# Advanced Equation Processing for TCAD

Philipp Schwaha[1], René Heinzl[1], Michael Spevak[2], and Tibor Grasser[1]

[1] Christian Doppler Laboratory at the Institute for Microelectronics, TU Wien
schwaha@iue.tuwien.ac.at,
[2] Institute for Microelectronics, TU Wien, spevak@iue.tuwien.ac.at,

**Abstract.** The need for an advanced equation processing frame work offering high performance as well as high expressiveness is presented. A system of coupled non-linear equations from the field of TCAD is used to demonstrate the approach and results for the investigated equations are presented.

## 1 Introduction

Semiconductor devices have become an ubiquitous commodity and users have come to expect a constant increase of device performance at higher integration densities and falling prices. To most efficiently eliminate the technological obstacles encountered the international technology roadmap for semiconductors (ITRS)[1] has been jointly developed by the semiconductor industry. Technology computer aided design (TCAD), which is a specialised branch of scientific computing, is a crucial tool for achieving the set goals as simulations help to greatly reduce the costs in development of new devices as well as manufacturing equipment. It is this quest for ever decreasing device dimensions and faster switching speeds that results in ever growing requirements on the employed simulation methodology. New, more detailed and more sophisticated models are required to simulate not only the operation but also the manufacture of the devices. While computer performance is steadily increasing the additional complexity of these simulation models easily outgrows this gain in computational power. It is therefore of utmost importance to employ the latest techniques of software development techniques to obtain high performance and thereby ensure adequate simulation times even for complex problems.

The great diversity of physical phenomena present in semiconductor devices themselves and in the processes involved in their manufacture make the field of TCAD extremely challenging. Each of the phenomena can be described by differential equations of varying complexity. The development of several different discretisation schemes has been necessary in order to best model the underlying physics and to accommodate the mathematical peculiarities of each of these equations while transferring them to the discrete world of digital computing.

This non trivial and highly complex scenario yields itself exceptionally well for the application of the functional programming paradigm. It is capable of providing a fast, flexible and extensible means to treat even very complex equations with unsurpassed elegance and ease.

## 2 The Problem

To demonstrate the importance of a method that is both easy and efficient a few examples for the equations encountered in the field of TCAD are presented in this section.

Boltzmann's equation for electron transport in semiconductors, as given in Equation 1, is the base for many calculations for device simulations.

$$\frac{\partial}{\partial t} f + \boldsymbol{v} \cdot \operatorname{grad}_r f + \boldsymbol{F} \cdot \operatorname{grad}_k f = \frac{\partial}{\partial t} f|_{\text{collisions}} \tag{1}$$

Here $f$ is the distribution function and $v$ the velocity of the charge carriers, while $\boldsymbol{F}$ denotes the force of an electric field on these particles.

Due to the complexity of Boltzmann's equation several techniques have been developed which result in different simpler models. One of these is the drift diffusion model, that can be derived from Equation 1 by applying the method of moments [2]. This results in current relations as shown in Equation 2. These equations are solved self consistently with Poisson's equation, given in Equation 3.

$$\mathbf{J}_\mathrm{p} = \mathrm{q}\, p\, \mu_\mathrm{p} \operatorname{grad} \Psi - \mathrm{q}\, D_\mathrm{p} \operatorname{grad} p \qquad \mathbf{J}_\mathrm{n} = \mathrm{q}\, n\, \mu_\mathrm{n} \operatorname{grad} \Psi + \mathrm{q}\, D_\mathrm{n} \operatorname{grad} n \tag{2}$$

$$\operatorname{div} \left(\operatorname{grad} \epsilon\, \Psi\right) = -\rho \tag{3}$$

To this end the equations are discretised using the Scharfetter-Gummel [3] scheme resulting in a non-linear equation system of the form

$$J_{n,ij} = \frac{q\, \mu_n\, U_\mathrm{th}}{d_{ij}} \left(n_j B(\Lambda_{ij}) - n_i B(-\Lambda_{ij})\right) \tag{4}$$

$$\Lambda_{ij} = \frac{\Psi_j - \Psi_i}{U_\mathrm{th}} \qquad B(x) = \frac{x}{\mathrm{e}^x - 1} \tag{5}$$

The discretisation of the differential operators using finite volumes yields:

$$\operatorname{div} x \approx \sum_{v \to e} x\, \frac{A}{V} \qquad \operatorname{grad} x \approx \frac{1}{d} \underset{e \to v}{\Delta} x \tag{6}$$

The formulation so obtained can be implemented using virtually any programming language but as has been explained previously a solution offering high performance is greatly desirable. In order to make maintenance of the code as easy as possible and to achieve a maximum of flexibility it is important to keep the code expressive. The great question is how to achieve both of these seemingly contradicting goals at the same time.

## 3 The Solution

The transformation of the presented equations into highly expressive C++ code under highest performance requirements is presented in the following. The source code of the discretised equations is shown in the accompanying code snipped.

```
// Poisson equation
equation_pot = (sum<vertex_edge>()
[
  diff<edge_vertex>(ZERO)[pot_quan] * area / dist
] + ( n - p + nA - nD) * ( vol * q / (eps0 * epsr)))  (vertex);

// Continuity equation for electrons
equation_n = (sum<vertex_edge>()
[
 diff<edge_vertex>(-n_quan*Bern(diff<edge_vertex>()[ pot_quan]/U_th),
                   -n_quan*Bern(diff<edge_vertex>()[-pot_quan]/U_th))
                   * q * mu_n * U_th * area/dist
]) (vertex);
```

The concept of the mathematical equations is modelled by directly mapping the mathematical expressions to function objects. The Bernoulli function, given in Equation 5, is mapped to `Bern`, while `pot_quan` is a functional object providing access to a quantity. The discretised differential operators are easily recognised as the `diff` and `sum` constructs. This whole expression is then examined at compile time to construct an optimised means to obtain the needed data. The complex resulting from this mapping is completed by specifying the evaluation locality, a vertex in this case, and thereby establishing the crossing of the compile-time run-time border. Thereby a link between the still continuous formulation of the equation and a specific tessellation of the simulation domain is formed. As a consequence the implementation makes no assumptions about the dimension or topology of the problem and is therefore suitable for any dimension and any topology. This dimension independent formulation is only made possible by the combination of the separation of algorithms from data sources by the use of iterators and the employment of functional objects.

By introducing the requirement that all functional objects have to provide their derivative, it is possible to reap an additional benefit from this implementation because the tedious and error prone calculation of the Jacobian, outlined in Equation 7, as needed for the solution of the nonlinear equation system, can be automated.

$$\boldsymbol{J} = \frac{\partial \boldsymbol{f}}{\partial x_i} \tag{7}$$

The combination of high expressiveness at such performance levels is only possible by using all the facilities provided by C++. Currently no other language offers sufficient support for all the necessary programming paradigms to enable this high level abstraction at this runtime speed.

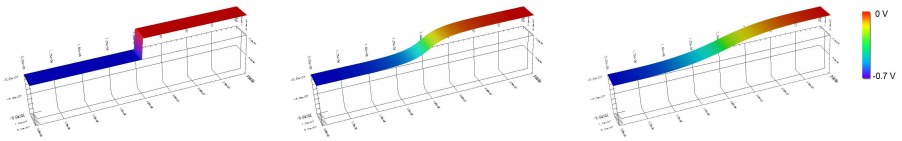


**Fig. 1.** Potential of a pn diode during different stages of the newton iteration. From initial (left) to the final result(right).

The linear solver needed for the solution of the equation system described by the thusly assembled matrix is provided by the Trilinos project [4].

To demonstrate that the proposed scheme is indeed operational we provide figures made showing calculation results. Figure 1 shows the potential within a pn diode at different stages of calculation. The leftmost figure shows the initial potential, while the rightmost depicts the final solution. The centre image shows an intermediate result that has not yet fully converged. The visualisation of the calculation is available in real time, making it possible to observe the evolution of the solution. It is realised by an extension to OpenDX [5] and proves to be invaluable for the adjustment of simulation parameters. Figure 2 shows the results for a two dimensional MOS FET transistor.

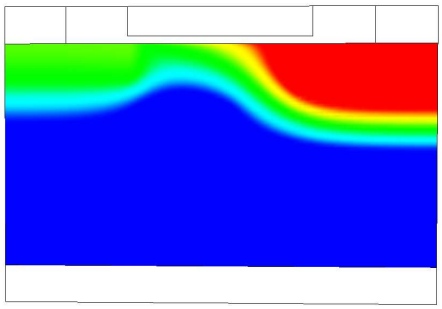

**Fig. 2.** Potential within a MOS FET.

We created the infrastructure enabling the formulation of this problem in such a highly expressive way that does not impose an overly high abstraction penalty but retains excellent runtime performance.

## 4 Conclusion

The importance of being able to cope with a varying number of dimensions becomes apparent when reconsidering Equation 1. In contrast to the drift diffusion model represented by Equations 2 it not only contains spacial dimensions but also includes a set of dimensions describing the impulses of the charge carriers forming a six dimensional phase space. So when turning to solve Equation 1 more accurately additional dimensions have to be considered. The described approach is capable to provide all the necessary infrastructure to accomplish this. An implementation, utilising the presented facilities, of such a method using spherical harmonics expansion of arbitrary order is currently under development. Due to the increased number of dimensions the matter of performance gains additional importance to maintain reasonable simulation times.

## References

1. International Technology Roadmap for Semiconductors, 2005.
2. S. Selberherr, *Analysis and Simulation of Semiconductor Devices* (Springer, Wien–New York, 1984).
3. D. Scharfetter and H. Gummel, IEEE Trans.Electron Devices **16**, 64 (1969).
4. M. A. Heroux *et al.*, ACM Transactions on Mathematical Software , for TOMS special issue on the ACTS Collection.
5. *IBM Visualization Data Explorer*, 3rd ed., IBM Corporation, Yorktown Heights, NY, USA, 1993.