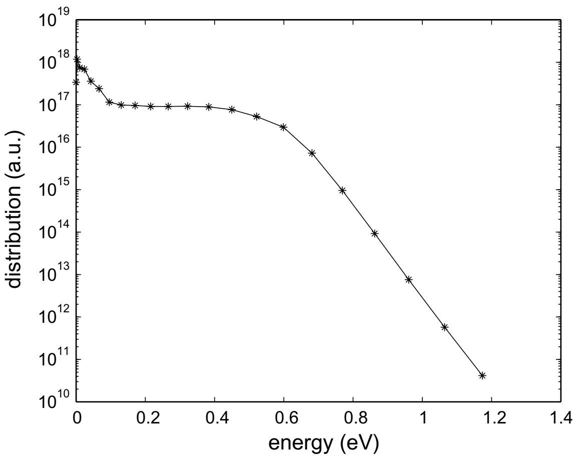

Fig. 6. Distribution of electrons in cell in MOSFET channel for 35-nm device with 0.9 V on gate and drain. The cell is approximately at the midpoint between source and drain but slightly closer to the drain.

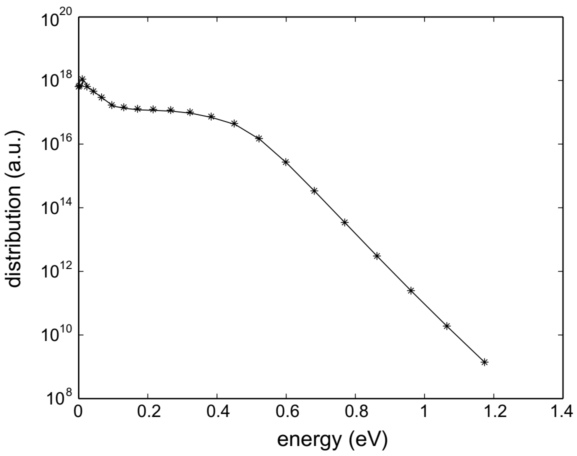

Fig. 7. Distribution of electrons in cell in channel for 35-nm channel length device with 0.6 V on the gate and 0.9 V on the drain contact. The cell is approximately at the midpoint between source and drain but slightly closer to the drain.

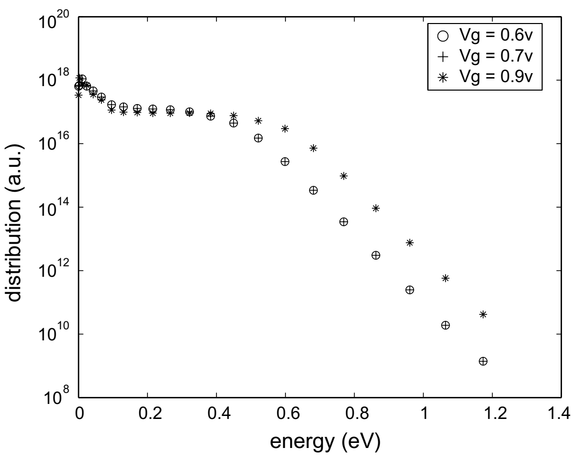

Fig. 8. Electron energy distribution in cell in channel for 35-nm channel length device with 0.9 V applied on the drain. The gate voltage is varied from 0.6 V to 0.9 V. Hot carrier damage increases as the gate voltage approaches the drain voltage.

roughly 15 iterations to reach convergence, and an iteration takes approximately 4.5 h of actual computer time on a Solaris workstation. Having seen some examples of the 35-nm channel length device, we now proceed to discuss results from simulation of the 70-nm length device.

3.2. 70-nm channel length device

In this section, results from the device with a length of 100 nm are presented. The channel length in this case is roughly 70 nm. The scattering frequency for an energy of 1.5 eV is 1.73×10^{14}/s. The total scattering frequency is the sum of the acoustic phonon scattering (2.613×10^{13}/s), intervalley phonon scattering (emission, 1.11×10^{14}/s), intervalley phonon scattering (absorption, 1.029×10^{13}/s), and ionized impurity scattering (2.55×10^{13}/s) frequencies for this value of energy. Fig. 9 shows the electron energy distribution in a cell near the center of the channel region with 1.5 V applied between the source and drain. Fig. 10 shows the electron energy distribution within the same cell but with 1.5 V applied to the gate and 1.0 V applied to the drain. Note that in this case, the numbers in the tail of the distribution are greater than those of the case with 1.5 V applied on both the gate and the drain. To see the effect of varying the drain voltage while holding the gate voltage

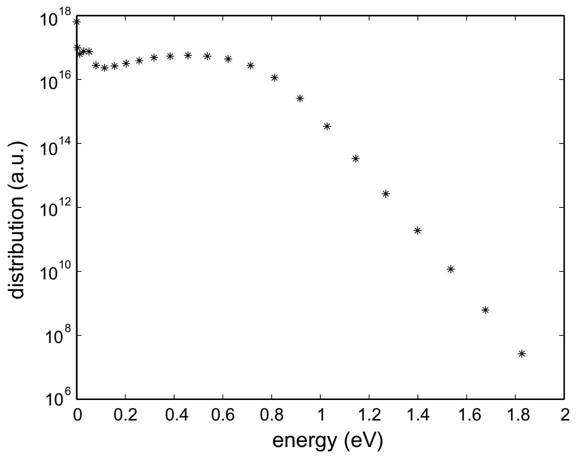

Fig. 9. Electron energy distribution in cell in channel for 70-nm length device with 1.5 V applied on the gate and drain. The cell is located in the channel approximately halfway between source and drain, slightly closer to the source. The potential in the cell is 1.39 V.

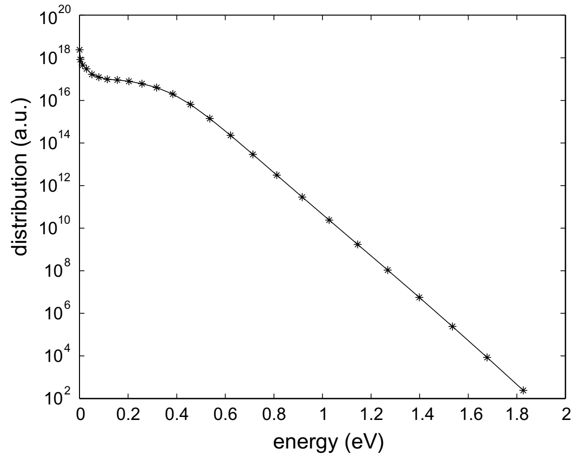

Fig. 10. Electron energy distribution in cell in channel for 70-nm channel length device with 1.5 V applied on the gate and 1.0 V applied on the drain. The cell is located in the channel approximately halfway between source and drain, slightly closer to the source. The potential in the cell is 1.26 V.

constant at 1.5 V, consider Fig. 11. The density of electrons in the tail, and hence the likelihood of hot-carrier damage, increases with the drain bias. While the potential on the gate confines the electrons near the interface, the potential on the drain provides a greater electric field to the electrons as they traverse the channel. The mesh used for the 70-nm device has 884 nodes and 1094 elements. A run converges by 15 iterations, and one iteration takes 5.5 h of actual computer running time on a Solaris workstation. Overall, the TPM has proven to be an accurate and efficient implementation of the CS in order to be able to extend the method to 2D semiconductor device simulation and the handling of 2D boundary conditions.

3.3. Summary

In the results we have seen in this section, the trend is to exhibit a bimodal distribution if the mean free path is comparable to the device dimension L. There is a peak near zero energy which corresponds to the electrons that are in thermal equilibrium with the lattice. There is a second peak at higher energy for those electrons which have gained a significant amount of energy from the field and have lost relatively little to collisions.

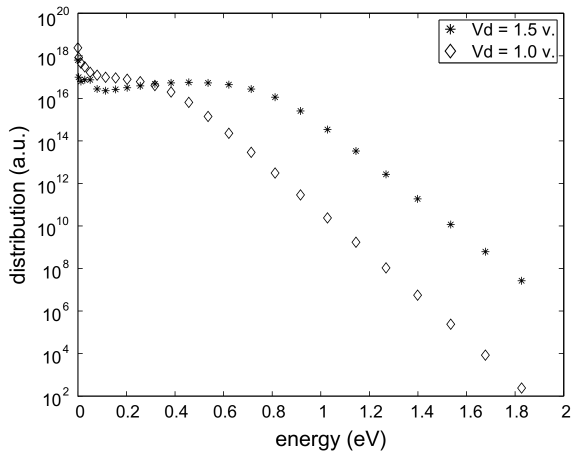

Fig. 11. Electron energy distribution in cell in channel for 70-nm channel length device with 1.5 V applied on the gate. The drain voltage is varied from 1.5 V to 1.0 V.

The peak value of energy, however, happens at a value of energy which is near to or slightly less than the value given by the electric field. This occurs because the electrons lose a portion of energy as they undergo collisions with the lattice. The electrons at higher energy have a shorter mean free path and are less "lucky" than those at lower energy. This explains how the distribution comes to have the two peaks in the 35-nm device, but the higher peak is not present in the 70-nm device because λ/L is smaller. The electrons start off, at low energy, with a long mean free path, in which case they go quite a distance without colliding. As the energy grows, the chance of collisions increases substantially, which in turn causes the number of electrons left in the peak to drop.

In the next section we present some concluding remarks about what we have seen in this work, i.e., the method that has been implemented and its effectiveness in solving for the electron energy distribution in a MOSFET. Proposed future work for this implementation of the CS is also discussed.

4. Conclusions

The transition probability matrix method has been extended to handling 2D boundary conditions for the simulation region. Additionally, the CS has been extended to calculation of polygon overlap for polygons of more than three sides. The approach was applied to 2D simulation of a MOSFET device, and results were obtained in a reasonable amount of simulation time. High stress conditions were applied to the transistor device, and the method obtained converged distribution functions in a scheme with low numerical diffusion and no statistical noise. A number of scattering mechanisms were implemented in the collision operator, including acoustic phonon scattering, optical (non-polar) phonon scattering, and impact ionization. In a case where scattering is approximately isotropic, the method only needs to find a variable R which is a function of space and energy, i.e., there is no angular dependence in phase space. Even though the distribution of particles f depends on angle, R does not. In the event that the scattering does depend on angle, R is still likely to be a simple function of angle. The method provides full information about the angular distribution in f, but it is much more efficient to be able to calculate R because R is a much smaller array. The CS minimizes numerical diffusion because of the long distances particles go between being launched and being mapped back to the mesh. The method presented here cuts down significantly on numerical diffusion compared to solving the Boltzmann transport equation by finite differences. The CS is more accurate than expansion by spherical harmonics, as usually implemented. In addition, it can resolve the tail better than a Monte Carlo simulation is able to do. Future work will involve adding coulomb collisions in order to assess their impact on the distribution of electrons and the conditions for when they are important. In conclusion, the method proved to be an efficient and accurate choice for obtaining the electron energy distribution in a MOSFET.

Acknowledgment

The authors thank I. Knezevic for valuable discussions.

Appendix A. Polygon overlap

In using the Convective Scheme, the calculation of overlap of two polygons is an integral part of the method. The overlap is used to determine the fraction of particles that gets mapped to a specific fixed cell. In Fig. A.1 below, the shaded area represents the area of overlap between two quadrilaterals. In order to calculate the area of overlap of two polygons, the coordinates of each of the polygons are accessed by a function in the code. If the polygon has more than three sides, it is absolutely necessary to put the corners "in order" along the perimeter. Otherwise, the area could be calculated incorrectly. For example, in the case of a quadrilateral, if diagonally opposite points are numbered next to each other, the result will be erroneous. So, steps are taken to ensure that the corners of a quadrilateral, for example, are ordered as one would follow the perimeter of the cell. This is accomplished by first finding the corner with the lowest value of x-coordinate and establishing it as the minimum. Given a vertical line drawn through that point, a line segment can be formed from that corner to each of the other corners of the cell. The angle between the vertical line segment and each of the other line segments is calculated. Once the angles are calculated, this information can be used to order the corners according to increasing angle.

After finding the centroid of the polygon, the code finds the inward-facing normal to each side of the polygon. The points needed in order to determine the area of overlap are corners of the overlying moving cell which lie inside the underlying fixed cell, corners of the underlying fixed cell that lie within the overlying moving cell, and points of intersection between the sides of the two polygons. The centroid of a triangle can be found by taking a vertex of a side and drawing a line segment from the midpoint of that side to the opposing vertex. This is done for each side, and where these line segments intersect is the centroid of the triangle. The centroid of a quadrilateral is located at the intersection of the two bimedians. This is formed by finding the midpoints of the sides, connecting the midpoints of opposite sides with line segments, and finding the intersection of these two line segments.

Generally, in the simulation the polygons are triangles and quadrilaterals. However, in the event that a polygon moves into a corner of the simulation region, it subsequently gets broken up into smaller pieces due to truncation and reflection. The code can actually process pentagons as well as triangles and quadrilaterals, because in some cases the pieces result in a pentagon in the corners of the simulation region. However, the code would only need to find the area of overlap between a pentagon and either a triangle or a quadrilateral. The reason for this is that at least one of the polygons will be a triangle or quadrilateral when mapping particles to the underlying mesh. For example, if a cell migrates to the lower right corner and gets distorted slightly by the electric field, the pieces into which the cell gets broken up may

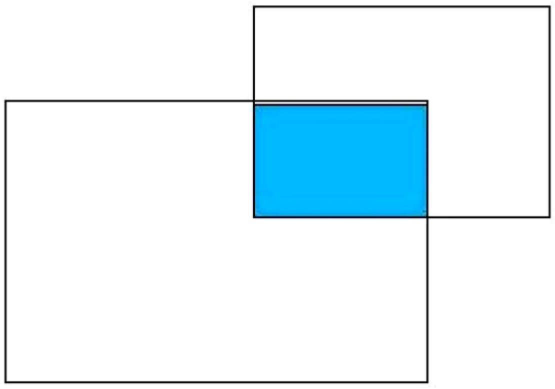
Fig. A.1. Overlap of two polygons.

be pentagons. The pieces are the temporary cells that result when doing reflection. One piece is the portion in the simulation region, and the other piece is the portion of the cell that is outside the simulation region and gets reflected.

A.1. Calculating the centroid of an irregular polygon

The centroid of a triangle can be found by taking the average of its three vertices. One way to obtain the centroid of the pentagon (or other polygon) is to first triangulate the polygon, then form a sum of the centroids of each triangle, weighted by the area of each triangle, and then normalize the whole sum by the total polygon area.

Yet, there is an alternate method: instead of a partition one can use positively and negatively oriented triangles as is done when calculating the area of a polygon. The simple algorithm is based on a sum of triangle centroids weighted with their signed area. The triangles are formed by taking one fixed vertex v_0 of the polygon and the two endpoints of consecutive edges of the polygon: (v_1, v_2), (v_2, v_3), etc. The area of a triangle with vertices v_0, v_1, v_2 is given by

$$area = 1/2((v_1(x) - v_0(x)) * (v_2(y) - v_0(y)) - (v_2(x) - v_0(x)) * (v_1(y) - v_0(y))) \tag{A.1}$$

So, the algorithm can be described as follows:

1. Start with any vertex of the polygon and use it as a vertex for the $n - 2$ triangles (where n is the number of vertices in the polygon) that are produced by connecting it with each of the other vertices.
2. Calculate the coordinates of the centroid (XC, YC) for each triangle using:

$$XC = \frac{v_0(x) + v_1(x) + v_2(x)}{3} \tag{A.2}$$

and

$$YC = \frac{v_0(y) + v_1(y) + v_2(y)}{3}. \tag{A.3}$$

3. Next calculate the area of each triangle according to:

$$Area = \frac{[(v_1(x) - v_0(x)) * (v_2(y) - v_0(y))] - [(v_2(x) - v_0(x)) * (v_1(y) - v_0(y))]}{2}. \tag{A.4}$$

4. Calculate the area of the entire polygon.
5. Calculate the fraction of the total area contributed by each triangle.
6. Next calculate the product of each coordinate from step 2 and the result from step 5.
7. Find the sum of the results from step 6, which yields the coordinates of the centroid for the whole polygon.

$$x\text{- coord} = \frac{(XC_1 * Area_1 + XC_2 * Area_2)}{(Area_1 + Area_2)}, \tag{A.5}$$

$$y\text{- coord} = \frac{(YC_1 * Area_1 + YC_2 * Area_2)}{(Area_1 + Area_2)}. \tag{A.6}$$

A.2. Finding a normal to a side

In order to find a normal to a side of the polygon, one can take a "half-side" segment and rotate it by 90° counterclockwise. The equations for rotation of axes, i.e., a counterclockwise rotation through angle α are:

$$x = x' \cos\alpha - y' \sin\alpha \tag{A.7}$$

and

$$y = y' \sin\alpha + y' \cos\alpha \tag{A.8}$$

When $\alpha = 90°$, these reduce to

$$x = -y' \tag{A.9}$$

and

$$y = x' \tag{A.10}$$

which means that in the 90° counterclockwise rotation, the x-coordinate becomes the y-coordinate, negated, and the y-coordinate becomes what the x-coordinate was.

A.3. Determining whether a point is inside

After finding the centroid of the polygon and finding the normal to each side, vectors are used to determine whether or not a point is inside the polygon. For a given side, we have a normal vector and also a vector pointing from the center of the face to the centroid. We take the dot product of $(xcentroid - xc) \cdot \hat{n}$. Then we take another point, called *xother*, and find a vector from the midpoint of the side, *xc*, to *xother*. We find the dot product of $(xother - xc) \cdot \hat{n}$. If the two dot products have the same sign, the point is on the inward-facing side of the line, i.e., inside the polygon.

A.4. Calculating the polygon area

In the actual calculation of the area of overlap, there are capabilities for calculating the overlap of both triangles and quadrilaterals with irregular polygons of three, four, five, or six sides. Using the functions that calculate the centroid, the normal to a side of a polygon, the intersection of two line segments, and the function for determining whether or not a point is inside a polygon, the corners of the region of overlap are determined.

Once these points have been found, the code checks for duplicate points and takes them out of the array. The points remaining are put in order of increasing angle, and then the polygon area is determined by breaking it up into triangles, calculating the area of each triangle and summing these together for the total area.

Appendix B. Boundary conditions

In this appendix on boundary conditions, the first section explains the boundaries that truncate cells entering or leaving the simulation region. Next, the reflecting boundaries at the top and bottom of the simulation region are explained. Following this comes a discourse on the corners of the simulation region.

B.1. Truncating boundaries

The boundaries on the right and left sides of the simulation region are truncating boundaries for moving cells entering and leaving the simulation region. A boundary cell may enter from the region outside the main simulation region or a moving cell may leave the simulation region (this could be a cell that originated from inside the simulation region or a boundary cell that came from one end and traversed the simulation region). For a transistor device built on a silicon wafer, there will be electrons flowing in and out of the device. When a cell that is straddling a truncating boundary maps particles, only the portion that overlaps the simulation region will map particles. The particles in the remainder of the cell are considered to have left and thus make no contribution to the fixed cells on the mesh. First the code checks whether the cell is completely outside of the simulation region. If this is the case, it next checks to see whether or not the cell is a boundary cell (a cell that originated outside the simulation region and made its way inside). If the cell is not a boundary cell and it is completely outside the simulation region, then it is just considered empty. If the cell is in fact a boundary cell, then the code checks whether or not the cell is moving away from the simulation region. If it is moving away, then that cell is also considered to be empty. If the boundary cell is moving toward the simulation region, it still has a chance to contribute to mapping particles before the end of the iteration. In this case it is not considered to be empty at the end of that timestep.

If the cell has some overlap with the simulation region, the next step is to take each side of the moving cell and check for intersection with the particular boundary (left or right). Points of intersection with that boundary as well as corners of the moving cell that are inside of the simulation region become corners of the "temporary" cell used to map particles in this case. The code next checks for duplicates (in the event that the same point gets found twice). This could happen, for example, if a corner of the moving cell lies directly on the boundary of the simulation region. Also, the appropriate fraction of particles is put into the temp cell (this is determined by the amount of overlap of the moving cell with the simulation region). In this section, we have discussed the boundaries that represent the ends of the MOSFET device that are open to the rest of the silicon wafer. The boundaries at the top and bottom of the device, and hence of the simulation region, which are treated as boundaries that reflect cells, are explained in the next section.

B.2. Reflecting boundaries

A cell that reflects off of the top or bottom boundary continues to move after being reflected. A temporary cell is used for mapping back particles, since the original cell would be beyond either the top or bottom boundary. The moving cell still exists beyond the boundary, and to find the temporary cell, the moving cell is reflected into the simulation domain. The original moving cell continues until either the end of an iteration or it reflects off a reflecting boundary a sixth time, at which time it gets emptied. More than one reflection off of a top or bottom boundary is allowed so that particles will not accumulate near a boundary in a non-physical manner. The code keeps track of the coordinates of the cell as if the cell continues to move beyond the top or bottom boundary. At the end of each timestep, the temporary cell or cells is used for mapping back particles. If the cell has been reflected an odd number of times, the y-component of electric field is negated, since the cell is (seemingly) moving in the opposite direction to its actual vertical direction of motion. For a cell that is reflecting off the top or bottom boundary, the code first checks for how many times the moving cell has reflected. It next reflects the cell the appropriate number of times and places the appropriate coordinates into the temporary cell. If the cell has progressed so far as to require a sixth reflection, that cell is immediately emptied by mapping all of the particles back into the cells that it overlaps.

If it is determined that the cell landed at a position straddling a boundary, there is a function that splits the cell into two cells. Each one of these cells is assigned the correct amount of particles by calculating the area of the temporary cell and the area of the moving cell and taking the ratio of the two. If the cell is straddling a boundary (top or bottom), then the points of intersection with that boundary get calculated and included as corners of the temporary cell as well. In some instances, the cell may end up at a reflecting boundary and at the same time be entering or leaving the simulation region. At this point, the cell is in a corner of the simulation region, and this case is covered next.

B.3. Corners

The corners represent one of the more challenging aspects of the code. Specific code was written for each corner (upper right corner, lower right corner, lower left corner, and upper left corner), with the triangles and quadrilaterals treated separately. A cell that is either above the top boundary or below the bottom boundary AND to the right or left of the simulation region is a "corner" cell. Once it has been determined that a cell is in a corner, the code first checks whether or not the cell is completely outside of the simulation region (either it has not entered yet from the right or left boundary OR it has exited through one of these boundaries). Unless a cell is still outside and is a boundary cell that has not yet entered, it gets emptied when it is completely outside the simulation region. This is determined by checking which direction the cell is heading; if it is outside the right boundary and is heading further to the right, then it gets emptied. Similarly, if the cell is outside of the left boundary and is heading further to the left, it then gets emptied at the end of the timestep.

Fig. B.2 depicts a cell mapping in the upper right corner of the simulation region, with the truncated portion highlighted. If the cell has overlap with the simulation region, the next step is to check for points of intersection with the right or left boundary. Points of intersection with the side boundary as well as corners of the original moving cell are read into an intermediate temporary cell. The code next checks for duplicates (in case the same point gets found more than once). The intermediate cell has 3, 4, or 5 corners. The appropriate

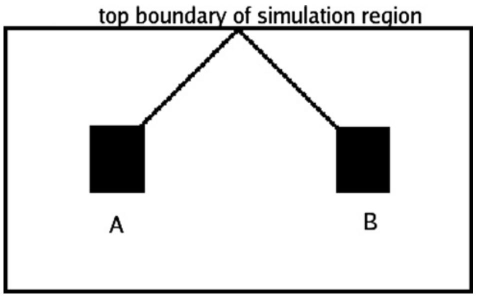

Fig. B.1. For a cell reflecting at the boundary, the angle of incidence equals the angle of reflection. The cell begins at position A and ends at position B at the end of a timestep. ϕ is the angle that the line of approach toward the top boundary makes with the positive x-axis, and $2\pi - \phi$ is the angle at which the cell rebounds after reflecting off the boundary.

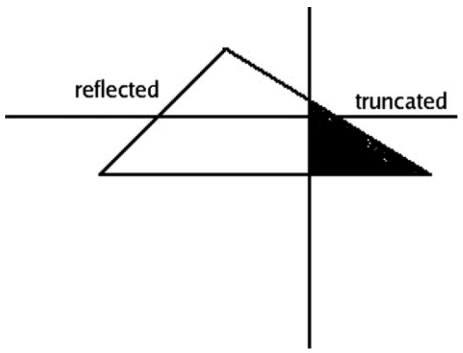

Fig. B.2. Cell mapping in upper right corner.

fraction of particles gets put into the intermediate temporary cell based on the area of overlap of the intermediate cell with the original moving cell. At this point the intermediate temporary cell gets treated much like a cell reflected off the top or bottom boundary. This is appropriate because that portion of the original moving cell is a cell that has reflected in this manner.

Appendix C. Collision frequencies

Any one of the scattering processes can induce the carrier to make a transition from an initial state **k** to a final state **k**′ with a probability $P(\mathbf{k}, \mathbf{k}')$. The number of electrons scattered depends upon this probability as well as the probability (i) of the state **k** being full and (ii) of the state **k**′ being empty. Then, the rate of scattering out of state **k** is determined by

$$P(\mathbf{k},\mathbf{k}')f(\mathbf{k})[1-f(\mathbf{k}')]. \tag{C.1}$$

In the same fashion, electrons may scatter into state **k** from the state **k**′ with a rate of

$$P(\mathbf{k},\mathbf{k}')f(\mathbf{k}')[1-f(\mathbf{k})], \tag{C.2}$$

where $P(\mathbf{k}, \mathbf{k}')$ is the scattering rate, arrived at through time-dependent, first-order perturbation theory, via the Fermi golden rule:

$$P(\mathbf{k},\mathbf{k}') = \frac{2\pi}{\hbar}|M(\mathbf{k},\mathbf{k}')|^2\delta(\xi_k - \xi_k' \pm \hbar\omega_q). \tag{C.3}$$

Here, the upper sign corresponds to the absorption of a phonon and the lower sign corresponds to the emission of a phonon. $M(\mathbf{k}, \mathbf{k}')$ is the matrix element of the perturbing electron–phonon interaction, given by

$$M(\mathbf{k},\mathbf{k}') = \langle\Psi_{k'q}|\delta\xi|\Psi_{kq}\rangle \tag{C.4}$$

and $\delta\xi$ is the scattering potential [1]. The general electron–phonon coupling strength is given by

$$\Delta_\eta(\mathbf{k},\mathbf{k}') = \left[\frac{2\rho_\mathrm{m} V \omega_{q,\eta}}{\hbar}\right]^{1/2} |M(\mathbf{k},\mathbf{k}')|^2, \tag{C.5}$$

where ρ_m is the mass density of the crystal and V is the macroscopic volume of the crystal. Scattering models commonly used in Monte Carlo methods as well as in this work may be obtained by assuming that the coupling strength Δ_η is independent of both $\mathbf{k}$ and $\mathbf{k}'$, and depends only on the energy of the initial state [2]. Types of scattering used in the simulation include acoustic phonon scattering, intervalley phonon scattering, ionized impurity scattering and impact ionization, which are described next.

C.1. Acoustic phonon scattering

Interaction of the electrons (or holes) with the acoustic modes of the lattice through the deformation potential is one of the most common phonon scattering processes. A long-wavelength acoustic wave moving through the lattice can cause a local strain in the crystal, the end result being a weak scattering potential. The rate of scattering out of the state defined by wavevector $\mathbf{k}$ is

$$\Gamma(\mathbf{k}) = \frac{\Xi_1^2 k_\mathrm{B} T (2m^*)^{3/2}}{2\pi\hbar^4 \rho_\mathrm{m} v_\mathrm{s}^2} E^{1/2}, \tag{C.6}$$

where ρ_m is the mass density, Ξ is the deformation potential, E is the energy of the carriers, m^* is the electron effective mass, and v_s is the velocity of sound.

The range of energy change involved in the case of acoustic phonon scattering is $2\hbar v_\mathrm{s} k$. Considering an average k-value of the order of 1.0e7/cm, this is about 1 meV. Since this energy is small in comparison with thermal energies of $kT = 25$ meV, scattering by absorption of acoustic phonons is approximately elastic [38].

C.2. Optical and intervalley scattering

In this section the non-polar optical phonon scattering (also referred to as intervalley scattering) is considered. Two possible types of phonons can be involved in this process as previously mentioned. The first one, g-phonon scattering, couples the two valleys along opposing ends of the (1 0 0) axis. The f-phonon scattering couples the (1 0 0) valley to the (0 1 0) and (0 0 1) valleys, etc [39]. The contribution of the different valleys is accounted for by the coupling constant Ξ_0, which is fitted to data [40]. In other words, the expression used for the scattering rate depends on the carrier energy, but there is no direct tracking of the valley in which the electrons reside. In the case of intervalley phonon scattering, which can also be referred to as scattering between different valence-band valleys, the assumption of constant frequency (for the emitted or absorbed phonon) is again reasonable. In order to obtain the matrix element, a deformable ion model is assumed, wherein the two sublattices move relative to each other. A deformation field results because the potential field of each ion is displaced, resulting in a small excess positive charge where the ions have moved apart and a slight negative charge where they are closer together.

The rate of scattering out of the state $\mathbf{k}$ is [39]

$$\Gamma(\mathbf{k}) = \sqrt{\frac{m^*}{2}} \frac{m^* D^2}{\pi\rho\hbar^3\omega_0} \left[N_q\sqrt{E_k + \hbar\omega_0} + (N_q + 1)\sqrt{E_k - \hbar\omega_0} u_0(E_k - \hbar\omega_0)\right], \tag{C.7}$$

where D is the macroscopic deformation field, having units of eV/cm, ρ is the mass density of the semiconductor, and N_q is the phonon occupation function. For intervalley scattering, the expression needs to be multiplied by the number of final ellipsoids to which the carrier can scatter, although this factor can be accounted for within the density-of-states effective mass, m^*, which appears in the equation.

The general electron–phonon interaction is an expansion in powers of q, so that the zero-order interaction is the q^0 order term. The first-order term for optical deformation potential scattering is a result similar to the acoustic deformation potential scattering, which is itself a first-order process. The rate of scattering out of the state $\mathbf{k}$ is [39]

$$\mathbf{\Gamma}(\mathbf{k}) = \frac{\sqrt{2}(m^*)^{\frac{5}{2}}\Xi_0^2}{\pi\rho\hbar^5\omega_0}[N_q(2E_k + \hbar\omega_0)\sqrt{E_k + \hbar\omega_0} + (N_q + 1)(2E_k - \hbar\omega_0)\sqrt{E_k - \hbar\omega_0}u_0(E_k - \hbar\omega_0)], \tag{C.8}$$

where Ξ_0 is the energy shift, or first-order deformation potential, ρ is the mass density, and N_q is the phonon occupation function. The first term inside the brackets is the absorption term, and the second term is the emission term. In the code a constant value of $\hbar\omega_0 = 63$ meV was used for the optical phonons, since this is the value for silicon found in the literature.

C.3. Ionized impurity scattering

When treating electron scattering from the Coulomb potential of an ionized impurity atom, it is important to take into account the long-range nature of the potential. Some type of cutoff mechanism needs to be invoked, because if the interaction is summed over all space, the integral will diverge. For spherical symmetry about the scattering center, or location of the ion, the potential is screened in such a way that gives rise to [1]

$$\mathbf{\Phi}(\mathbf{r}) = \frac{e}{4\pi\epsilon_\infty\mathbf{r}}\exp(-q_\mathrm{d}r), \tag{C.9}$$

where q_d is the Debye screening wave vector.

A charged center in the lattice can be thought of as a purely coulombic potential. This potential, being inversely proportional to the radial distance r, distorts a plane wave at all distances, resulting in a scattering cross-section that is effectively infinite. In this work, the Conwell–Weisskopf approximation is employed [38]. The approximation is applied by limiting the impact parameter to half the average separation distance of the impurities, or

$$b_\mathrm{max} = \frac{1}{2}N_\mathrm{I}^{-1/3}, \tag{C.10}$$

where N_I is the impurity concentration in the semiconductor. This results in a total cross-section of

$$\sigma = \pi b_\mathrm{max}^2. \tag{C.11}$$

The mass of the impurity atom in the lattice greatly exceeds that of the electron, and thus the collisions are very close to elastic. The scattering rate for ionized impurity scattering is given by the product of the cross-section, $\sigma(\theta)$, the concentration of scatterers, N_I, and the velocity, v, of the carrier, or

$$\mathbf{\Gamma}(\mathbf{k}) = N_\mathrm{I}\sigma(\theta)v, \tag{C.12}$$

where

$$N_\mathrm{I}\sigma(\theta)v = \left(\frac{Ze^2}{4\pi\epsilon_\infty}\right)^2\frac{\pi N_\mathrm{I}}{\sqrt{2m^*E_k^3}}\log\left[1 + \left(\frac{4\pi\epsilon_\infty E_k}{N_\mathrm{I}^{1/3}Ze^2}\right)^2\right], \tag{C.13}$$

where ϵ_∞ is the high-frequency permittivity, E_k is the energy of the carriers, and $Z = 1$ in the case of silicon [38].

C.4. Impact ionization

Impact ionization, also known as the inverse Auger process, occurs when an energetic electron gives up its energy to a valence electron, allowing it to be raised to the conduction band and creating a hole in the valence band. Given an electron with energy above a certain threshold value, impact ionization can occur. When impact ionization occurs, three current carriers result – two electrons and a hole. There are a multiplicity of threshold energies for impact ionization found in the literature [39]. Many scientists employ the Keldysh formula approach [41], which was derived based on a direct gap semiconductor with parabolic bands. This approach is often applied even to silicon simulations. Values of threshold energy used range from the bandgap energy of silicon to twice the bandgap. The threshold energy is taken to be 2.1 eV. The reason that the value of energy used is greater than the bandgap energy is due to the fact that in an indirect semiconductor, the process involves a change in momentum or a Brillouin zone shift. The scattering rate is

$$\mathbf{\Gamma}_{\mathbf{i}}(\mathbf{E}) = \frac{m_e^3 e^4}{8\pi^2 \epsilon_s^2 \hbar^3 m_h^2}\left(1+\frac{m_h}{m_0}\right)\left(\frac{E-E_T}{E_T+E_D}\right)^2 u_0(E-E_T), \tag{C.14}$$

where m_e is the appropriate electron effective mass, m_h is the appropriate hole effective mass, E_T is the threshold energy for impact ionization and $E_D = \hbar^2 q_d^2/2m_h$ is an energy which corresponds to a hole with the Debye screening wave vector [39].

References

[1] David K. Ferry, Semiconductors, Macmillan, 1991.

[2] Antonio Abramo et al., A comparison of numerical solutions of the Boltzmann transport equation for high-energy electron transport in silicon, IEEE Trans. Electron Dev. 41 (9) (1994) 1646–1654.

[3] E.G.S. Paige, Progress in Semiconductors, Vol. 8: The Electrical Conductivity of Germanium, John Wiley and Sons, Inc., 1964.

[4] M.H. Jorgensen, N.O. Gram, N.I. Gram, Negative differential conductivity and current oscillations in lightly doped N-type silicon, Solid State Commun. 10 (4) (1972) 337–340.

[5] M.C. Vecchi, M. Rudan, Modeling electron and hole transport with full-band structure effects by means of the spherical-harmonics expansion of the BTE, IEEE Trans. Electron Dev. 45 (1) (1998) 230–238.

[6] T. Kurosawa, Notes on theory of hot electrons in semiconductors, J. Phys. Soc. Jpn. 20 (6) (1965) 937–942.

[7] I.B. Levinson, Relaxation time warming function and escape effect of hot electrons in semiconductors, Sov. Phys. – Solid State, USSR 6 (7) (1965) 1965.

[8] H. Budd, Path variable formulation of the hot carrier problem, Phys. Rev. 158 (3) (1967) 798–804.

[9] H.D. Rees, Numerical solution of electron motion in solids, J. Phys. C: Solid State Phys. 5 (6) (1972) 641–656.

[10] T. Kurosawa, Monte Carlo calculation of hot electron problems, J. Phys. Soc. Jpn. Suppl. 21 (1966) 424–426.

[11] C. Jacoboni, L. Reggiani, The Monte Carlo method for the solution of charge transport in semiconductors with applications to covalent materials, Rev. Mod. Phys. 55 (3) (1983) 645–705.

[12] W. Fawcett, A.D. Boardman, S. Swain, Monte Carlo determination of electron transport properties in gallium arsenide, J. Phys. Chem. Solids 31 (9) (1970) 1963–1990.

[13] C. Jacoboni, Recent developments in the hot-electron problem, in: G. Fumi (Ed.), Proceedings of the 13th International Conference on the Physics of Semiconductors, Rome, North-Holland, Amsterdam, 1976, pp. 1195–1205.

[14] A.M. Kriman, M.J. Kann, D.K. Ferry, R. Joshi, Role of the exchange interaction in the short-time relaxation of a high-density electron-plasma, Phys. Rev. Lett. 65 (13) (1990) 1619–1622.

[15] H. Shichijo, K. Hess, Band-structure-dependent transport and impact ionization in GaAs, Phys. Rev. B 23 (8) (1981) 4197–4207.

[16] M.V. Fischetti, S.E. Laux, Monte Carlo analysis of electron transport in small semiconductor devices including band-structure and space–charge effects, Phys. Rev. B 38 (14) (1988) 9721–9745.

[17] J. Jakumeit, U. Ravaioli, Semiconductor transport simulation with the local iterative Monte Carlo technique, IEEE Trans. Electron Dev. 48 (5) (2001) 946–955.

[18] K. Kometer, G. Zandler, P. Vogl, Lattice-gas cellular-automaton method for semiclassical transport in semiconductors, Phys. Rev. B 46 (3) (1992) 1382–1394.

[19] M. Saraniti, S.M. Goodnick, Hybrid fullband cellular automaton/Monte Carlo approach for fast simulation of charge transport in semiconductors, IEEE Trans. Electron Dev. 47 (10) (2000) 1909–1916.

[20] W.N.G. Hitchon, G.J. Parker, J.E. Lawler, Physical and numerical verification of discharge calculations, IEEE Trans. Plasma Sci. 21 (2) (1993) 228–238.

[21] W.N.G. Hitchon, D.J. Koch, J.B. Adams, An efficient scheme for convection-dominated transport, J. Comput. Phys. 83 (1) (1989) 79–95.

[22] K.M. Kramer, W.N.G. Hitchon, Semiconductor Devices: A Simulation Approach, Prentice-Hall, 1997.

[23] W.N.G. Hitchon, Plasma Processes for Semiconductor Fabrication, Cambridge University Press, 1999.

[24] J.M. Higman, K. Hess, C.G. Hwang, Coupled Monte Carlo-drift diffusion analysis of hot-electron effects in MOSFETs, IEEE Trans. Electron Dev. 36 (5) (1989) 930–937.

[25] C. Jungemann, S. Yamaguchi, H. Goto, On the accuracy and efficiency of substrate current calculations for sub-μm nMOSFETs, IEEE Electron Dev. Lett. 17 (10) (1996) 464–466.

[26] S. Harris, An Introduction to the Theory of the Boltzmann Equation, Holt, Rinehart and Winston, 1971.

[27] G.J. Parker, W.N.G. Hitchon, E.R. Keiter, Transport of ions during ion implantation, Phys. Rev. E 54 (1) (1996) 938–945.

[28] A.J. Christlieb, W.N.G. Hitchon, Three-dimensional solutions of the Boltzmann equation: heat transport at long mean free paths, Phys. Rev. E 65 (5) (2002) 056708-1–056708-12.

[29] W.N.G. Hitchon, E.R. Keiter, Kinetic simulation of a time-dependent 2-dimensional plasma, J. Comput. Phys. 112 (2) (1994) 226–233.

[30] H.L. Berk, K.V. Roberts, Methods Comput. Phys. 9 (1970) 88.

[31] A.J. Christlieb, W.N.G. Hitchon, E.R. Keiter, A computational investigation of the effects of varying discharge geometry for an inductively coupled plasma, IEEE Trans. Plasma Sci. 28 (6) (2000) 2214–2231.
[32] J. Feng, W.N.G. Hitchon, Boltzmann equation description of flows at long mean free paths, Phys. Rev. E 70 (3) (2004) 036704-1–036704-10.
[33] E. Takeda, C.Y. Yand, A. Miura-Hamada, Hot-Carrier Effects in MOS Devices, Academic Press, 1995.
[34] K. Hasnat, C.-F. Yeap, S. Jallepalli, S.A. Hareland, W.-K. Shih, V.M. Agostinelli Jr., A.F. Tasch, C.M. Maziar, Thermionic emission model of electron gate current in submicron NMOSFETs, IEEE Trans. Electron Dev. 44 (1) (1997) 129–138.
[35] W. Shockley, Problems related to p–n junctions in silicon, Solid-State Electron. 2 (1) (1961) 35–60.
[36] P.A. Wolff, Theory of electron multiplication in silicon and germanium, Phys. Rev. 95 (6) (1954) 1415–1420.
[37] K.G. Anil, S. Mahaptra, I. Eisele, Experimental verification of the nature of high energy tail in the electron energy distribution in n-channel MOSFETs, IEEE Electron Dev. Lett. 22 (10) (2001) 478–480.
[38] B.K. Ridley, Quantum Processes in Semiconductors, Oxford University Press, 1999.
[39] K. David, Ferry, Semiconductor Transport, Taylor and Francis, 2000.
[40] D.K. Ferry, First-order optical and intervalley scattering in semiconductors, Phys. Rev. B 14 (4) (1976) 1605–1609.
[41] L.V. Keldysh, Concerning theory of impact ionization in semiconductors, Sov. Phys. JETP 21 (6) (1965) 1135.
[42] D.S. Ang, T.W.H. Phua, H. Liao, C.H. Ling, High-energy tail electrons as the mechanism for the worst-case stress degradation of the deep submicrometer N-MOSFET, IEEE Electron Dev. Lett. 24 (7) (2003) 469–471.