FDSIM: Finite-Difference Simulation Tools for 1-D and 2-D Semiconductor Device

Author: Yu-Hung Liao (b01901152@ntu.edu.tw)
Instructed by Professor Yu-Ren Wu

Abstract The "fdsim" package contains a set of simulation tools which provides user-friendly interfaces for freely configuring the geometry of a semiconductor device. Both 1-D and 2-D solvers are provided, and the simulations for heterojunctions are supported. The Shockley-Read-Hall (SRH) recombination model and the Tsu-Esaki electron tunneling current are integrated with the drift-diffusion problem, and the integrated model is solved self-consistently. Therefore, a variety of semiconductor device physics can be studied with the tools.

Introduction

Dependencies

The solvers are written in python2.7, utilizing the following open-source packages.

- numpy: for building the mesh matrices and vector representations of the variables
- scipy: for creating sparse-matrix operators and solving the linear algebra
- matplotlib: for visualizing the band diagram and the carrier concentrations

Structure of the Solver

The "dev_solver2D" python class in Fig. 1 provides methods to configure and solve the boundary value problem. It inherits from both the "p_solver2D" (Poisson solver) and the "J_solver2D" (current solver) which are children of the "solver2D" class and share the same variable data. The "__solver" class is the base class for all solvers classes. The functions of these classes are explained by the charts in Fig. 1.

The concepts and implementations for solving the 1-D problems are very similar. Though not presented in Fig. 1, all 1-D solvers are also based on "__solver" class. However, the details for a 2-D problem is more complicated. Therefore, the following discussions will focus on the 2-D cases.

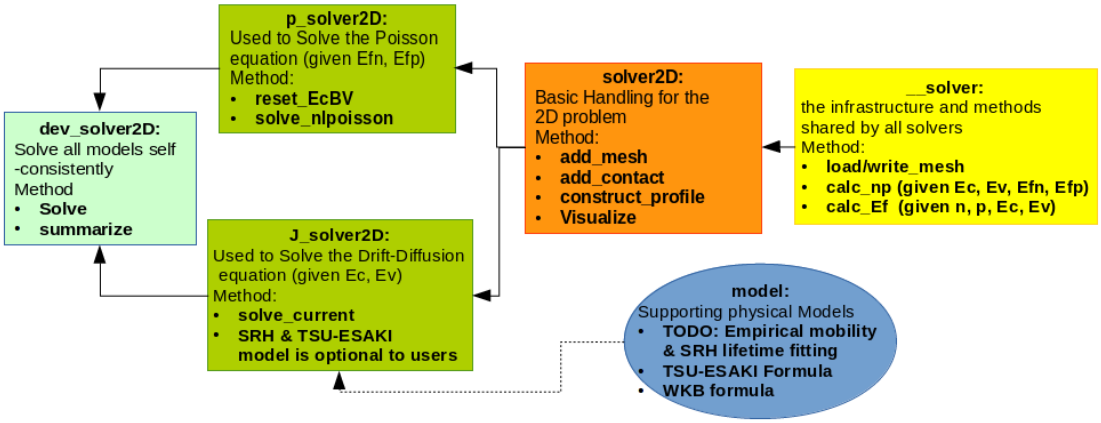

Figure 1. The hierarchy of the solver2D classes. Relations represented by solid line with arrow: inheritance (parents to children); dashed line with arrow: module utility support.

User Interface and the Solving Routine

Building a Boundary Value Problem

The geometry of a boundary value problem is presented by a collection of meshes of which the material and doping concentration are configurable. Besides, the boundary condition is determined by the voltages and carrier concentrations of the contacts at the edges of meshes. The solver provides two methods, "add_mesh" and "add_contact", for the users to freely specify the profile.

- Adding mesh

When the user attempts to add a new mesh, the position, size and material of the new mesh must be specified. Then the solver will do the following tasks in chronological order.

1. Overlays between new and existing meshes will be detected first. If any overlays are found, the solver will warn the user and abort the process.
2. The solver then creates a new "mesh2D" object and assigns unique indexes to each mesh point incrementally. The indices will later be used for mapping all 2-D points to a 1-D vector.
3. All junctions between the newly-created and the old meshes will be detected. A "junction" object is created for each junction, recording the neighboring indexes and materials of both sides.
4. The method will return a reference of the newly-created mesh object so that the user can use it to set the doping profile.

While the variables of all 2-D points will be mapped to 1-D vectors for solving the problem, the mesh2D objects which contain 2-D data are kept for visualization purposes. It also provides calculation convenience for solving the continuity equation. Initial values for the "Ec", "Ev", "Efn", "Efp", "n" and "p" of the new mesh are set according to the intrinsic properties and the electron affinity of the mesh material.

On the other hand, the junction objects created in step 3 is crucial for integrating all meshes and creating a complete profile. It helps the solver build the operator matrices and find possible tunneling paths.

- Adding Contact

The contact is a point in a 1-D problem and a line segment in a 2-D problem. The "add_contact" method takes the position and length of the segment as arguments. After the checking routines similar to "add_mesh", a "contact" object is created if the given segment is on the edge of some meshes. The contact object logs the material and indices of the mesh points next to the contact. At the end, a reference of the contact is returned.

The "n" and "p" attributes of the contacts should be set as boundary values of the drift-diffusion equation. Besides, the "V" attribute can be set for the Poisson equation, and the solver will calculate the corresponding E_c (= 4.5-χ-V, χ: electron affinity) and

E_v (=E_c-E_g) which are actually used in solving the problems.

Constructing the Complete Profile

After all the meshes, contacts and the doping profiles are set, the user has to call the "construct_profile" method before initiating the solving routine. When this method is called, the following tasks will be respectively done by different classes.

1. The solver2D class will allocate memories for seven 1-D vectors of equal lengths which contain variables (E_c, E_v, n, p, E_{fn}, E_{fp} and N_B) mapped from all mesh points. The indices of all pairs of neighboring points will be logged for constructing the operator matrices. Indices of all externally contacted points will also be logged.
2. The p_solver2D class will create the Poisson operator for the 1-D representative vectors, and modify the RHS of the nonlinear Poisson equation according to the conduction band edge offsets extracted from the properties of all "junction" objects. More details we be discussed in the following sections.
3. The J_solver2D class will allocate memories which will be used for constructing the continuity equations matrices based on the Scharfetter-Gummel discretization. It will also find possible 1-D tunneling paths and create "T_path" objects to wrap them. More details we be discussed in the following sections.

Note that the boundary values (V, n and p of the contacts) can still be changed after "construct_profile" method is called. The profile only has to be constructed once, and various boundary conditions can be applied on it.

Iterations for Solving the Problem

Figure 2 represents the solving routine. Firstly, the Poisson equation is solved with given E_{fn} and E_{fp}, then the resulted E_c and E_v are used by the current solver. The current solver solves the current models self-consistently, and the corresponding E_{fn} and E_{fp} are again used to solve the Poisson equation. The iteration continues until the maximum change of E_c is smaller than the tolerance.

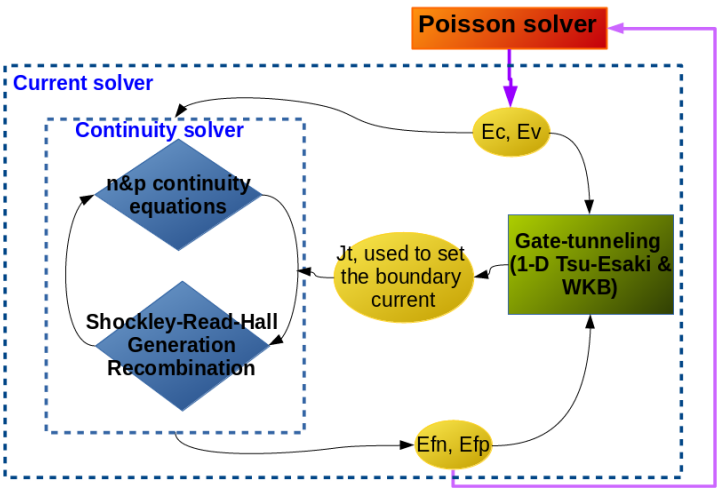

Figure 2. The scheme of iterations for solving all equations self-consistently.

The current solver consists of a continuity solver and a calculating functions for 1-D tunneling currents. The continuity solver solves the n and p continuity equations with SRH model, then the quasi-fermi levels at the insulating junctions and the electric potentials are used to calculate the tunneling currents. The calculated tunneling current is used to determine the boundary currents for the continuity solver at the tunneling path boundaries, and the continuity solver starts to solve again. This iteration terminates when the maximum change of E_{fn} and E_{fp} is smaller than the tolerance, and the E_{fn} and E_{fp} is provided to the Poisson solver.

Implementation of The Poisson Solver

For simplicity, the following discussions will use 1-D mathematical expressions. These equations can be easily extended to higher dimensional cases. The discretized Poisson equation including displacement continuity can be presented as

$$\frac{\epsilon_{i+0.5}(Ec_{i+1} - Ec_i) - \epsilon_{i-0.5}(Ec_i - Ec_{i-1})}{\Delta x^2} = \rho_i$$

If a contact point is met, the boundary Ec is a given value and can be moved to the RHS. That is,

$$\frac{-\epsilon_{\mathbf{N-0.5}}\mathbf{Ec_{N-1}} - \epsilon_{\mathbf{N-1.5}}(\mathbf{Ec_{N-1}} - \mathbf{Ec_{N-2}})}{\mathbf{\Delta x^2}} = \rho_{\mathbf{i}} - \frac{\epsilon_{\mathbf{N-0.5}}\mathbf{Ec_N}}{\mathbf{\Delta x^2}}$$

For the boundary points that are not externally contacted, it is assumed that the vertical electric field is zero, therefore the equations becomes

$$\frac{Ec_1 - Ec_0}{\Delta x} \rightarrow \frac{\epsilon_{1.5}(Ec_2 - Ec_1)}{\Delta x^2} = \rho_1$$

Considering the above cases, the sparse matrix representing the LHS of the Poisson equation can be created using the indices of all neighboring pairs and boundary contacts. The matrix is first constructed by the coordinate list (COO) and then transformed into the compressed sparse row (CSR) form.

Implementation of The Current Solver

Applying Scharffetter-Gummel Discretization

The continuity equation for the electron can be discretized by the Scharffetter-Gummel method. The current density between to mesh points can be expressed as

$$J_{i+0.5} = \frac{qD_n}{\Delta x}[\frac{t}{e^t - 1}n_{i+1} - \frac{-t}{e^{-t} - 1}n_i]$$

$$t = -(E_{c,i+1} - E_{c,i})/kT$$

Therefore, the divergence of electron currents can be expressed as an operator on the electron densities which is dependent to the electric fields. The rest is very similar

to the last section. Unless there exist contacts or tunneling currents, the vertical current densities are zero at the boundary points. That is, for those boundary points

$$\frac{-J_{n,0.5}}{q\Delta x} = -\frac{\partial n_0}{\partial t}$$

However, if tunneling currents are allowed, the equation has to be modified.

$$\frac{-J_{n,0.5}}{q\Delta x} = -\frac{\partial n_0}{\partial t} - \frac{J_{T,0}}{q\Delta x}$$

The calculation of the electron tunneling currents (J_{Tn}) will be introduced later.

Note that there are no well-defined fermi levels in the insulator regions, and the electron and hole densities are assumed to be zeros inside these meshes. Careful treatments on the operator must be taken for these regions to avoid a singular matrix, otherwise the operator is not solvable.

SRH Recombination Model

When the Shockley-Read-Hall model is taken into account, that carrier densities change rates with time becomes

$$\frac{\delta n_i}{\delta t} = \frac{\delta p_i}{\delta t} = -R^{SRH}$$

$$R^{\mathrm{SRH}} = \frac{n \cdot p - n_{i,e}^2}{\tau_p \cdot (n + n_1) + \tau_n \cdot (p + p_1)}.$$

The trap levels are assumed at the mid-gap in the forbidden band. Hence, the n and p continuity equations are arranged as

$$\frac{J_{n,i+0.5} - J_{n,i-0.5}}{q\Delta x} - \frac{np}{\tau_p(n + n_i) + \tau_n(p + n_i)} = -\frac{n_i^2}{\tau_p(n + n_i) + \tau_n(p + n_i)}$$

$$\frac{J_{p,i+0.5} - J_{p,i-0.5}}{q\Delta x} + \frac{np}{\tau_p(n + n_i) + \tau_n(p + n_i)} = \frac{n_i^2}{\tau_p(n + n_i) + \tau_n(p + n_i)}$$

By iteratively solving the n and p, the continuity equations with SRH recombination can be solved self-consistently. Figure 3 illustrates the difference of the solutions when the SRH model is turned on. When the SRH model is used, diffusion lengths for the low level injected electrons and holes are presented. The diffusion lengths are consistent to theoretically calculations.

Note that the effect of the recombination model is significant only when the scale of the simulated device is comparable with the diffusion lengths, which are typically of the order of 10~100μm.

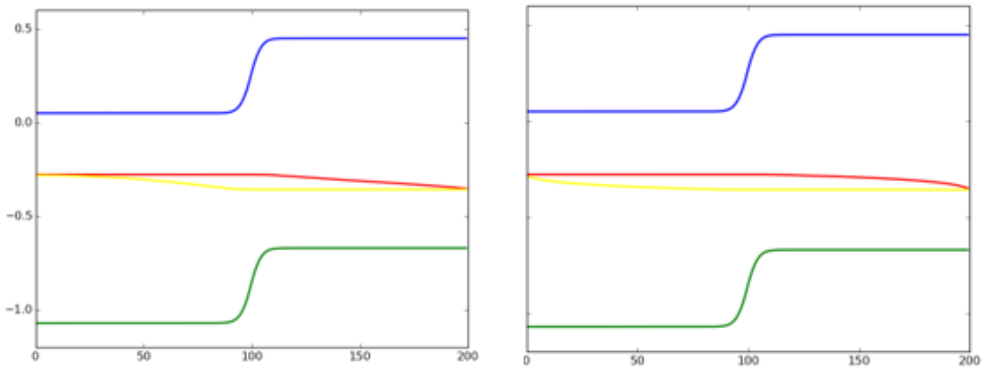

Figure 3. Visualized band diagrams of 1-D p-n junctions. The left is solved with the SRH model, while the right is not.

Calculation of Tunneling Current

The electron tunneling current is calculated for each 1-D tunneling path by the Tsu-Esaki formula and the WKB approximation for the transmission coefficient. Figure 4 are simple examples of the tunneling simulation results.

- The Tsu-Esaki Formula

The Tsu-Esaki formula is based on three assumptions.

I. Effective mass approximation
II. Parabolic bands
III. Parallel momentum conservation

Therefore, it can only be directly applied on tunneling paths between same materials. Only tunneling currents over oxide layers between same materials will be calculated by numerically integrating the following formula.

$$J = \frac{4\pi m_{\mathrm{eff}} q}{h^3} \int_{\mathcal{E}_{\mathrm{min}}}^{\mathcal{E}_{\mathrm{max}}} TC(\mathcal{E}_x) N(\mathcal{E}_x)\, \mathrm{d}\mathcal{E}_x \ ,$$

Where m_{eff} is the effective mass of the conducting of semiconducting material. The N and TC in the integrand are the "source function" and the "transmission coefficient".

$$N(\mathcal{E}_x) = \mathrm{k_B} T \ln\left(\frac{1 + \exp\left(-\frac{\mathcal{E}_x - \mathcal{E}_{\mathrm{f},1}}{\mathrm{k_B} T} \right)}{1 + \exp\left(-\frac{\mathcal{E}_x - \mathcal{E}_{\mathrm{f},2}}{\mathrm{k_B} T} \right)} \right) \ .$$

The electron fermi levels at the edges of the tunneling paths are used in the source function. The calculation of TC will be done with the WKB approximation introduced in the bellow.

- The WKB approximations

Assuming a constant electric field over the tunneling barrier, the WKB tunneling coefficients has analytical solutions. For direct tunneling,

$$TC(\mathcal{E}) = \exp\left(-4\frac{\sqrt{2t_{\text{diel}}}}{3\hbar \text{q} E_{\text{diel}}}\left((\text{q}\Phi - \mathcal{E})^{3/2} - (\text{q}\Phi_0 - \mathcal{E})^{3/2}\right)\right)$$

For Fowler–Nordheim tunneling,

$$TC(\mathcal{E}) = \exp\left(-4\frac{\sqrt{2m_{\text{diel}}}}{3\hbar \text{q} E_{\text{diel}}}(\text{q}\Phi - \mathcal{E})^{3/2}\right)$$

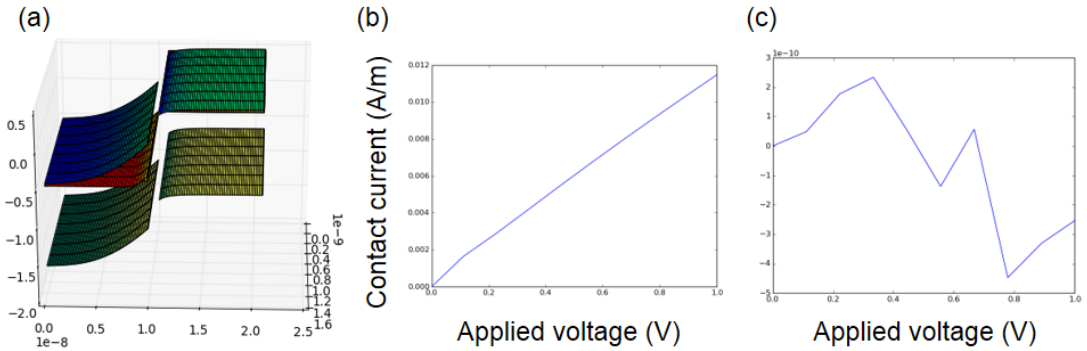

Figure 4. (a) The band diagram of two n+ silicon layers with 1nm SiO2 between them. The band diagram of the SiO2 is not presented for better readability. (b), (c) The contact currents as functions of applied voltages. The tunneling model is (c) and isn't applied (d), respectively.

Case Studies

I. A Planar Bulk MOSFET

In the simulation results in Fig. 5, The overdrive currents in good agreements with the theory. However, the subthreshold leakage is very large in this case.

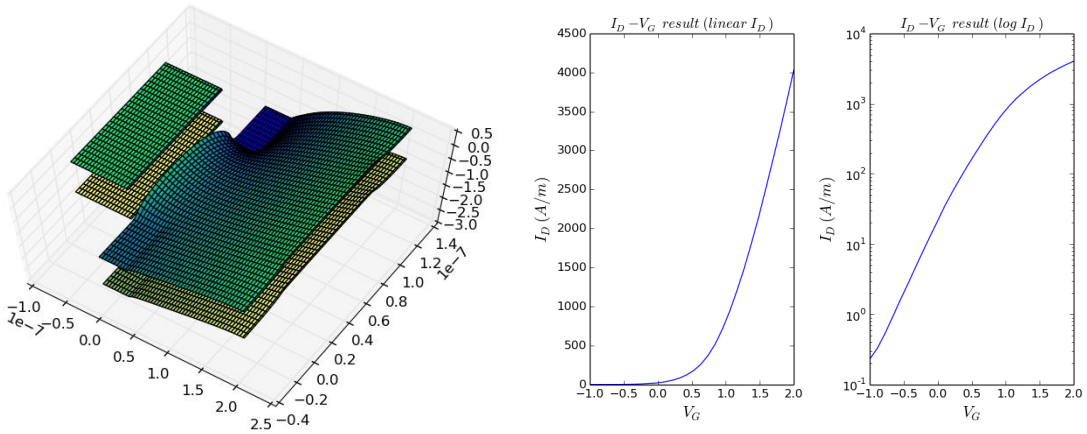

Figure 5. (a) The band diagram of a bulk MOSFET with n+ polysilicon gate. The energies of the SiO2 layer is not presented for better readability. (b), (c) The simulated I_D-V_G relations. The drain currents are in linear scale (c) and log scale (d), respectively.

The bad switching performance can be explained by the MOSFET short channel effect. As shown in Fig. 6, the source and drain depletion layer overlaps, and only the potential near the surface are well controlled by the gate voltage. This leads to a large source-drain leakage below the surface.

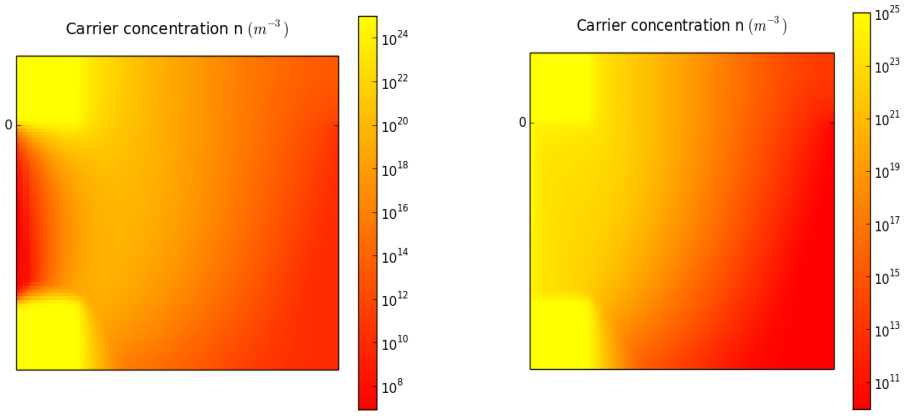

Figure 6. Electron densities in the silicon region of the MOSFET for different gate voltages. (a) V_G = -1V (b) V_G = 2V. The gate oxides are at the left of the regions.

II. A Planar Fully-depleted Silicon-On-Insulator (FDSOI) MOSFET

Different from the case above, the simulation results for FDSOI with thick BOX under it presents very good sub-threshold performance with subthreshold swing ~ 61mV/dec. The results are also in good agreement with the theoretical predictions.

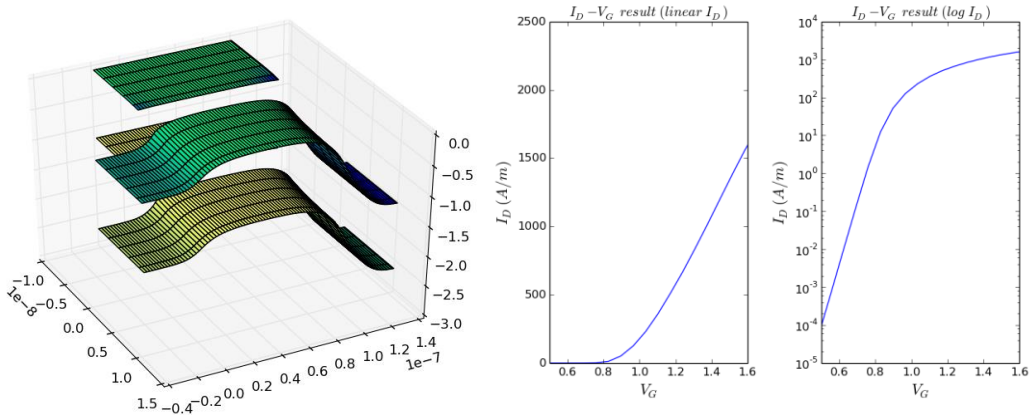

Figure 7. (a) The band diagram of the FDSOI MOSFET with n+ polysilicon gate when V_G=0.5V. The energies of the SiO2 layer is not presented for better readability. (b), (c) The simulated I_D-V_G relations. The drain currents are in linear scale (c) and log scale (d), respectively.