Implementation of Automatic Differentiation to Python-based Semiconductor Device Simulator

Tsutomu Ikegami
National Institute of Advanced Industrial Science and Technology (AIST)
Tsukuba, Japan
t-ikegami@aist.go.jp

Koichi Fukuda
National Institute of Advanced Industrial Science and Technology (AIST)
Tsukuba, Japan
fukuda.koichi@aist.go.jp

Junichi Hattori
National Institute of Advanced Industrial Science and Technology (AIST)
Tsukuba, Japan
j.hattori@aist.go.jp

***Abstract*—A Python-based device simulator named Impulse TCAD was developed. The simulator is built on top of a nonlinear finite volume method (FVM) solver. To describe physical behavior of non-standard materials, both device properties and their dominant equations can be customized. The given FVM equations are solved by the Newton method, where required derivatives of the equations are derived automatically by using an automatic differentiation technique. As a demonstration, a steady state analysis of the negative capacitance field effect transistors with ferroelectric materials is selected, where the coupled Poisson and Devonshire equations are implemented in several different ways.**

***Keywords*—TCAD, device simulation, automatic differentiation, Python, negative capacitance**

I. Introduction

Reducing the power consumption of semiconductor devices is an urgent issue in both edge devices of the internet of things and server machines in data centers. To overcome the physical limitation inherent in conventional silicon devices, many researchers are trying to utilize singular physical phenomena like quantum tunneling and novel materials like ferroelectrics. On the other hand, semiconductor device simulation has long been a standard tool to realize the rigorous silicon roadmap [1]–[3]. Therefore, for the development of the next-generation semiconductor devices, non-standard physical models have to be implemented in the device simulators. A new device simulator named Impulse TCAD [4], [5] was thus developed to support such studies on the new physical models. Impulse TCAD is written mainly in the Python language, which is also used to describe the run script to control the simulation pathway. In Impulse TCAD, users can declare device properties that characterize target materials, and define dominant equations that describe their physical behaviors. It is even possible to intervene in the iterative solver of the equations, which helps several difficult cases to converge. Because all these customizations are managed in the Python domain, a special customization for a specific project can even be possible from the run script, without modifying the main body of the simulator. Modifications are also reflected instantly without the complicated compile and link process, which accelerates the development process of new models. In the following, the basic equations used in Impulse TCAD are introduced along with the background mechanism, followed by a representative example. Applying this to the negative capacitance field effect transistors is also shown as a demonstration.

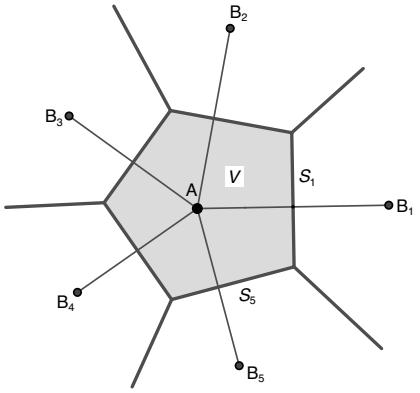

Fig. 1. Control volumes in FVM. Each material region is divided into control volumes, which are designated by node points (A and B_i).

II. Impulse TCAD

A. Finite Volume Method

Impulse TCAD is a semiconductor device simulator based on the finite volume method (FVM). Device structures are represented in 3D unstructured meshes, on which the vertex-centered non-linear FVM is implemented. To setup FVM, each material region is divided into tiny control volumes (CV) as shown in Fig. 1. Device properties in CV, such as electric potential and electron density, are represented on "node" points (A and B_i). Similarly, properties on the boundary surface (or on the "edge") between neighbouring CVs A and B_i are dependent only on the properties of nodes A and B_i. In Impulse TCAD, the FVM equations are set up on the node A as

$$g = \sum_i f(A, B_i) S_i + s(A) V = 0, \tag{1}$$

Where $f(A, B_i)$ is a flux term directing from B_i to A, $s(A)$ is a source term, S_i is an area of the boundary surface, V is a volume of the CV, and the summation is taken over all neighbouring nodes. The conservation rule Eq. (1) is satisfied

978-1-7281-0940-4/19/$31.00 ©2019 IEEE

simultaneously over all CVs, by adjusting variable properties v on each node. The Newton method is employed to calculate the adjustment δv:

$$\begin{aligned} \delta g &\simeq \sum_i \left(\frac{\partial f}{\partial v_A} \delta v_A + \frac{\partial f}{\partial v_{B_i}} \delta v_{B_i} \right) S_i + \frac{\partial s}{\partial v_A} \delta v_A V \\ &= -g, \end{aligned} \tag{2}$$

which is written collectively as $(\partial \boldsymbol{G}/\partial \boldsymbol{V})\delta \boldsymbol{V} = -\boldsymbol{G}$. Here, v and g are collected from all nodes to form vectors $\boldsymbol{V}$ and $\boldsymbol{G}$, respectively. The Jacobian $\partial \boldsymbol{G}/\partial \boldsymbol{V}$ is a large and sparse matrix, which is constructed from derivatives $\partial f/\partial v$ and $\partial s/\partial v$. The update $\delta \boldsymbol{V}$ at each Newton iteration is obtained by solving the large linear equation.

In Impulse TCAD, users set up the FVM equations by providing the flux term $f(A, B_i)$ and the source term $s(A)$ symbolically. To illustrate, a setup of coupled Poisson and drift-and-diffusion equations is shown below:

```
# Poisson
f_Psi = eps * (Psi.j - Psi.i) / dx_d
s_Psi = q * (ND.i - NA.i + Hole.i - Elec.i)

# Drift & Diffusion
E = abs((Psi.j - Psi.i) / dx_d)
mu_n = CaugheyThomas(E, T, mu_n, beta_n, Vsat0_n, A_n, T0_n)
mu_p = CaugheyThomas(E, T, mu_p, beta_p, Vsat0_p, A_p, T0_p)
x = q / (kB*T) * (Psi.j - Psi.i)
f_Elec = -mu_n * kB*T / dx_d * C_Bernoulli( x, Elec.j, Elec.i)
f_Hole =  mu_p * kB*T / dx_d * C_Bernoulli(-x, Hole.j, Hole.i)
s_Elec = -q * ShockleyReadHall(Elec.i,Hole.i,ni,tau_n,tau_p)
s_Hole = -s_Elec

Eqn = ( [(f_Psi, s_Psi), (f_Elec, s_Elec), (f_Hole, s_Hole)],
        [Psi, Elec, Hole] )
```

Here, Psi, Elec, Hole, NA, and ND are the device properties for the electric potential, electron density, hole density, acceptor density, and donor density, respectively, where suffixes ".i" and ".j" denote the values on the primary (A) and adjacent (B_i) nodes, respectively. The property dx_d gives the edge length between A and B_i. Other symbols are defined elsewhere as numerical constants. At the last line, the FVM equations are defined as a list of (flux, src) terms, along with a list of variable properties for which the equations have to be solved. As shown in the listing, the flux and source terms are constructed by using the device properties in a straightforward manner.

B. Automatic Differentiation

The given FVM equations are solved by the Newton method, where the automatic differentiation facility of Theano [6] is used. The device properties listed above are implemented as Theano symbols, and equations written with them compose Theano expressions. Indeed, the functions CaugheyThomas, C_Bernoulli, and ShockleyReadHall return Theano expressions for the mobility [7], the interpolated electric current [8], and the generation-recombination rate [9], respectively.

Under the hood, differential expressions of the FVM equations are derived with respect to the given variable properties. C source codes are then generated to calculate both the equations and their derivatives numerically, which are compiled and linked to build a module accessible from Python. The equations are also analyzed to extract a list of incorporated device properties. For example, [Psi.i, Psi.j, dx_d] and [Elec.i, Hole.i, ND.i, NA.i] are extracted from f_Psi and s_Psi, respectively. Note that eps and q in these equations are constants, which are embedded in the source codes directly. The module is called repetitively to build $\boldsymbol{G}$ and its Jacobian, where the required arguments are assembled according to the property list. This Jacobian construction is parallelized by MPI [10]–[14], as well as the direct solver for the linear equation [15]–[19].

C. Comparison with other simulators

In our survey, PRISM [20], [21] and Genius [22] employ an automatic differentiation scheme in the semiconductor device simulation. PRISM is based on a FORTRAN package IVPACK, where derivatives are automatically generated for functions programmed with IVPACK function calls. Mobility models and generation-recombination terms can be extended easily with the IVPACK scheme, though details are not available. Genius implements a C++ framework of automatic differentiation, with which users can customize materials. In Genius, dominant equations and associating physical models are predefined in the simulator, and users can modify physical parameters (such as electric permittivity), as well as the functional form of the physical models (such as the band gap narrowing model). To customize Genius, building a dynamic library with a set of prescribed APIs is mandatory, though rebuilding the whole simulator can be avoided.

In contrast to these simulators, the automatic differentiation scheme is hidden from users in Impulse TCAD. Instead of composing individual derivatives of physical models, the automatic differentiation is used to derive the Jacobian of the FVM equations (1) directly. This approach allows users to customize dominant equations themselves, without the hassles of defining auxiliary functions. The customizations are also reflected instantly, without the complicated compile and link process. These features are prerequisite to develop non-standard physical models of new materials, as shown in the next section.

III. Application to Ferroelectric Material

This section demonstrates how Impulse TCAD helps the development of new physical models by taking the negative capacitance field effect transistors (NC FET) as an example [4], [23]–[26]. In the NC FET, ferroelectric materials like HfO_2 are embedded in the gate stack. The device structure used in the present study is shown in Fig. 2. To solve the steady states of the device, the standard Poisson and drift-and-diffusion equations are applied on materials other than HfO_2.

The electric potential ψ of the HfO_2 regions is solved by the Poisson equation coupled with the Devonshire equation [27].

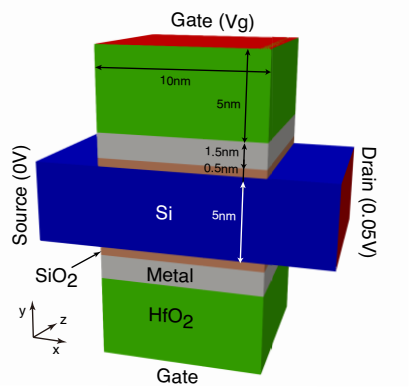

Fig. 2. Device structure of the negative capacitance FET.

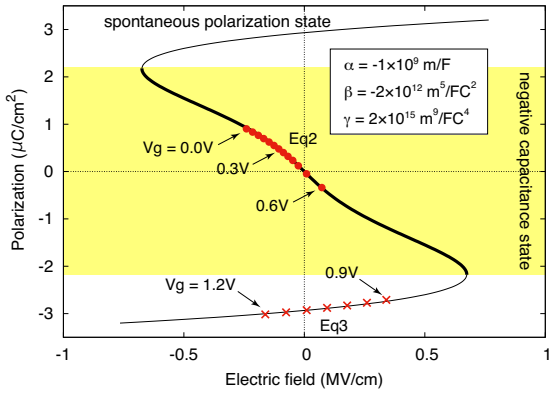

Fig. 3. The polarization of a ferroelectric material as a function of the electric field. The employed material parameters are shown in the inset. The calculated steady states are plotted by red dots and crosses for Eq2 and Eq3, respectively.

The polarization of the ferroelectrics is described by the Devonshire equation,

$$E = 2\alpha P + 4\beta P^3 + 6\gamma P^5, \tag{3}$$

where $E = -\nabla\psi$ is an electric field, P is a polarization, and α, β, γ are constant parameters. The E-P relationship of Eq. (3) is depicted in Fig. 3. The negative capacitance (NC) state is plotted with a thick line, which is of our major interest. The ferroelectric axis of HfO_2 is taken to be perpendicular to the channel (Y axis), and the polarizations parallel to the channel are treated as normal dielectrics.

Assuming that there is no charge in the HfO_2 regions, the Poisson equation is written in the integrated form as

$$\int_S (\epsilon_0 E + P) dS = 0. \tag{4}$$

Several approaches are possible to implement Eq. (4) in Impulse TCAD. To start with, the device properties are declared as follows:

```
1  Psi    = NodeProp("Psi")
2  Pol    = NodeProp("Pol")
3  Elf    = NodeProp("Elf")
4  Pos    = NodeVec("Pos")
5  dx_d = EdgeProp("dx_d")
```

Psi, Pol, and Elf are the scalar properties giving electric potential, polarization, and electric field, respectively, while Pos is the 3D vector property giving spatial location. An edge property dx_d gives an edge length between A and B_i. Constant parameters eps0, eps, alpha, beta, and gamma are also prepared elsewhere.

In the first approach, the polarization Pol is supplied as a parameter:

```
6  E    = -(Psi.i - Psi.j) / dx_d
7  Dir  = (Pos.i - Pos.j) / dx_d
8  Evec = E * Dir
9  Px   = eps * Evec[0]
10 Py   = (Pol.i + Pol.j) / 2
11 Pz   = eps * Evec[2]
12 P    = Px*Dir[0] + Py*Dir[1] + Pz*Dir[2]
13 Eq1  = ([(eps0 * E + P, 0)], [Psi])
```

Before each Newton iteration, Pol is calculated from the gradient of Psi, by solving Eq. (3) for the NC state. The polarization vector (Px, Py, Pz) is calculated at the edge center, which is projected on the edge to define the electric flux density along the edge. In this approach, the dependence of Pol on Psi is not reflected directly in the Newton method, which causes severe difficulty in convergence. Note that the HfO_2 regions of Fig. 2 consist from 1500 node points, and Eq. (4) has to be satisfied on them simultaneously.

In the second approach, the ferroelectric polarization is symbolically calculated in the equation:

```
14 Py2 = P_of_E(Evec[1])
15 P2  = Px*Dir[0] + Py2*Dir[1] + Pz*Dir[2]
16 Eq2 = ([(eps0 * E + P2, 0)], [Psi])
```

Here, P_of_E() is a Theano operator defined in Impulse TCAD, which solves Eq. (3) to give P from E for the NC state. As a result, $\partial P/\partial\psi_A$ and $\partial P/\partial\psi_{B_i}$ are incorporated in the Jacobian, by way of the chain rule. This approach successfully calculates the steady states of the NC devices [24], [25], with the gate potential V_g varied at the step of 0.05 V. For the converged states, the electric field and the polarization are averaged over the upper HfO_2 region, and are plotted in Fig. 3 using red dot symbols. When V_g is increased beyond 0.6 V, some of the node points start to escape from the NC state, where the simulation stops.

In the previous two approaches, the spontaneous polarization (SP) states of the ferroelectrics (thin lines in Fig. 3) is ignored. In the third approach, the SP state is incorporated by solving Eq. (3) coupled together with the Poisson equation:

```
17 Devonshire = 2*alpha*Pol.i + 4*beta*Pol.i**3 \
18              + 6*gamma*Pol.i**5 - Elf.i
19 Eq3 = ([(eps0*E + P, 0), (0, Devonshire)], [Psi, Pol])
```

Both Psi and Pol are treated as variable node properties, which are solved in a coupled manner. Similarly to Pol in the first approach, the electric field Elf is prepared before each Newton iteration and supplied as a parameter, so that the dependence of Elf on Psi is ignored in the Newton method. This approach is still convergent, however, because Elf is less

sensitive to V_g than Pol. Note that the full Newton approach is also possible for Elf [28], by constructing the electric field from those E defined on the edges. Starting from an initial condition tailored for the SP state, the simulation converges with some intervention in the iterative process, which are plotted in Fig. 3 using red cross symbols. The Newton method did not converge for V_g lower than 0.9 V, probably because the NC state becomes possible for some node points, causing oscillation in the Newton iteration. Because both of the SP and NC states can be described, this approach is employed in the transient analysis [4], [26], where the transition between the SP and NC states are simulated.

IV. Conclusion

A new semiconductor device simulator named Impulse TCAD is introduced, which allows easy handling of non-standard physical models. Both device properties and their dominant equations can be customized dynamically, which accelerates the development of simulation schemes for novel devices. Auxiliary functions required to solve the equations are generated automatically by using the automatic differentiation technique. Usability of the simulator is demonstrated by taking NC FET as an example, where the physical model of ferroelectric materials is implemented in several different ways on the same platform.

Impulse TCAD has several other features not mentioned in the manuscript that makes the simulator more accessible. A set of materials are predefined, where typical dominant equations and boundary conditions are packaged. They can be used as a starting point of the customization. Several utility functions are also provided, with which users can set up simulations according to prescriptions, export snapshots to files, import device properties from the snapshot files, check consistency of equations, and so on. The exported snapshots can be examined by using the state-of-the-art viewers [29] directly without conversion, which allows ready steering of simulations. With these features, we hope Impulse TCAD will contribute to the development of the next generation semiconductor devices.

References

[1] S. Selberherr, A. Schutz, and H. W. Potzl, "MINIMOS - a two-dimensional mos transistor analyzer," *IEEE Journal of Solid-State Circuits*, vol. 15, no. 4, pp. 605–615, Aug 1980.

[2] C. S. Rafferty, M. R. Pinto, and R. W. Dutton, "Iterative methods in semiconductor device simulation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 4, no. 4, pp. 462–471, October 1985.

[3] W. L. Engl, H. K. Dirks, and B. Meinerzhagen, "Device modeling," *Proceedings of the IEEE*, vol. 71, no. 1, pp. 10–33, Jan 1983.

[4] T. Ikegami, K. Fukuda, J. Hattori, H. Asai, and H. Ota, "A tcad device simulator for exotic materials and its application to a negative-capacitance fet," *Journal of Computational Electronics*, vol. 18, no. 2, pp. 534–542, Jun 2019.

[5] "Impulse TCAD." [Online]. Available: https://unit.aist.go.jp/neri/en/ImpulseTCAD/index.html

[6] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016. [Online]. Available: http://arxiv.org/abs/1605.02688

[7] D. M. Caughey and R. E. Thomas, "Carrier mobilities in silicon empirically related to doping and field," *Proceedings of the IEEE*, vol. 55, no. 12, pp. 2192–2193, Dec 1967.

[8] D. L. Scharfetter and H. K. Gummel, "Large-signal analysis of a silicon read diode oscillator," *IEEE Transactions on Electron Devices*, vol. 16, no. 1, pp. 64–77, Jan 1969.

[9] W. Shockley and W. T. Read, "Statistics of the recombinations of holes and electrons," *Phys. Rev.*, vol. 87, pp. 835–842, Sep 1952.

[10] L. D. Dalcin, R. R. Paz, P. A. Kler, and A. Cosimo, "Parallel distributed computing using Python," *Advances in Water Resources*, vol. 34, no. 9, pp. 1124 – 1139, 2011, new Computational Methods and Software Tools.

[11] L. Dalcín, R. Paz, M. Storti, and J. D'Elía, "MPI for Python: Performance improvements and MPI-2 extensions," *Journal of Parallel and Distributed Computing*, vol. 68, no. 5, pp. 655 – 662, 2008.

[12] L. Dalcín, R. Paz, and M. Storti, "MPI for Python," *Journal of Parallel and Distributed Computing*, vol. 65, no. 9, pp. 1108 – 1115, 2005.

[13] MPI Forum, "MPI: a message passing interface standard," *Int. J. Supercomput. Appl.*, vol. 8, no. 3-4, pp. 159–416, 1994.

[14] ——, "MPI2: a message passing interface standard," *Int. J. High Perform. Comput. Appl.*, vol. 12, no. 1-2, pp. 1–299, 1998.

[15] P. R. Amestoy, I. S. Duff, J. Koster, and J.-Y. L'Excellent, "A fully asynchronous multifrontal solver using distributed dynamic scheduling," *SIAM Journal on Matrix Analysis and Applications*, vol. 23, no. 1, pp. 15–41, 2001.

[16] P. R. Amestoy, A. Guermouche, J.-Y. L'Excellent, and S. Pralet, "Hybrid scheduling for the parallel solution of linear systems," *Parallel Computing*, vol. 32, no. 2, pp. 136–156, 2006.

[17] X. S. Li, "An overview of SuperLU: Algorithms, implementation, and user interface," *ACM Trans. Math. Softw.*, vol. 31, no. 3, pp. 302–325, Sep. 2005.

[18] X. S. Li and J. W. Demmel, "SuperLU_DIST: A scalable distributed-memory sparse direct solver for unsymmetric linear systems," *ACM Trans. Mathematical Software*, vol. 29, no. 2, pp. 110–140, June 2003.

[19] X. Li, J. Demmel, J. Gilbert, iL. Grigori, M. Shao, and I. Yamazaki, "SuperLU Users' Guide," Lawrence Berkeley National Laboratory, Tech. Rep. LBNL-44289, September 1999, http://crd.lbl.gov/~xiaoye/SuperLU/. Last update: October 2014.

[20] W. Schoenmaker, R. Vankemmel, R. Cartuyvels, W. Magnus, and B. Tijskens, "Status of the device simulator PRISM," *COMPEL - The international journal for computation and mathematics in electrical and electronic engineering*, vol. 10, no. 4, pp. 631–640, 1991.

[21] E. Tijskens, W. Schoenmaker, and K. De Meyer, "Automatic numerical evaluation of derivatives and its use in device simulators," in *NUPAD IV. Workshop on Numerical Modeling of Processes and Devices for Integrated Circuits,*, May 1992, pp. 251–255.

[22] Cogenda, "Genius: 3D parallel device simulator." [Online]. Available: http://www.cogenda.com/article/Genius

[23] S. Salahuddin and S. Datta, "Use of negative capacitance to provide voltage amplification for low power nanoscale devices," *Nano Letters*, vol. 8, no. 2, pp. 405–410, 2008, pMID: 18052402.

[24] H. Ota, T. Ikegami, J. Hattori, K. Fukuda, S. Migita, and A. Toriumi, "Fully coupled 3-D device simulation of negative capacitance FinFETs for sub 10 nm integration," in *2016 IEEE International Electron Devices Meeting (IEDM)*, Dec 2016, pp. 12.4.1–12.4.4.

[25] H. Ota, K. Fukuda, T. Ikegami, J. Hattori, H. Asai, S. Migita, and A. Toriumi, "Perspective of negative capacitance FinFETs investigated by transient TCAD simulation," in *2017 IEEE International Electron Devices Meeting (IEDM)*, Dec 2017, pp. 15.2.1–15.2.4.

[26] H. Ota, T. Ikegami, K. Fukuda, J. Hattori, H. Asai, K. Endo, S. Migita, and A. Toriumi, "Multi-domain dynamics of ferroelectric polarization and its coherency-breaking in negative capacitance field-effect transistors," in *2018 IEEE International Electron Devices Meeting (IEDM)*, Dec 2018, pp. 9.1.1–9.1.4.

[27] A. F. Devonshire, "Xcvi. theory of barium titanate," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 40, no. 309, pp. 1040–1063, 1949.

[28] J. Hattori, T. Ikegami, K. Fukuda, H. Ota, S. Migita, and H. Asai, "Device simulation of negative-capacitance field-effect transistors with a ferroelectric gate insulator," in *2018 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, Sep 2018, pp. 214–219.

[29] U. Ayachit, *The ParaView Guide: A Parallel Visualization Application*. Kitware, 2015.