Element Edge Based Discretization for TCAD Device Simulation

Juan Sanchez [1,1] and Qiusong Chen [2]

[1]DEVSIM LLC
[2]Affiliation not available

November 8, 2023

Abstract

Technology computer-aided design (TCAD) semiconductor device simulators solve partial differential equations (PDE) using the finite volume method (FVM), or related methods. While this approach has been in use over several decades, its methods continue to be extended, and are still applicable for investigating novel devices. In this paper, we present an element edge based (EEB) FVM discretization approach suitable for capturing vector-field effects. Drawing from a 2D approach in the literature, we have extended this method to 3D. We implemented this method in a TCAD semiconductor device simulator, which uses a generalized PDE (GPDE) approach to simulate de- vices with the FVM. We describe how our EEB method is compatible with the GPDE approach, allowing the modeling of vector effects using scripting. This method is applied to solve polarization effects in a 3D ferro capacitor, and a 2D ferroelectric field-effect transistor. An example for field- dependent mobility in a 3D MOSFET is also presented.

Posted on 20 Apr 2020 — CC-BY 4.0 — https://doi.org/10.36227/techrxiv.14129081.v2 — This is a preprint. Version of Record available at https://doi.org/10.1109/TED.2021.3094776

Element Edge Based Discretization for TCAD Device Simulation

Juan E. Sanchez, *Senior Member, IEEE*, and Qiusong Chen

***Abstract*— Technology computer-aided design (TCAD) semiconductor device simulators solve partial differential equations (PDE) using the finite volume method (FVM), or related methods. While this approach has been in use over several decades, its methods continue to be extended, and are still applicable for investigating novel devices. In this paper, we present an element edge based (EEB) FVM discretization approach suitable for capturing vector-field effects. Drawing from a 2D approach in the literature, we have extended this method to 3D. We implemented this method in a TCAD semiconductor device simulator, which uses a generalized PDE (GPDE) approach to simulate devices with the FVM. We describe how our EEB method is compatible with the GPDE approach, allowing the modeling of vector effects using scripting. This method is applied to solve polarization effects in a 3D ferro capacitor, and a 2D ferroelectric field-effect transistor. An example for field-dependent mobility in a 3D MOSFET is also presented.**

***Index Terms*— Device Simulation, FeFET, MOSFET, Semiconductor, TCAD**

I. Introduction

TECHNOLOGY computer-aided design (TCAD) semiconductor device simulators solve continuum partial differential equations (PDEs) of the drift-diffusion model for semiconductors on a discretized mesh [1]. The finite volume method (FVM) is the most widely used approach for assembling the PDEs. In the TCAD device simulation literature, the FVM is referred as finite boxes or control volume and is closely related to finite differences [2].

The popularity of the FVM is due to the success of the Scharfetter-Gummel method (SGM) in calculating the electron and hole current densities from exponentially varying carrier densities [3], which exhibits superior numerical stability than other approaches [1], [4]. While the finite element method (FEM) presents advantages in some circumstances [5], [6], the lack of an equivalent to the SGM often leads to hybrid FVM/FEM approaches when solving the device equations with an otherwise FEM solver [7]–[9].

Standard simulation models fit well into a node based approach, as volume integration and surface integration of the PDEs are accounted for in the calculation of quantities at the mesh nodes, and at the edges connecting to adjacent mesh nodes.

Advanced simulation models use vector fields and require information from node quantities off of the edge. For mobility models, this requires knowledge of electric field parallel and normal to the direction of current flow [10]. For the ferroelectric effect, the vector components of electric field and polarization are considered [11], [12]. Nodal approaches have been proposed for evaluation of these models, which consider all adjacent nodes to the node of interest [13], [14]. These approaches consider a constant field over the entire control volume bounded by these nodes.

In contrast, an element edge based (EEB) approach has the advantage that the control volume is bounded by the sub volume along each edge of the element being considered. This makes it possible to consider phenomena occurring over smaller scales, such as for impact ionization models [15]. Laux used a 2D EEB approach to model generalized mobility [10] and impact ionization [15], where vector fields are calculated for each edge of a triangular element. In [16], this discretization was compared with other approaches for impact ionization, and it was found to be less susceptible to changes in mesh size than the other methods they considered. This was attributed to the EEB approach using a smaller effective control volume for each element edge. In this paper, we adopt this approach for 2D and extend it for tetrahedral elements in 3D.

Generalized PDE (GPDE) simulators use an equation description as input from the user [4], [6], [14], [17]. The goal is to allow the rapid development of new models and applications for the continuum based approach.

Many GPDE simulators do not consider vector effects, and equation assembly is often restricted to node or edge based quantities [4]. Other simulators do not completely model equation sensitivities to vector effects. Some provide problem specific operators, limiting the general purpose nature of their approach. One group compared 3 TCAD simulation approaches to model the ferroelectric capacitance effect [14]. They report better convergence for methods where more complete model sensitivities are considered, such as ensuring that the electric field is fully coupled with the field dependent polarization model.

In this paper, we describe an EEB approach to modeling vector-field effects in DEVSIM, an open source GPDE semiconductor device simulator, for both 2D and 3D device simulation [18]. We describe the approach in Section II. In

Q. Chen acknowledges financial support from Guizhou Provincial Science and Technology Projects (Grant No: QKHJC-ZK[2021]329)

J. E. Sanchez is with DEVSIM LLC, PO Box 50096, Austin, TX 78763, USA (e-mail: jsanchez@devsim.com)

Q. Chen, is with the Department of Materials Science, Fudan University, 220 Handan Road, Shanghai 200433, China. Q. Chen is also with the School of Physics and Electronic Science, Guizhou Education University, 115 Gaoxin Road, Wudang District, Guiyang, Guizhou , 550018 , China. (e-mail: chenqiusong@gmail.com)

Section III, we present simulation examples for hysteresis effects in a ferroelectric capacitor in 3D and in a ferroelectric field effect transistor (FeFET) in 2D. In addition, we present simulation results for transverse electric field effects on mobility in a 3D MOSFET.

II. Simulator Methods

A. Equation Assembly

1) Newton Method: The Newton method requires the evaluation of the model equations, as well as the derivatives with respect to the solution variable [1]. This results in a formulation:

$$\mathbf{J}\boldsymbol{\Delta x} = -\boldsymbol{F} \tag{1}$$

where $\mathbf{J}$ is the Jacobian, $\boldsymbol{\Delta x}$ is update to the solution vector, and $\boldsymbol{F}$ is referred to as the right-hand side (RHS) vector. As the method converges at each iteration, $|\boldsymbol{F}| \to 0$ and $|\boldsymbol{\Delta x}| \to 0$, to the limits of floating point precision. For TCAD simulation, the nonlinear nature of the PDEs requires a reasonable initial guess, and accurate derivatives, to get convergence or a good convergence rate.

The PDEs considered in this paper fit in the form

$$\boldsymbol{F}_a : \quad \nabla \cdot \boldsymbol{C}_a + \nabla \cdot \boldsymbol{A}_a + \boldsymbol{B}_a = \mathbf{0} \tag{2}$$

where $\boldsymbol{C}_a$ is an EEB quantity, $\boldsymbol{A}_a$ is an edge quantity, and $\boldsymbol{B}_a$ is a node quantity. The label $\boldsymbol{F}_a$ refers to PDE a being considered, as there are multiple simultaneously coupled PDEs in semiconductor simulation[1].

2) Node Assembly: Fig. 1 shows a mesh node in 2D surrounded by triangular elements. The $\boldsymbol{B}_a$ term is integrated over the `NodeVolume`. The RHS entry for this scalar quantity is then:

$$F_{a_i} \mathrel{+}= B_{a_i} \cdot \texttt{NodeVolume}_i \tag{3}$$

for each node i on the simulation mesh. In this, and subsequent equations, $+=$ represents addition of the term to an existing matrix or vector entry. Similarly, $-=$ represents subtraction.

For the Jacobian:

$$J_{a_i,x_j} \mathrel{+}= \frac{\partial B_{a_i}}{\partial x_j} \cdot \texttt{NodeVolume}_i \tag{4}$$

where x_i is a simulation variable on the same node. The Jacobian entry is placed in the row corresponding to equation a and the column corresponding to simulation variable x.

3) Edge Assembly : Fig. 2 shows the `EdgeCouple` over which flux is integrated for the $\boldsymbol{A}_a$ term in (2). The calculation of `EdgeCouple` is described in Section II-A.4.

For each edge between adjacent nodes $\langle i, j \rangle$

$$F_{a_i} \mathrel{+}= A_{a_{i,j}} \cdot \texttt{EdgeCouple}_{i,j} \tag{5}$$

$$F_{a_j} \mathrel{-}= A_{a_{i,j}} \cdot \texttt{EdgeCouple}_{i,j} \tag{6}$$

where $A_{a_{i,j}}$ is the vector quantity evaluated on a mesh edge, and $\texttt{EdgeCouple}_{i,j}$ is the cross section of the mesh edge.

[1]This paper does not consider our approach to time integration methods or boundary conditions.

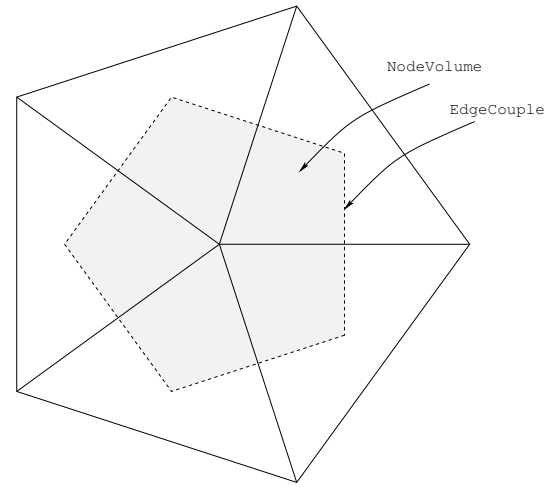

Fig. 1. The 2D mesh cell with an arbitrary number of triangle elements connected to node. The volume of a node is bounded by the perpendicular bisectors of each triangle [19].

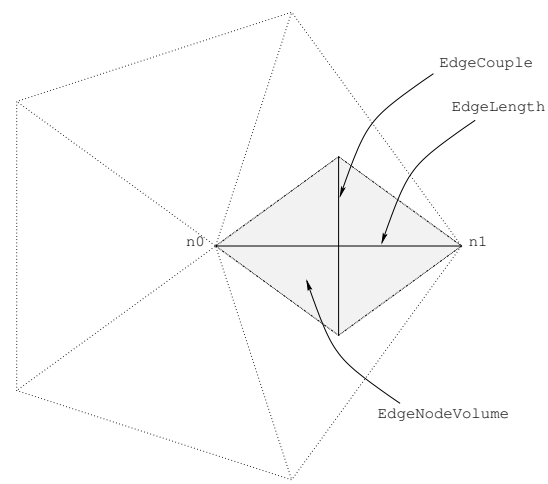

Fig. 2. The 2D mesh cell depicting how flux is integrated along each edge [19]. The `EdgeCouple` is the length of the perpendicular bisectors for the 2 triangles along the edge connecting nodes $\langle \texttt{n0}, \texttt{n1} \rangle$.

The Jacobian entries are:

$$J_{a_i,x_k} \mathrel{+}= \frac{\partial A_{a_{i,j}}}{\partial x_k} \cdot \texttt{EdgeCouple}_{i,j} \tag{7}$$

$$J_{a_j,x_k} \mathrel{-}= \frac{\partial A_{a_{i,j}}}{\partial x_k} \cdot \texttt{EdgeCouple}_{i,j} \tag{8}$$

where x_k is the simulation variable on one of the two nodes on the edge.

4) Element Edge Assembly: We consider the $\boldsymbol{C}_a$ contribution to (2), which is EEB, and is dependent on variables on all element nodes.

Fig. 3 shows a 2D representation of the volume and area over which the integration occurs. In 2D, the `ElementEdgeCouple` on each edge is from the triangle circumcenter to the center of the edge. Fig. 4 shows the element edge volume in 3D, where the `ElementEdgeCouple` is from the circumcenter of the tetrahedral element, to the circumcenters of the element triangles connected to the element edge, to the center of the element edge. In both 2D and 3D, the `EdgeCouple` in Section II-A.3 is calculated by summing the

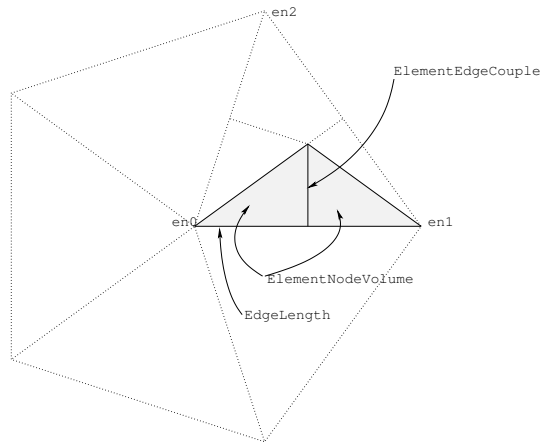

Fig. 3. The mesh cell in 2D. Each triangle is subdivided into 3 sub volumes. The labels for nodes $\langle \texttt{en0}, \texttt{en1}, \texttt{en2} \rangle$ are specific to the element edge in the sub element bounded by the solid line [19].

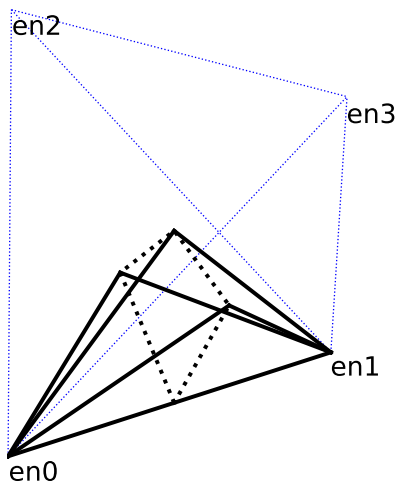

Fig. 4. The element cell in 3D. The tetrahedron is subdivided into 6 subvolumes along each edge. The element edge of interest has the nodes labeled $\langle \texttt{en0}, \texttt{en1} \rangle$. The other nodes are labeled $\langle \texttt{en2}, \texttt{en3} \rangle$.

$\texttt{ElementEdgeCouple}$ for all elements connected to the edge being considered.

Similar to (5)–(6), the RHS contribution for $\boldsymbol{C}_a$ is:

$$F_{a_i} \mathrel{+}= C^t_{a_{i,j}} \cdot \texttt{ElementEdgeCouple}^t_{i,j} \tag{9}$$

$$F_{a_j} \mathrel{-}= C^t_{a_{i,j}} \cdot \texttt{ElementEdgeCouple}^t_{i,j} \tag{10}$$

where t is the index of the element containing the edge.

Similar to (7)–(8), the Jacobian entries are:

$$J_{a_i,x_k} \mathrel{+}= \frac{\partial C^t_{a_{i,j}}}{\partial x_k} \cdot \texttt{ElementEdgeCouple}^t_{i,j} \tag{11}$$

$$J_{a_j,x_k} \mathrel{-}= \frac{\partial C^t_{a_{i,j}}}{\partial x_k} \cdot \texttt{ElementEdgeCouple}^t_{i,j} \tag{12}$$

where x_k is a simulation variable on one of the nodes on the element. In 2D, this is the 2 nodes, $\langle \texttt{en0}, \texttt{en1} \rangle$, on the element edge being considered, and a third node, $\texttt{en2}$, on a triangular element, as shown in Fig. 3. In 3D, there are 2 nodes on the each element edge, $\langle \texttt{en0}, \texttt{en1} \rangle$, as well as 2 additional nodes on the tetrahedral element $\langle \texttt{en2}, \texttt{en3} \rangle$, as shown in Fig. 4.

5) Edge Volume Assembly : It is also possible to do a volume integration using either the edge based or EEB assembly. This is useful for models such as the density gradient model [20] or impact ionization model [15]. In equations, (9)–(12), the $\texttt{ElementNodeVolume}$ in Fig. 3 is substituted for $\texttt{ElementEdgeCouple}$ and the assembly is only done with the $+=$ operation.

B. Element Edge Based Fields

1) Calculating the Vector Field: To calculate a vector field on an edge of a tetrahedral element, we begin by calculating the components along each mesh edge in the device region. For the case of electric field, this is

$$E_{i,j} = (\psi_i - \psi_j) / \texttt{EdgeLength}_{i,j} \tag{13}$$

where ψ_i and ψ_j are the potentials at nodes $\langle i, j \rangle$, and $\texttt{EdgeLength}_{i,j}$ is the distance between them, as shown in Fig. 2. The derivatives with respect to the node potentials are:

$$\frac{\partial E_{i,j}}{\partial \psi_i} = 1/\texttt{EdgeLength}_{i,j} \qquad \frac{\partial E_{i,j}}{\partial \psi_j} = -1/\texttt{EdgeLength}_{i,j} \tag{14}$$

In Fig. 5, we show how these scalar fields are used to calculate $\boldsymbol{E}_{\boldsymbol{t_0}}$, the vector field based on all edges connect to node 0. The electric field on each edge is assumed to be

$$E_{i,j} = \boldsymbol{s}_{i,j} \cdot \boldsymbol{E}_{\boldsymbol{t_0}} \tag{15}$$

where $\boldsymbol{E}_{t_0}$ is the field over the entire element, and $\boldsymbol{s}_{i,j}$ is the unit vector along the edge connecting nodes $\langle i, j \rangle$.

For tetrahedra with nodes $\langle 0, 1, 2, 3 \rangle$, we use the x, y, z components of the unit vectors to calculate the electric field for all edges connected to node 0

$$\begin{pmatrix} s^x_{0,1} & s^y_{0,1} & s^z_{0,1} \\ s^y_{0,2} & s^y_{0,2} & s^z_{0,2} \\ s^z_{0,3} & s^y_{0,3} & s^z_{0,3} \end{pmatrix} \begin{pmatrix} E^x_{t0} \\ E^y_{t0} \\ E^z_{t0} \end{pmatrix} = \begin{pmatrix} E_{0,1} \\ E_{0,2} \\ E_{0,3} \end{pmatrix} \tag{16}$$

which can be written more compactly as

$$\mathbf{S}_{t_0} \cdot \boldsymbol{E}_{t_0} = \begin{pmatrix} E_{0,1} & E_{0,2} & E_{0,3} \end{pmatrix}^T \tag{17}$$

so that $\boldsymbol{E}_{\boldsymbol{t_0}}$ is found by solving this small linear system.

The derivatives with respect to the node potentials are then

$$\mathbf{S}_{\mathbf{t_0}} \cdot \frac{\partial \boldsymbol{E}_{\boldsymbol{t_0}}}{\partial \psi_0} = \begin{pmatrix} \frac{\partial E_{0,1}}{\partial \psi_0} & \frac{\partial E_{0,2}}{\partial \psi_0} & \frac{\partial E_{0,3}}{\partial \psi_0} \end{pmatrix}^T \tag{18}$$

$$\mathbf{S}_{\mathbf{t_0}} \cdot \frac{\partial \boldsymbol{E}_{\boldsymbol{t_0}}}{\partial \psi_1} = \begin{pmatrix} \frac{\partial E_{0,1}}{\partial \psi_1} & 0 & 0 \end{pmatrix}^T \tag{19}$$

$$\mathbf{S}_{\mathbf{t_0}} \cdot \frac{\partial \boldsymbol{E}_{\boldsymbol{t_0}}}{\partial \psi_2} = \begin{pmatrix} 0 & \frac{\partial E_{0,2}}{\partial \psi_2} & 0 \end{pmatrix}^T \tag{20}$$

$$\mathbf{S}_{\mathbf{t_0}} \cdot \frac{\partial \boldsymbol{E}_{\boldsymbol{t_0}}}{\partial \psi_3} = \begin{pmatrix} 0 & 0 & \frac{\partial E_{0,3}}{\partial \psi_3} \end{pmatrix}^T \tag{21}$$

In 3D, we treat the field for the edge between nodes $\langle 0, 1 \rangle$ using an average:

$$\boldsymbol{E}_{t_{0,1}} = 0.5 \cdot (\boldsymbol{E}_{t_0} + \boldsymbol{E}_{t_1}) \tag{22}$$

where $\boldsymbol{E}_{t_1}$ is calculated by considering node 1 as the node of interest in (16)–(21).

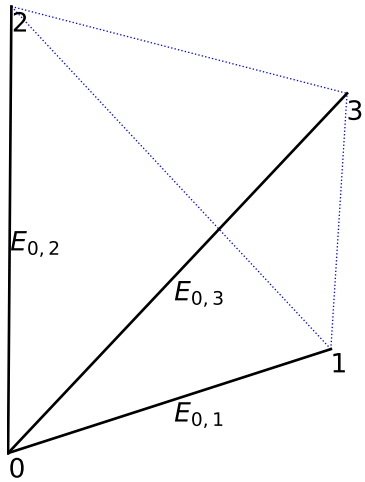

Fig. 5. Depiction on how vector field is calculated from the scalar fields calculated along the $\mathbf{3}$ edges connected to node $\mathbf{0}$.

The derivative with respect to the node potential is then:

$$\frac{\partial \boldsymbol{E}_{t_{0,1}}}{\partial \psi_k} = 0.5 \cdot \left(\frac{\partial \boldsymbol{E}_{t_0}}{\partial \psi_k} + \frac{\partial \boldsymbol{E}_{t_1}}{\partial \psi_k} \right) \tag{23}$$

where k refers to one of the 4 nodes on the element.

We assume that the average employed in (22)–(23) is appropriate for creating vectors from edge based models. It can be shown for electric field that $\boldsymbol{E}_{t_0} = \boldsymbol{E}_{t_1}$, so the average is valid. For cases like current density, which does not have this property, this assumption appears valid for the simulations in Section III-C.

In 2D, we use an edge average from [10], [15], based on a weighting the length of perpendicular bisectors of the triangular elements. Our simulator also provides the ability to customize the weighting.

Once the vector fields are attained, they are used to calculate the components for the element edge assembly in (9)–(12), by first converting to a scalar value, and then adding to $\mathbf{J}$ and $\boldsymbol{F}$. Scalar models defined as an edge or element edge model, can also be used in calculations with the vector field components. This will be described in Section II-C, where the approach is described for the polarization vector in ferroelectric materials, as well as for the transverse electric field used for mobility models.

2) Computation: In practice, the $\boldsymbol{S}_{t_0}$ term in (17) is calculated once for each node of a tetrahedral element and is assigned as $\boldsymbol{S}_{t_0}$ or $\boldsymbol{S}_{t_1}$, based on the element edge of interest. Since only geometric information is stored, the matrix factorization is done once at the beginning of the simulation, and is solved for all EEB field calculations across all iterations of the simulation. It is possible to take advantage of high performance math routines, so that the matrix factorization and back substitution is done in an efficient manner [21].

3) Limitations on Element Type : In our approach, we have restricted the method to triangular elements in 2D and tetrahedral elements in 3D. The original finite box methods restricted themselves to box and cube elements [1], and were later expanded [2].

In 3D, it is necessary that the centers of the tetrahedra circumspheres are inside the element, so that the volume and surface integrals are calculated correctly [2]. In 2D, the circumcenter of triangular elements should also be inside the circumcircle, or equivalently not obtuse [1]. While there are methods to accommodate elements violating this condition, their presence often leads to less accurate results and convergence difficulties.

4) Derivative Accuracy: Good convergence behavior is often dependent on having accurate derivatives for the simulation models [1]. We demonstrate this with the simulation example in Section III-A. Accurate derivatives are also important small-signal analysis, and the impedance field method [22]–[24].

C. Generalized PDE Approach

In DEVSIM, models are implemented as symbolic expressions, and they are assembled and solved [17], [19]. Symbolic differentiation is employed to get the required derivatives. The simulator accepts equations in the form of:

Node models on nodes (Section II-A.2)
Edge models on edges (Section II-A.3)
Element edge models on element edges (Section II-A.4)

The GPDE equations are input by calling commands from a Python script [25]. The simulator is a module loaded by the interpreter, mostly written in C++ [17].

1) Ferroelectric Model: A ferroelectric model may be implemented directly, using scripting. For a ferroelectric insulator, the equation is:

$$\nabla \cdot (\varepsilon \boldsymbol{E} + \boldsymbol{P}) = 0 \tag{24}$$

where $\boldsymbol{P}$ is the polarization.

We first start with the edge model for electric field and its derivatives

```
EF,  EF:psi@n0,  EF:psi@n1
```

where `EF` is evaluated using an expression for (13) and the rest using (14) for nodes $\langle 0, 1 \rangle$, respectively. The user defines the model and its derivatives using the symbolic math engine in the simulator.

From the edge model and its derivatives, the `element_from_edge_model` command is used to create element edge models as:

```
EF_x, EF_y, EF_z
```

for each vector component using (22) and their derivatives using (23)

```
EF_x:psi@en0, EF_y:psi@en0 EF_z:psi@en0
EF_x:psi@en1, EF_y:psi@en1 EF_z:psi@en1
EF_x:psi@en2, EF_y:psi@en2 EF_z:psi@en2
EF_x:psi@en3, EF_y:psi@en3 EF_z:psi@en3
```

The first term represents the derivative of `EF_x` with respect to the potential at node 0, and the other terms follow in a similar fashion.

During evaluation, indexes $\langle \mathtt{en0}, \mathtt{en1} \rangle$ reference nodes on the element edge and $\langle \mathtt{en2}, \mathtt{en3} \rangle$ reference the other element nodes on the tetrahedron. The simulator tracks the element edge indexes on a per element edge basis, so that the same PDE expression will work for each element edge being considered.

The simulator then evaluates the symbolic expression for (24) using

```
(PX+Permittivity*EF_x)*unitx +
(PY+Permittivity*EF_y)*unity +
(PZ+Permittivity*EF_z)*unitz
```

where `unitx`, `unity`, and `unitz` are the unit vector components along the element edge and `Permittivity` is a parameter. The terms `PX`, `PY`, and `PZ` are components which computed in using expressions involving the current and previous solutions for the electric field. We utilized a widely accepted empirical function of hyperbolic tangent to simulate the hysteresis loop [11], [12], [26]. Scripting was used to generate these terms for all of the x, y, and z components and the derivatives of the polarization vector and electric fields.

2) Field Dependent Mobility: To calculate the surface mobility for the simulations in Section III-C, we first calculate the electric field parallel to current flow as:

$$E^{\parallel}_{t_{0,1}} = \boldsymbol{j}_{t_{0,1}} \cdot \boldsymbol{E}_{t_{0,1}} / |\boldsymbol{j}_{t_{0,1}}| \tag{25}$$

where $\boldsymbol{j}_{t_{i,j}}$ is the current density vector calculated using

$$\mathbf{S}_{t_0} \cdot \boldsymbol{j}_{t_0} = \begin{pmatrix} j_{0,1} & j_{0,2} & j_{0,3} \end{pmatrix}^T \tag{26}$$

where $j_{0,1}$, $j_{0,2}$, and $j_{0,3}$ are edge current densities using a suitable low field mobility, and then averaged onto the element edge using the same average as (22). The vector $\boldsymbol{j}_{t_{0,1}}$ is used to find the direction of current flow, and is not the final current density. The transverse electric field is then calculated using:

$$|E^{\perp}_{t_{i,j}}| = \sqrt{|\boldsymbol{E}_{t_{0,1}}|^2 - |\boldsymbol{E}^{\parallel}{}_{t_{0,1}}|^2} \tag{27}$$

The expressions for $|E^{\perp}_{t_{i,j}}|$ are then used to calculate the surface components of the mobility model [27]. The bulk and surface mobility components are then averaged together, and the velocity saturation model is applied to get the EEB mobility. The EEB current densities are then integrated using (9)–(12), having been calculated using the SGM applied on each element edge.

III. Simulation Examples

A. Ferrocapacitor Simulation

TetGen [28] was used to generate a tetrahedral mesh of a cube and contacts on the top and bottom. Not all of the elements met the requirements mentioned in Section II-B.3. A script was written to mark bad elements for refinement, and it was possible to achieve a mesh meeting the simulator's requirements.

The mesh and physics are loaded via scripts into the simulator. Fig. 6 shows the compensated charge versus the applied voltage for a ferroelectric capacitor. From the initial zero bias condition, the voltage is ramped over the range between $\pm 15\ V$. The hysteresis in the charge due to the polarization is apparent in the figure.

As a test of the accuracy of our approach, we repeated the simulations with derivative terms with respect to nodes $\langle \mathtt{en2}, \mathtt{en3} \rangle$ set to 0. This emulates the condition in GPDE simulators which evaluate vector effects over all adjacent nodes, but may restrict their derivative information on edges [4], [14].

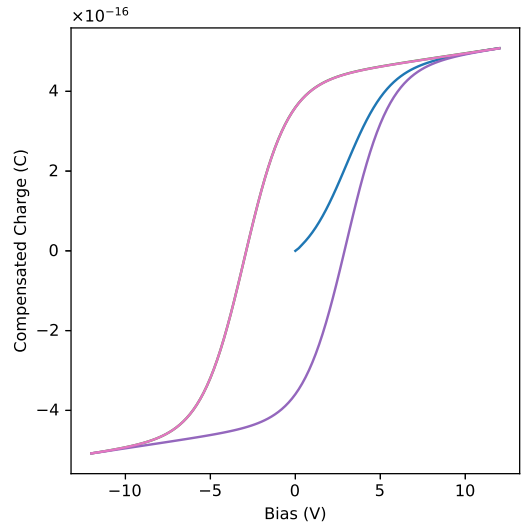

Fig. 6. The ferro capacitor 3D simulation.

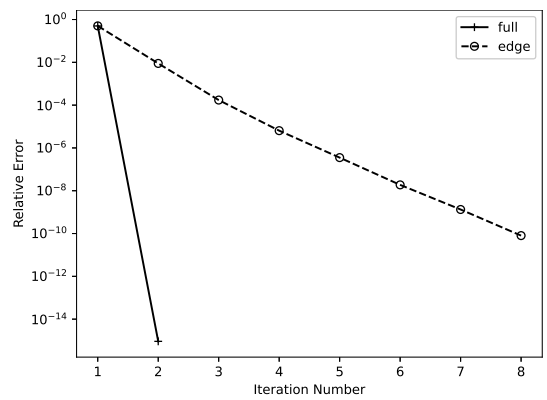

Fig. 7. The relative error versus iteration number. The solid line is for derivatives for all nodes on the element. The dashed line is when only nodes on the edge are considered.

The primary convergence criteria in our simulator is relative error, which is calculated at each node i as:

$$r_i^m = \frac{|\psi_i^m - \psi_i^{m-1}|}{|\psi_i^{m-1}| + 10^{-10}} \tag{28}$$

where m is the iteration number, and ψ_i^m is the new solution value after the update. The relative error is taken at the node with the largest value of r.

Fig. 7 shows relative error versus iteration number for one of the bias points. With all derivatives accounted for, it takes 2 iterations to get a relative error near 10^{-15}, which is the limit of accuracy for double precision floating point arithmetic. This number of iterations is expected, since the equation derivatives are constant with respect to bias. When the off edge derivatives are neglected, the iteration count is higher to to attain a relative error within 10^{-10}.

B. FeFET Simulation

We simulated a 2D p-type FeFET with a top contact/bottom gate architecture. Ohmic contacts were set at the source and drain electrodes. A triangular mesh was generated using

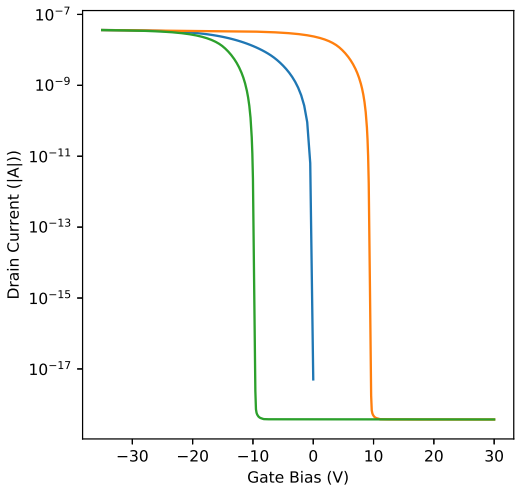

Fig. 8. The p-type FeFET 2D simulation.

Gmsh [29]. Extended precision math mode was enabled in the equation assembly to prevent numerical noise in the off state current results. Fig. 8 shows the drain current versus gate voltage. The simulation details and an analysis of the switching processes of FeFET with respect to its structure will be presented in [30]. It is important to note that when the derivative off the edge on node `en2` was set to 0, the simulation failed before the sweep of biases could finish. Therefore, the full set of derivatives we implemented using the EEB approach was necessary to run the simulations.

C. MOSFET Simulation

We simulated a 3D MOSFET, based on a doping profile and 2D structure for a 90 nm n-channel MOSFET archived in [31]. For the simulations, we created a 3D mesh using Cubit [32]. The structure is shown in Fig. 9.

Using the scripting interface, we implemented the Poisson, electron-continuity, and hole-continuity equations, with SRH recombination [1]. We implemented the mobility model from [27], with surface mobility components dependent on the electric field normal to current flow.

After refinement, the bulk region had 270,693 tetrahedra and 50,446 nodes. Solving this region with the oxide regions, nitride regions, and a polysilicon gate electrode, the simulation matrix had 166,225 rows for the fully coupled drift-diffusion simulation.

We then compared the mobility models using these three cases:

- Concentration dependent bulk mobility model
- Bulk mobility with velocity saturation
- Bulk and surface mobility model with velocity saturation

where the normal electric field for surface mobility was implemented using (27).

Figure 10 show a contour plot of the velocity saturated mobility with both the bulk and surface models enabled. Figure 11 shows I_{DS} versus V_{GS} for the three cases. The I_{DS} versus V_{GS} dependence is shown in Fig. 12. Consistent with the theory, the $E^{\perp}$ dependent mobility results in a lowering of the drain current.

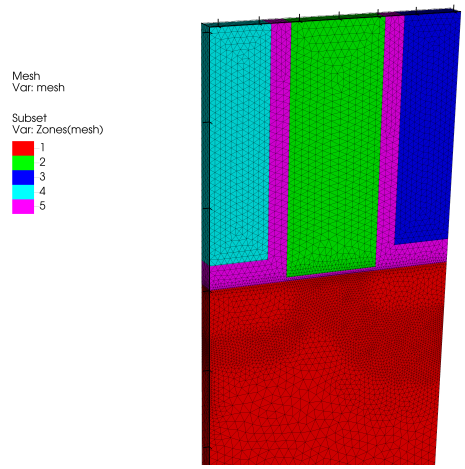

Fig. 9. The 90 nm 3D MOSFET. The polysilicon gate (2) is surrounded by oxide (5) and two nitride regions (3, 4). The bulk region (1) has a 120 nm drawn gate length. The source and drain contacts are both 50 nm underneath the nitride regions. A body contact was placed on the bottom of the 60 nm silicon region. The oxide thickness is 4.9 nm and the device is 25nm thick.

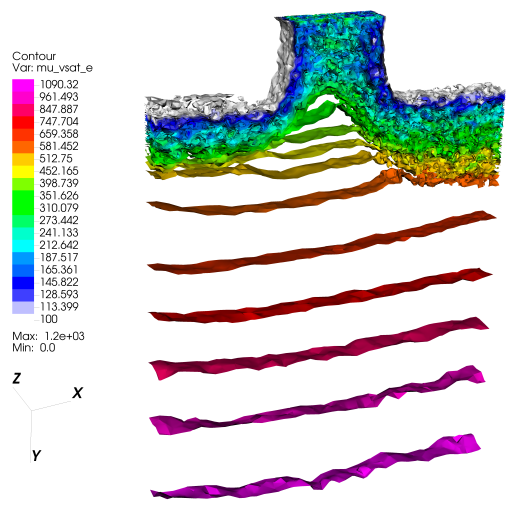

Fig. 10. Contour plot of the bulk and surface electron mobility with velocity saturation for a bias of V_{GS} = 1 V, V_{DS} = 1 V and V_{SB} = 0.

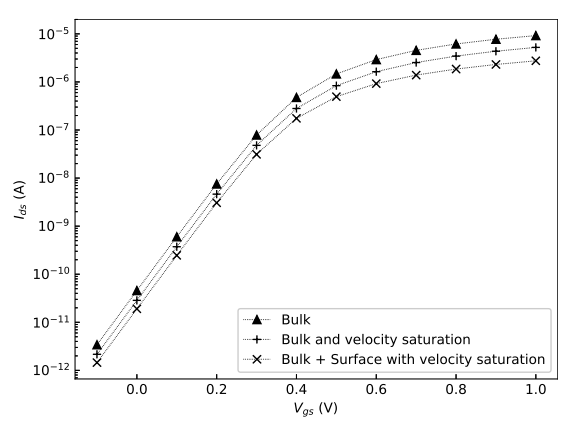

Fig. 11. I_D versus V_{GS} for V_{DS} = 0.1 V and V_{SB} = 0.

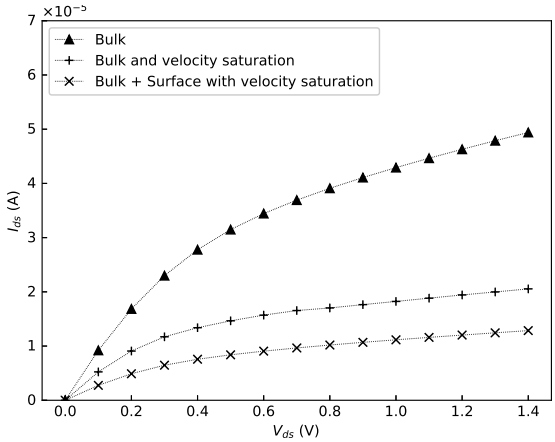

Fig. 12. I_D versus V_{DS} for V_{GS} = 1 V. and V_{SB} = 0

Acknowledgment

J. E. Sanchez thanks E. J. Chiles, M. E. Sanchez, and C. M. Gotwalt for their assistance in the editing this manuscript. The authors thank the anonymous reviewers for their feedback.

References

[1] S. Selberherr, *Analysis and simulation of semiconductor devices*. NY: Springer-Verlag, 1984.

[2] M. R. Pinto, "Comprehensive semiconductor device simulation for silicon ULSI," Ph.D. dissertation, Stanford University, 1990.

[3] D. L. Scharfetter and H. K. Gummel, "Large-signal analysis of a silicon Read diode oscillator," *IEEE Trans Electron Devices*, vol. ED-16, no. 1, pp. 64–77, Jan. 1969, 10.1109/T-ED.1969.16566.

[4] K. M. Kramer and W. N. G. Hitchon, *Semiconductor Devices: A Simulation Approach*. Up Saddle River, NJ: Prentice Hall PTR, 1997.

[5] D. J. Cummings, M. E. Law, S. Cea, and T. Linton, "Comparison of discretization methods for device simulation," in *2009 Int Conf SISPAD*, 2009, pp. 1–4, DOI: 10.1109/SISPAD.2009.5290236.

[6] E. Patrick, N. Rowsey, and M. E. Law, "Total dose radiation damage: A simulation framework," *IEEE Trans Nucl Sci*, vol. 62, no. 4, pp. 1650–1657, 2015, DOI: 10.1109/TNS.2015.2425226.

[7] P. Bochev, K. Peterson, and X. Gao, "A new control volume finite element method for the stable and accurate solution of the drift-diffusion equations on general unstructured grids," *Comput Methods Appl Mech Eng*, vol. 254, pp. 126–145, 02 2013, DOI: 10.1016/j.cma.2012.10.009.

[8] COMSOL, "Analyze semiconductor devices at the fundamental level with the semiconductor module," last accessed 09/15/2020. [Online]. Available: https://www.comsol.com/semiconductor-module

[9] L. Chen and H. Bagci, "Steady-state simulation of semiconductor devices using discontinuous Galerkin methods," *IEEE Access*, vol. 8, pp. 16 203–16 215, 2020, DOI: 10.1109/ACCESS.2020.2967125.

[10] S. E. Laux and R. G. Byrnes, "Semiconductor device simulation using generalized mobility models," *IBM J. Res. Dev.*, vol. 29, no. 3, pp. 289–301, May 1985, DOI: 10.1147/rd.293.0289.

[11] S. L. Miller, R. D. Nasby, J. R. Schwank, M. S. Rodgers, and P. V. Dressendorfer, "Device modeling of ferroelectric capacitors," *J Appl Phys*, vol. 68, no. 12, pp. 6463–6471, 1990, DOI: 10.1063/1.346845.

[12] M. Ghittorelli, T. Lenz, H. Sharifi Dehsari, D. Zhao, K. Asadi, P. W. M. Blom, Z. M. Kovács-Vajna, D. M. de Leeuw, and F. Torricelli, "Quantum tunnelling and charge accumulation in organic ferroelectric memory diodes," *Nat Commun*, vol. 8, no. 1, p. 15741, 2017, DOI: 10.1038/ncomms15841.

[13] Z. Yu, D. W. Yergeau, and R. W. Dutton, "Algorithm for evaluating nodal vector quantities in device simulation and its applications to modeling quantum mechanical effects in sub-50nm MOSFETs," in *Int Symp VLSI Tech, Syst App*, 2003, pp. 261–264, DOI: 10.1109/VTSA.2003.1252603.

[14] T. Ikegami, K. Fukuda, and J. Hattori, "Implementation of automatic differentiation to Python-based semiconductor device simulator," in *2019 Int Conf SISPAD*, 2019, pp. 1–4, DOI: 10.1109/SISPAD.2019.8870377.

[15] S. E. Laux and B. M. Grossman, "A general control-volume formulation for modeling impact ionization in semiconductor transport," *IEEE Trans Electron Devices*, vol. 32, no. 10, pp. 2076–2082, 1985, DOI: 10.1109/T-ED.1985.22241.

[16] O. Triebl and T. Grasser, "Investigation of vector discretization schemes for box volume methods," in *Tech Proc 2007 NSTI Nanotechnology Conf and Trade Show, Vol 3*, 2007, pp. 61–64.

[17] J. E. Sanchez, "Semiconductor device simulation using DEVSIM," in *Open Source TCAD/EDA for Compact Modeling*, W. Grabinski and D. Tomaszewski, Eds. New York: Springer, 2021.

[18] DEVSIM LLC, "DEVSIM TCAD semiconductor device simulator," Available: https://devsim.org.

[19] ——, "DEVSIM manual," Available: https://devsim.net, DOI: 10.5281/zenodo.4583208.

[20] A. Wettstein, O. Penzin, and E. Lyumkis, "Integration of the density gradient model into a general purpose device simulator," *VLSI Design*, vol. 15, no. 4, pp. 751–759, 2002, DOI: 10.1080/1065514021000012363.

[21] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, 3rd ed. Philadelphia, PA: SIAM, 1999.

[22] S. E. Laux, "Techniques for small-signal analysis of semiconductor devices," *IEEE Trans Electron Devices*, vol. 32, no. 10, pp. 2028–2037, 1985, DOI: 10.1109/T-ED.1985.22235.

[23] F. Bonani, G. Ghione, M. R. Pinto, and R. K. Smith, "An efficient approach to noise analysis through multidimensional physics-based models," *IEEE Trans Electron Devices*, vol. 45, no. 1, pp. 261–269, 1998, DOI: 10.1109/16.658840.

[24] K. El Sayed, A. Wettstein, S. D. Simeonov, E. Lyumkis, and B. Polsky, "Investigation of the statistical variability of static noise margins of sram cells using the statistical impedance field method," *IEEE Transactions on Electron Devices*, vol. 59, no. 6, pp. 1738–1744, 2012, DOI: 10.1109/TED.2012.2189860.

[25] Python Software Foundation, "Python," https://python.org.

[26] S. L. Miller and P. J. McWhorter, "Physics of the ferroelectric nonvolatile memory field effect transistor," *J Appl Phys*, vol. 72, no. 12, pp. 5999–6010, 1992, DOI: 10.1063/1.351910.

[27] M. N. Darwish, J. L. Lentz, M. R. Pinto, P. M. Zeitzoff, T. J. Krutsick, and H. H. Vuong, "An improved electron and hole mobility model for general purpose device simulation," *IEEE Trans Electron Devices*, vol. 44, no. 9, pp. 1529–1538, 1997, DOI: 10.1109/16.622611.

[28] H. Si, "TetGen, a Delaunay-based quality tetrahedral mesh generator," *ACM Trans. Math. Softw.*, vol. 41, no. 2, Feb. 2015, DOI: 10.1145/2629697.

[29] C. Geuzaine and J.-F. Remacle, "Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities," *Int J Numer Methods Eng*, vol. 79, pp. 1309–1331, 2009, DOI: 10.1002/nme.2579.

[30] Q. Chen, D. Lin, Q. Wang, J. Yang, J. Sanchez, and G. Zhu, "An investigation of the switching processes of ferroelectric field-effect transistors using 2D simulation," *submitted for publication*.

[31] D. A. Antoniadis, I. J. Djomehri, K. M. Jackson, and S. Miller, "Well-Tempered bulk-Si NMOSFET device home page," Nov. 2001, DOI: 10.5281/zenodo.4672795.

[32] Coreform LLC, "Coreform Cubit (Version 2021.3)," Available: https://coreform.com.