Proceedings of the 7th IASTED International Conference
SOFTWARE ENGINEERING AND APPLICATIONS
November 3-5, 2003, Marina Del Rey, CA, USA

A GENERALLY APPLICABLE APPROACH FOR ADVANCED EQUATION ASSEMBLING

Stephan Wagner[1], Tibor Grasser[1], Claus Fischer*, and Siegfried Selberherr[2]

[1]Christian-Doppler-Laboratory for TCAD in Microelectronics
at the Institute for Microelectronics

[2]Institute for Microelectronics
Technical University Vienna, A-1040 Vienna, Austria
E-mail: Wagner@iue.tuwien.ac.at
Phone: +43(1)58801-36037, Fax: +43(1)58801-36099

*Firma Dr. Claus Fischer
Gustav Fuhrichweg 24/1, A-2201 Gerasdorf bei Wien Austria

KEY WORDS
Partial Differential Equations, Finite Boxes Discretization, Boundary Conditions, Linear Equation Systems

ABSTRACT
We present a generally applicable approach which allows to efficiently assemble equations necessary for solving a system of nonlinear partial differential equations discretized on a grid. Since the nonlinear problem is usually solved by a damped Newton algorithm the solution of a linear equation system has to be obtained at each step. Our assembly approach for these systems has been originally developed for the simulation of semiconductor devices based on the Finite Boxes discretization scheme. It has been rigorously implemented in a simulator module which is currently used in the general purpose device and circuit simulator MINIMOS-NT. In addition to the assembly itself, several requirements of the simulation process, namely the representation of boundary conditions, physically motivated variable transformation, and numerical conditioning, are taken into account.

1 Introduction

The Finite Boxes discretization method is employed in various kinds of numerical tools and simulators for the fast and accurate solution of nonlinear partial differential equation (PDE) systems. The resulting discretized problem is then usually solved by damped Newton iterations which require the solution of a linear equation system at each step. The extensibility and efficiency of any simulator highly depends on the capabilities of the core modules responsible for handling the linear equation systems.

We present an advanced approach for designing the equation assembly process which has been implemented in the general purpose device and circuit simulator MINIMOS-NT [1]. Besides the basic semiconductor equations [2], several different types of transport equations can be solved. Among these are the hydrodynamic equations which capture hot-carrier transport [3], the lattice heat flow equation to cover thermal effects like self-heating [4], and the circuit equations to connect single devices to a circuit [5], both electrically and thermally. Furthermore, various interface and boundary conditions are taken care of, which include Ohmic and Schottky contacts, thermionic field emission over and tunneling through various kinds of barriers. This demands a sophisticated system handling the equation assembly in order to keep the simulator design flexible. To implement such a system, the requirements have been identified and generalized. A crucial aspect is also the requirement of assembling and solving complex-valued linear equation systems. For that reason the module is able to handle both real-valued and complex-valued contributions and systems.

2 The Analytical Problem

In order to analyze the electronic properties of an arbitrary semiconductor structure under all kinds of operating conditions, the effects related to the transport of charge carriers under the influence of external fields must be modeled. In MINIMOS-NT carrier transport can be treated by the drift-diffusion and the hydrodynamic transport models.

Both models are based on the semiclassical Boltzmann transport equation which is a time-dependent partial integro-differential equation in the six-dimensional phase space. By the so-called method of moments this equation can be transformed in an infinite series of equations. Keeping only the zero and first order moment equations (with proper closure assumptions) yields the basic semiconductor equations (drift-diffusion model).

These equations as given first by VanRoosbroeck [6] are the Poisson equation (1), the continuity equations for

electrons (2) and holes (3) including a drift and diffusion term:

$$\mathrm{div}(\varepsilon \cdot \mathrm{grad}\ \psi) = -\rho \tag{1}$$

$$\mathrm{div}\,(D_n \cdot \mathrm{grad}\ n - \mu_n \cdot n \cdot \mathrm{grad}\ \psi) = R + \frac{\partial n}{\partial t} \tag{2}$$

$$\mathrm{div}\,(D_p \cdot \mathrm{grad}\ p + \mu_p \cdot p \cdot \mathrm{grad}\ \psi) = R + \frac{\partial p}{\partial t} \tag{3}$$

The unknown quantities of this equation system are the electrostatic potential ψ, and the electron and hole concentrations n and p, respectively. ε is the dielectric permittivity of the semiconductor, ρ denotes the space charge density, D_n and D_p are the diffusion coefficients, μ_n and μ_p stand for the carrier mobilities, and R describes the net recombination rate. These variables depend on the unknown quantities ψ, n, and p [2] and have to be modeled properly [7]. The equation system is rendered by these models in a nonlinear form.

The heat flow equation (4) is added to account for thermal effects in the device:

$$\mathrm{div}(\kappa_{\mathrm{L}} \cdot \mathrm{grad} T_{\mathrm{L}}) = \rho_{\mathrm{L}} \cdot c_{\mathrm{L}} \tag{4}$$

This equation requires proper modeling of the thermal conductivity κ_{L}, the mass density ρ_{L}, and the heat capacity c_{L}. The parameters of equations (1) to (3) depend also on the lattice temperature T_{L} and have to be modeled properly.

Considering two additional moments gives the hydrodynamic model [8], where the carrier temperatures are allowed to be different from the lattice temperature. Since the current densities depend then on the respective carrier temperature, two more quantities, the electron temperature T_n and the hole temperature T_p, are added.

Basically, a device structure can be divided into several segments to enable simulation of advanced heterostructures and to properly account for all conditions (which may cause very abrupt changes) at the contacts and interfaces between these segments, respectively. Every segment represents an independent domain D in one, two, or three dimensions where the PDEs are posed. The equations are implicitly formulated for a quantity x as $f_{(x)} = 0$ and termed control functions. In order to fully define the mathematical problem, suitable boundary conditions for contacts, interfaces, and external surfaces have to be applied.

Generally, such a system cannot be solved analytically, and the solution must be calculated by means of numerical methods. This approach normally consists of three tasks:

1. The domain D is partitioned into a finite number of subdomains D_i, in which the solution can be approximated with a desired accuracy.

2. The PDE system is approximated in each of the subdomains by algebraic equations. The unknown functions are approximated by functions with a given structure. Hence, the unknowns of the algebraic equations are approximations of the continuous solutions at discrete grid points in the domain. Thus, generally a large system of nonlinear, algebraic equations is obtained with unknowns comprised of approximations of the unknown functions at discrete points.

3. A solution of the unknowns of the nonlinear algebraic system must be computed. In the best case an exact solution of this system can be obtained, which represents a good approximation of the solution of the analytically formulated problem (which cannot be solved exactly). The quality of the approximation depends on the resolution of the partitioning into subdomains as well as on the suitability of the approximating functions.

3 The Discretized Problem

For the derivation of the discrete problem several methods can be applied. We deal here with point residual methods: the finite difference method based on rectangular grids or the finite boxes (box integration) method allowing general unstructured grids. In the case of orthogonal rectangular grids both methods yield the same discretization.

Nonlinear partial differential equations of second order can appear in three variants: elliptic, parabolic, and hyperbolic PDEs. The Poisson equation as well as the steady-state continuity equations form a system of elliptic PDEs, whereas the heat-flow equation is parabolic. To completely determine the solution of an elliptic PDE boundary conditions have to be specified. Since parabolic and hyperbolic PDEs describe evolutionary processes, time normally is an

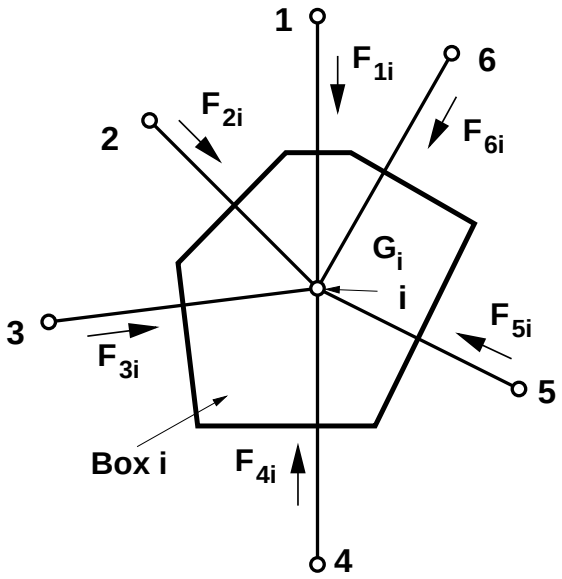

Figure 1. Box i with 6 neighbors

independent variable and an initial condition is additionally required. Hence, also the transient continuity equations are parabolic.

Applying the finite boxes discretization scheme [2] the equations are integrated over a control volume (subdomain, usually obtained by a Voronoi tesselation) D_i which is associated with the grid point P_i. For this grid point a general equation for the quantity x is implicitly given as

$$f^{\mathrm{S}}_{x_i} = \quad \sum_j F_{x_{i,j}} + G_i \quad = 0 \tag{5}$$

where j runs over all neighboring grid points in the same segment, $F_{x_{i,j}}$ is the flux between points i and j, and G_i is the source term (see Fig. 1).

Grid points on the boundary $\partial\mathrm{D}$ are defined as having neighbor grid points in other segments. Thus, (5) does not represent the complete control function $f_{(x)}$, since all contributions of fluxes into the contact or the other segment are missing. For that reason, the information for these boxes has to be completed by taking the boundary conditions into account. Common boundary conditions are the Dirichlet condition which specifies the solution on the boundary $\partial\mathrm{D}$, the Neumann condition which specifies the normal derivative, and the linear combination of these conditions giving an intermediate type:

$$\mathbf{n} \cdot \mathrm{grad} x + \sigma x = \delta \tag{6}$$

Generally, the form of these conditions depends on the respective boundary models. For that reason the equation assembly is often performed in a coupled way, causing complicated modules. For instance, it is absolutely necessary to differ between interior and boundary points. Considering a general tetrahedron, there exist many kinds of boundary points (depending on the number of edges involved), which have to be treated separately. This leads to a complicated implementation of the models and can make simplifications necessary. Thus, due to organizational and implementational issues this form of coupling should be avoided.

More complex models with exponential interdependence between the solution variables such as thermionic field emission interface conditions [9] have also been implemented.

The method which has been developed allows to implement the segment models which describe the interior fluxes and their derivatives independently from the boundary models. The segment models do not have to differentiate the point type, they do not even have to care about the boundary model used. The assembly system is responsible for combining all relevant contributions by using the information given by the boundary models.

3.1 Interface Conditions

To account for complex interface conditions, grid points located at the boundary of the segments (see Fig. 2a) have three values, one for each segment (see Fig. 2b) and a third point located directly at the interface which can be used to formulate more complicated interface conditions like e.g., interface charges. However, to simplify notation these interface values will be omitted in our discussion and only the two interface points, i and i', are used.

Basically, the two equations $f^{\mathrm{S}}_{x_i}$ and $f^{\mathrm{S}}_{x_{i'}}$ are completed by adding the missing boundary fluxes $F_{x_{i,i'}}$:

$$f_{x_i} = f^{\mathrm{S}}_{x_i} + F_{x_{i,i'}} = 0 \tag{7}$$

$$f_{x_{i'}} = f^{\mathrm{S}}_{x_{i'}} - F_{x_{i,i'}} = 0 \tag{8}$$

The intermediate type of interfaces (6) and thus also the two other types of interfaces are generally given in linearized form by:

$$\alpha(x_i - \beta x_{i'} + \gamma) = F_{x_{i,i'}} \tag{9}$$

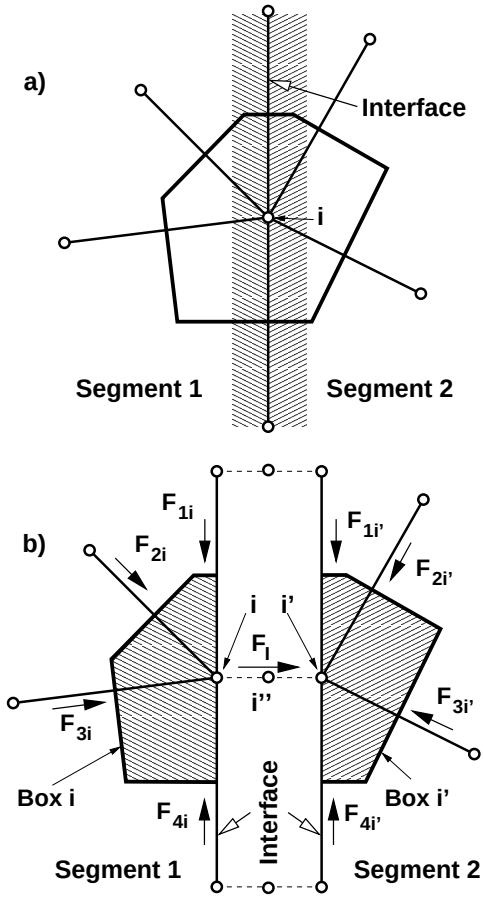

Figure 2. Splitting of interface points: Interface points as given in a) are split into three different points having the same geometrical coordinates b)

α, β, and γ are linearized coefficients, $F_{x_{i,i'}}$ represents the flux over the interface. The three types of interfaces differ in the magnitude of α.

In the case of an arbitrary splitting of a homogeneous region into different segments, the boundary models have to ensure that the simulation results remain unchanged. By adding (8) to (7), the box of grid point P_i can be completed and the boundary flux is eliminated. The merged box is now valid for both grid points, for that reason the respective equation can not only be used for grid point P_i, but also for $\mathrm{P}_{i'}$.

Whereas the segment models assemble the so-called segment matrix, the interface models are responsible for assembling and configuring the interface system consisting of a boundary and special-purpose transformation matrix. New equations based on (9) can be introduced into the boundary matrix without any limitations on α, thus from 0 (Neumann) to ∞ (Dirichlet). The interface models are also responsible for configuring the transformation matrix to combine the segment and boundary matrix correctly. Depending on the interface type there are two possibilities:

- Dirichlet boundaries are characterized by $\alpha \to \infty$. Thus, the implicit equation $x_i = \beta x_{i'} - \gamma$ can be used as a substitute equation. As these equations are normally not diagonally dominant they have a negative impact on the condition number and are configured to be preeliminated (see Section 4).

- For the other types (explicit boundary conditions) the boundary flux is simply added to the segment fluxes. In the case of a large α the transformation matrix can be used to scale the entries by $1/\alpha$ because of the preconditioner used in the solver module.

Note, that all interface-dependent information is administrated by the respective interface model only.

As an additional feature the transformation matrix can be used to calculate several independent boundary quantities by combining the specific boundary value with the segment entries (also in the case of Dirichlet boundaries). For example, the dielectric flux over the interface is calculated as $\sum_i f^{\mathrm{S}}_{x_i}$ and introduced as a solution variable because some interface models require the cross-interface electric field strength to determine tunnel processes. Calculation of the normal electric field is thus trivial. Note, that this is not the case when the normal component of the electric field $\vec{E}_n$ has to be calculated using neighboring points in the case unstructured two- or three-dimensional grids are used.

Fig. 3 illustrates these concepts. The transformations are set up to combine the various segment contributions with the boundary system.

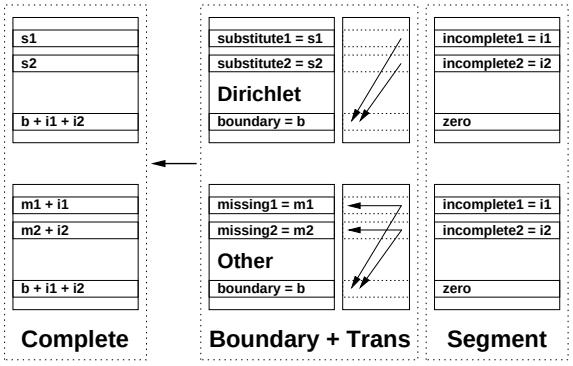

Figure 3. The complete equations are a combination of the boundary and the segment system. This combination is controlled by the transformation matrix and depends on the interface type.

3.2 Boundary Conditions

Contacts are handled in a similar way to interfaces. However, in the contact segment there is only one variable available for each solution quantity (x_{C}). Note, that contacts are represented by spacial multi-dimensional segments. Furthermore, all fluxes over the boundary are handled as additional solution variables F_{C} (e.g., contact charge Q_{C} for Poisson equation, contact electron current $I_{n\mathrm{C}}$ for the electron continuity equation, or H_{C} as the contact heat flow).

With i running over all segment grid points, for explicit boundary conditions one gets

$$f_{x_i} = f^{\mathrm{S}}_{x_i} + F_{x_{i,\mathrm{C}}} = 0 \quad (10)$$

$$f_{F_{\mathrm{C}}} = F_{\mathrm{C}} + \textstyle\sum_i f^{\mathrm{S}}_{x_i} = 0 \quad (11)$$

For example, at Schottky contacts explicit boundary conditions apply. The semiconductor contact potential ψ_{s} is fixed and given as the difference of the metal quasi-Fermi level (which is specified by the contact voltage ψ_{C}) and the metal workfunction difference potential ψ_{wf} [9].

$$\psi_{\mathrm{s}} = \psi_{\mathrm{C}} - \psi_{\mathrm{wf}} \quad (12)$$

For Dirichlet boundary conditions one gets

$$f_{x_i} = x_{\mathrm{C}} - h(x_i) = 0 \quad (13)$$

$$f_{F_{\mathrm{C}}} = F_{\mathrm{C}} + \textstyle\sum_i f^{\mathrm{S}}_{x_i} = 0 \quad (14)$$

Here, x_{C} is the boundary value of the quantity, which is a solution variable, whereas (14) is used as constitutive relation for the actual flow over the boundary F_{C}. $h(x_i)$ denotes the substitute equation. For example, at Ohmic contacts Dirichlet boundary conditions apply. The metal quasi-Fermi level is equal to the semiconductor quasi-Fermi level. With the constant built-in potential ψ_{bi} (calculated after [10]), the contact potential at the semiconductor boundary reads

$$\psi_{\mathrm{s}} = \psi_{\mathrm{C}} + \psi_{\mathrm{bi}}. \quad (15)$$

For Neumann boundaries the flux over the boundary is zero hence the equation assembled by the segment model is already complete.

Having a separate solution variable for the contact voltage avoids numerical problems with large arguments of the Bernoulli function B. If the Scharfetter-Gummel discretization scheme [11] is used, applying the contact voltage directly to the boundary grid point can cause large arguments of B and hence numerical problems.

This is avoided by having a separate variable for the contact voltage. At the beginning of the iteration procedure the constitutive relation for ψ_{C} is violated and will only successively be adapted which guarantees numerical stability.

The generalized boundary condition is the constitutive relation for the contact potential ψ_{C} and reads:

$$f_{\psi_{\mathrm{C}}} = \alpha\psi_{\mathrm{C}} + \beta I_{\mathrm{C}} + \gamma Q_{\mathrm{C}} - \delta = 0 \tag{16}$$

where Q_{C} is the contact charge and $I_{\mathrm{C}} = I_{n_{\mathrm{C}}} + I_{p_{\mathrm{C}}} + \frac{\partial Q_{\mathrm{C}}}{\partial t}$ the contact current. It should be noted that all these quantities are solution variables which are directly available.

3.3 Solving of the Nonlinear System

MINIMOS-NT organizes the solving of the nonlinear, but discretized control functions $\mathbf{f} = \mathbf{0}$ using a damped Newton algorithm (k is the number of the iteration step) [2]:

$$\mathbf{J}^k \cdot \mathbf{x}^{k+1} = \mathbf{f}(\mathbf{v}^k) \tag{17}$$

$$\mathbf{v}^{k+1} = \mathbf{v}^k + F_{\mathrm{d}} \cdot \mathbf{x}^{k+1} \tag{18}$$

$$\mathbf{J} = -\frac{\partial \mathbf{f}}{\partial \mathbf{v}} \tag{19}$$

where $\mathbf{J}$ is the Jacobian matrix, $\mathbf{f}(\mathbf{v})$ the residual and $\mathbf{x}$ the update or correction vector (solution vector of the linear system) that is then used to calculate the next solution vector $\mathbf{v}$ of the Newton approximation.

To avoid overshoot of the solution several damping schemes suggested by Deuflhard [12] or Bank and Rose [13] are providing a damping factor F_{d}. For each Newton iteration step a linear equation system $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ has to be assembled and solved.

3.4 Assembly of the Complete Linear Equation System

The semiconductor device is divided into several segments that are geometrical regions employing a distinct set of models. The implementation of each model is completely independent from other models and each model is basically allowed to enter its contributions to the linear equation system. All boundary and interface issues are completely separated from the general segment models. Hence, also completely independent assembly structures for the boundary and segment system are used.

Thus, the system matrix $\mathbf{A}$ (the Jacobian matrix in Newton approximation) will be assembled from two parts, namely the direct part $\mathbf{A}_{\mathrm{b}}$ (boundary models) and the transformed part $\mathbf{A}_{\mathrm{s}}$ (segment models). The latter is multiplied by the row transformation matrix $\mathbf{T}_{\mathrm{b}}$ from the left before contributing to the system matrix $\mathbf{A}$. The right hand side vector $\mathbf{b}$ is treated the same way:

$$\mathbf{A} = \mathbf{A}_{\mathrm{b}} + \mathbf{T}_{\mathrm{b}} \cdot \mathbf{A}_{\mathrm{s}} \tag{20}$$

$$\mathbf{b} = \mathbf{b}_{\mathrm{b}} + \mathbf{T}_{\mathrm{b}} \cdot \mathbf{b}_{\mathrm{s}} \tag{21}$$

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b} \tag{22}$$

Although in principle every model is allowed to add entries to all components, the assembly module checks two pre-requisites before actually entering the value: first, the quantity the value belongs to is marked to be solved (the user may request only a subset of all provided models) and secondly the priority of the model is high enough to modify the row transformation properties. As stated before, the row transformation is used to complete missing fluxes in boundary boxes. Since a grid point can be part of more than two segments, a ranking using a priority has been introduced. For example, contact models have usually the highest priority and thus their contributions are always used for completion. All three matrices $\mathbf{A}_{\mathrm{b}}$, $\mathbf{A}_{\mathrm{s}}$, and $\mathbf{T}_{\mathrm{b}}$ and the two vectors $\mathbf{b}_{\mathrm{b}}$ and $\mathbf{b}_{\mathrm{s}}$ may be assembled simultaneously, so no assembly sequence must be adhered to. In addition, a forth matrix $\mathbf{T}_{\mathrm{v}}$ is assembled which contains information for an additional variable transformation.

4 The Assembly Module

MINIMOS-NT consists of two separate modules responsible for assembling and solving linear equation systems:

1. the assembly module which is directly accessed by the implemented physical models of the simulator, provides an effective application programming interface, various transformation algorithms and the preelimination system. In addition, sorting and scaling plug-ins can be called.

2. the solver module which is plugged into the assembly module, is responsible for solving the so-called inner linear equation system. The module currently used provides a direct (Gaussian) method and two iterative solver schemes.

The key demands on the assembly module (class) can be summarized as follows:

1. The Application Programming Interface provides methods for

   - adding values to the segment system
   - adding values to the boundary system
   - adding values to the transformation matrix
   - deleting equations

- setting elimination flags
- administration of priority information

2. The row transformation performs a linear combination of rows to extinguish large entries (see Section 3.4).

3. The variable transformation is used to reduce the coupling of the semiconductor equations. Especially in the case of mixed quantities in the solution vector, a variable transformation is sometimes helpful to improve the condition of the linear system. The representation chosen here allows to specify fairly arbitrary variable transformations to be applied to the system. Basically, a matrix $\mathbf{T}_{\mathrm{v}}$ is assembled and multiplied with the system matrix.

4. The preelimination is required to eliminate problematic equations by Gaussian elimination in order to improve the condition of the inner system matrix. Matrix $\mathbf{A}_{\mathrm{s}}$ consists of fluxes that will (if the control functions are correctly assigned to the variables) satisfy the criterion of diagonal-dominance that is necessary to make the linear equation system solvable with an iterative solver. The transformations and additional terms imposed by the boundary conditions may heavily disrupt this feature both in structural and numerical aspects. Some of the boundary or interface conditions can make the full system matrix so ill-conditioned that this simply prevents iterative linear solvers from converging.

5. Specific plug-ins are called for

   - Scaling: Since a threshold value (tolerance) is used to decide whether to keep or skip an entry, the preconditioner used (Incomplete-LU factorization) requires a system matrix having entries of the same order of magnitude.
   - Sorting: Reduction of the bandwidth of a matrix to reduce the fill-in.
   - Solving: Calculate the solution vector of the linear equation system.

6. After reverting all transformations and backsubstituting the preeliminated equations, the output of the assembly module is the complete solution vector. In addition, the right-hand-side vector is returned which can be used for various norm calculations.

5 Conclusion

We presented the concept and implementation of an advanced assembly approach successfully applied in the device and circuit simulator MINIMOS-NT. All conceptional and numerical features required for assembling and solving linear systems arising from semiconductor device and circuit simulation are provided. We developed a formulation which allows to independently treat segments, boundaries, and interface models. All fluxes over boundaries are available as solution variables, which simplifies the formulation of boundary conditions and circuit equations.

The presented concepts result in superior stability of MINIMOS-NT without restricting model implementation and further development. The general approach for treating boundary conditions yields in combination with several preconditioning measures diagonal-dominant linear equation systems well prepared for advanced solver algorithms. As a result, boundary conditions for specific operating points can be directly applied without successively stepping to the desired value as is very common even in commercial simulators.

References

[1] http://www.iue.tuwien.ac.at/software/minimos-nt, "Minimos-NT 2.0 User's Guide, IμE." Institut für Mikroelektronik, Technische Universität Wien, Austria, 2002.

[2] S. Selberherr, *Analysis and Simulation of Semiconductor Devices*. Wien–New York: Springer, 1984.

[3] K. Bløtekjær, "Transport Equations for Electrons in Two-Valley Semiconductors," *IEEE Trans.Electron Devices*, vol. ED-17, no. 1, pp. 38–47, 1970.

[4] G. Wachutka, "Rigorous Thermodynamic Treatment of Heat Generation and Conduction in Semiconductor Device Modeling," *IEEE Trans.Computer-Aided Design*, vol. 9, pp. 1141–1149, Nov. 1990.

[5] T. Grasser and S. Selberherr, "Fully-Coupled Electro-Thermal Mixed-Mode Device Simulation of SiGe HBT Circuits," *IEEE Trans.Electron Devices*, vol. 48, no. 7, pp. 1421–1427, 2001.

[6] W. VanRoosbroeck, "Theory of Flow of Electrons and Holes in Germanium and Other Semiconductors," *Bell Syst.Techn.J.*, vol. 29, pp. 560–607, 1950.

[7] C. Snowden, *Semiconductor Device Modelling*. Springer, 1989.

[8] T. Grasser, T. Tang, H. Kosina, and S. Selberherr, "A Review of Hydrodynamic and Energy-Transport Models for Semiconductor Device Simulation," *Proc.IEEE*, vol. 91, no. 2, pp. 251–274, 2003.

[9] D. Schroeder, *Modelling of Interface Carrier Transport for Device Simulation*. Springer, 1994.

[10] C. Fischer, *Bauelementsimulation in einer computergestützten Entwurfsumgebung*. Dissertation, Technische Universität Wien, 1994. http://www.iue.tuwien.ac.at.

[11] D. Scharfetter and H. Gummel, "Large-Signal Analysis of a Silicon Read Diode Oscillator," *IEEE Trans.Electron Devices*, vol. ED-16, no. 1, pp. 64–77, 1969.

[12] P. Deufhard, "A Modifi ed Newton Method for the Solution of Ill-Conditioned Systems of Nonlinear Equations with Application to Multiple Shooting," *Numer.Math.*, vol. 22, pp. 289–315, 1974.

[13] R. Bank and D. Rose, "Global Approximate Newton Methods," *Numer.Math.*, vol. 37, pp. 279–295, 1981.