DOPDEES/PMM: A System for Portable Model Description

Alp H. Gencer and Scott T. Dunham
Boston University, Department of Electrical and Computer Engineering
8 St. Mary's St., Boston, MA 02215, alp@bu.edu

ABSTRACT

We have developed a multi-purpose partial differential equation solver, DOPDEES, which is capable of solving initial value problems in one spatial dimension using a finite differences method. The system of partial differential equations is specified using a "dial-an-operator" paradigm, and the program uses Tcl as a front end. DOPDEES has commands for structure generation and modification, as well as features that make simulation of time-varying parameters (e.g. temperature ramps) possible. The solver is fast and suitable for use as a process simulator and model test bed in 1-D.

Process Modeling Modules (PMM) is a framework for platform-independent scriptization of commonly encountered models in the process modeling community. PMM frees the user from the burden of specifying all equations and parameters for commonly known models, but still retains the power to specify arbitrary equations and models. It also provides a framework for transferring models directly from the university to the industry, without the need for a vendor to implement those models in a commercial process simulator.

Keywords: Process modeling, partial differential equations, process simulator

DOPDEES

The shrinking dimensions of VLSI devices leads to more complex phenomena becoming important during fabrication. This requires a flexible environment for developing process models. The system should not restrict the user to predefined models, but rather should enable implementation of any system of continuum equations. DOPDEES is such an effort.

DOPDEES is a multi-purpose PDE initial value solver in one spatial dimension that uses a "dial-an-operator" paradigm for specification of the equation system [1]. The primary aim of the code is rapid development of continuum models and it may be used for a variety of systems. It is intended to be easy to use and easy to extend with new operators.

DOPDEES can be used in any system where the user wants to obtain solutions for coupled partial differential equations (PDEs). The program was originally written to solve diffusion/reaction problems and this is why it comes with operators suited for such systems. However, if the operators or functions the user needs are not included, it is relatively easy to add them. Another positive feature is that the user is able to select the integration engine, as different systems benefit from different numerical algorithms which associated trade-offs in speed and stability. There are also commands for grid generation and result extraction. The user interface of DOPDEES is in Tcl, which provides for powerful input scripting.

DOPDEES solves a given set of partial differential equations. The (one dimensional) space is assumed to be divided into chunks called *regions*, and it is assumed that different equations need to be solved in different regions (Fig 1). The solution variables are called *fields*, and it is assumed that for all fields equations of the type $\partial f/\partial t = \ldots$ exist that describe the PDEs. A field specified in one region doesn't exist in others, unless explicitly specified. Actually, the only communication between regions happens through boundary transfers.

The right hand sides of the partial differential equations are specified as a sum of *operators*: $\partial f/\partial t = \sum \mathrm{op}_i$ This approach is called *dial-an-operator*, since the user can choose the operators on the right hand side. The operators can make use of *functions*, which can be defined in terms of fields and defined parameters as well as other functions.

DOPDEES actually discretizes the user-specified partial differential equations and then solves the resulting set of ordinary differential equations (ODE) using a standard ODE solver, or solver "engine." This considerably reduces the code development time, while the use of well-known, bug-free code leads to stability in the program.

DOPDEES supports time-varying parameters, which enables the user to simulate systems like temperature ramps. The parameters and fields are handled alike using the same functions, the only difference being that a parameter is a single valued quantity in a region, whereas fields have different values at each spatial grid point. When functions are defined, the resulting quantity is single valued if all arguments are single valued, otherwise it is assumed to be multi-valued. This elimi-

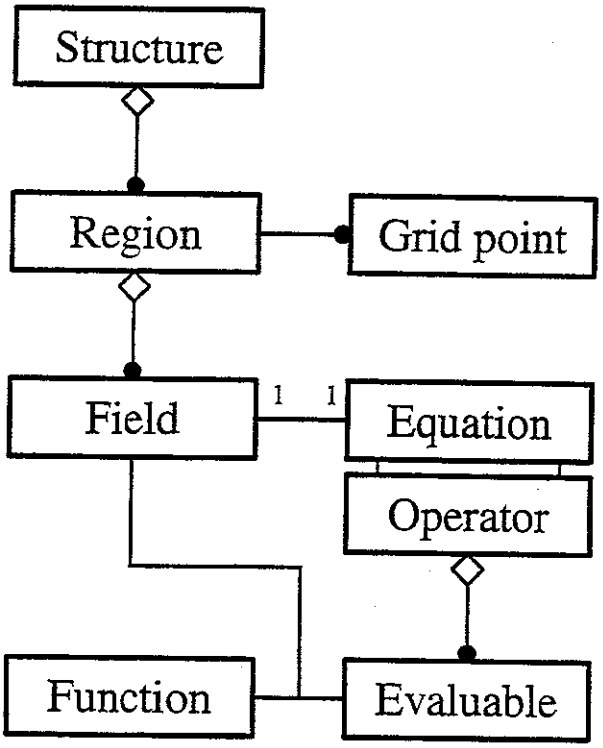

Figure 1: A structure in DOPDEES consists of regions, which each have a number of fields (solution variables) and a grid structure. Each field is associated with an equation of the type $\partial f/\partial t = \sum \mathrm{op}_i$.

nates the need for writing separate functions for parameters and fields, but still leads to efficient execution.

PROCESS MODELING MODULES (PMM)

The concept of Process Modeling Modules (PMM) developed from our realization that there was a gap to be filled between a process modeling software like SUPREM and a partial differential equation (PDE) solver like DOPDEES or Àlamode. Process simulators are not flexible enough to provide choice over the equations to be solved for; they are hard-coded and user control is limited to a fixed set of models and parameter values. On the other hand, when using a general PDE solver, it is necessary to specify every single equation and parameter in all input decks, even if some of the equations and parameters are considered to be well-known. In addition to the overhead involved, this may also lead to inconsistencies between users, or groups of users.

Thus, one form of software provides almost no flexibility, while the other requires the same work to be done over and over, often leading to errors. We attack this dilemma by writing reusable *modules* that can be incorporated into input decks of PDE solvers. This approach retains the flexibility of a PDE solver, while providing

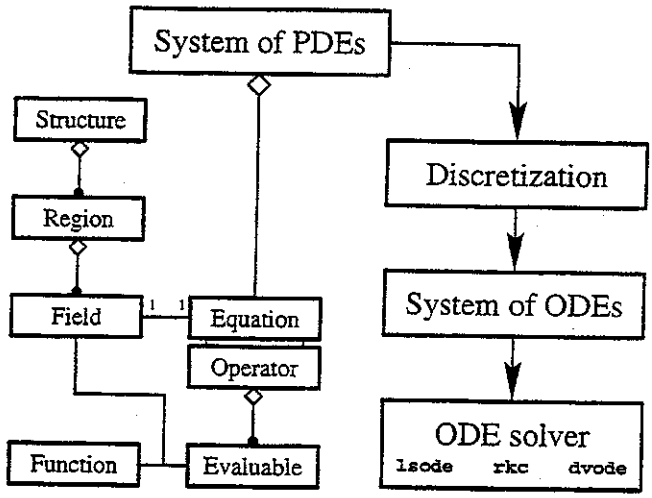

Figure 2: DOPDEES handles PDEs by discretizing them using finite differences and feeding them into a standard ODE solver.

reasonable defaults for models and parameters.

Several standard models have been implemented in PMM, as well as many models that have been developed at Boston University. Fig 3 shows available models as well as their interdependencies. PMM (Process Modeling Modules) uses a hierarchical model description, such that models can include other models, and ensures interoperability of models by consistency checks. Currently, PMM includes a collection of modules frequently encountered in VLSI process modeling:

- Dopant diffusion, simple and coupled with point defects.
- Point defect diffusion, generation and recombination.
- Dopant-dopant pairing.
- Two-stream diffusion in polysilicon.
- Clustering of point defects and precipitation of dopants.

PMM model descriptions are independent of the PDE solver used, thus providing a mechanism for transfer of models between simulation packages. In addition to DOPDEES, PMM has been interfaced with the Àlamode solver [2], which is part of the SUPREM OO7 system. We have also published a white paper describing the common scripting platform [3]. Avant! shows an effort to integrate PMM with their new process simulator. We believe that the use of the PMM concept will speed model development and facilitate transfer of those models between groups of users. The most direct impact would be rapid transfer of models from university research groups to industry, without the need to wait for implementation by commercial software vendors.

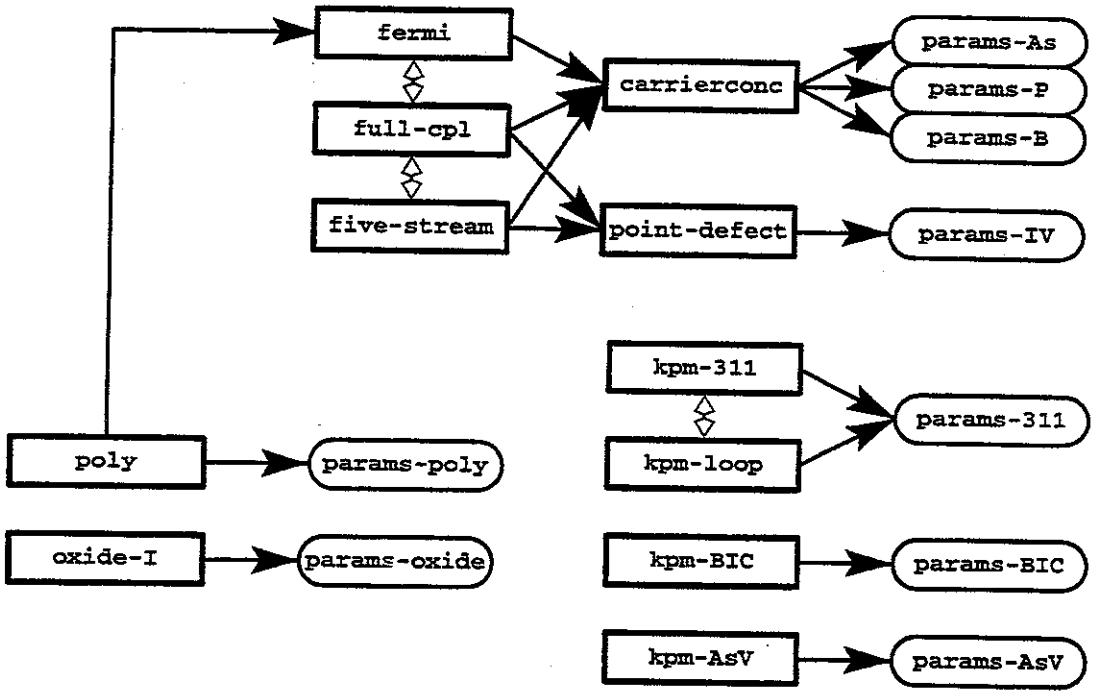

Figure 3: Models implemented in PMM and their interdependencies. Modules can inherit parameters from lower level modules and do consistency checks.

SIMULATION OF DOPANT DIFFUSION

As an example we will consider diffusion of two dopants, boron and arsenic, in silicon. The diffusivity of both boron and arsenic are dependent on the local Fermi level, and therefore they can affect each others diffusion through the net dopant concentration. At high concentrations, arsenic is know to become immobile and electrically inactive. We will consider a simple model where the effect of point defect interactions is not taken into account (the Fermi model) and the deactivation of arsenic is assumed to happen abruptly above the solid solubility. More rigorous models exists both for diffusion and deactivation and have been implemented in DOPDEES/PMM framework. Under conditions where the point defect concentrations are close to their equilibrium values, this model should provide reasonable results.

The set of PDEs in our system can be formulated as follows:

$$\frac{\partial C_{\mathrm{B}}}{\partial t} = \vec{\nabla} D_{\mathrm{B}} \left(\vec{\nabla} C_{\mathrm{B}} - C_{\mathrm{B}} \vec{\nabla} \ln(n/n_i) \right) \quad (1)$$

$$\frac{\partial C_{\mathrm{As}}}{\partial t} = \vec{\nabla} D_{\mathrm{As}} \left(\vec{\nabla} C_{\mathrm{As}}^{\mathrm{act}} + C_{\mathrm{As}}^{\mathrm{act}} \vec{\nabla} \ln(n/n_i) \right) \quad (2)$$

$$C_{\mathrm{As}}^{\mathrm{act}} = \min(C_{\mathrm{As}}, C_{\mathrm{ss}}) \quad (3)$$

$$\frac{n}{n_i} = \left(\frac{C_{\mathrm{As}}^{\mathrm{act}} - C_{\mathrm{B}}}{2n_i} \right) + \sqrt{\left(\frac{C_{\mathrm{As}}^{\mathrm{act}} - C_{\mathrm{B}}}{2n_i} \right)^2 + 1} \quad (4)$$

The diffusivities of boron and arsenic are assumed to depend on n/n_i in a standard manner. In addition to the silicon region, we also have an oxide region on top of the structure and boron is assumed to segregate preferentially to the oxide. Figure 4 shows the script ued in DOPDEES for model definition. Note that the operators need to specify to which field (i.e. equation) they belong to. Using PMM greatly simplifies the input file since it calls the pre-defined Fermi module (Fig. refdopdees-rpm). Note that the user can still specify an arbitary set of PDEs, for this case PDEs for boron segregation to and diffusion in the oxide, when using PMM modules. This ensures maximum flexibility while simplifying the input. Figure 6 shows the initial and final doping profiles in the structure.

REFERENCES

[1] B.J. Mulvaney, W.B. Richardson, and T.L. Crandle, *IEEE Trans. Comp. Aided Des.* 8, 336 (1989).

[2] D. Yergeau. *Alamode — A Layared Modeling Environment* (1996).

[3] D. Yergeau, R. Dutton, A.H. Gencer, and S.T Dunham. A model implementation environment to support rapid prototyping of new TCAD models: A case study for dopant diffusion. Technical report SEMATECH (1997).

```
set TEMPK [unit 900 C]

# Silicon atom concentration
set CSi [unit 5e22 cm-3]
# "Average" lattice spacing in Si
set ASi [expr pow($CSi,-0.333333)]

# Define operators
region select Oxide

# Diffusion of boron in oxide
set DB_Ox [arrhenius 1.83e-2 3.82 cm2/s]
op diff CB $DB_Ox CB

region select Silicon

# Active arsenic concentration
set CSS_As [arrhenius 2.22e22 0.47 cm-3]
set CAs_act [func min CAs $CSS_As]

# Carrier concentration in silicon
set NI [arrhenius [expr 3.9e16*pow($TEMPK,1.5)] \
    0.605 cm-3]
set nni [func carrierconc [func sum $CAs_act \
    [func prod -1 CB]] $NI]

# Diffusion of boron in Silicon
set DB_0 [arrhenius 0.037 3.46 cm2/s]
set DB_P [arrhenius 0.76  3.46 cm2/s]
set DB [func diffusivity $nni $DB_0 0 $DB_P 0]
op diff CB $DB CB

# Drift of boron
op diff CB [func prod -1 $DB CB] [func log $nni]

# Diffusion of arsenic
set DAs_0 [arrhenius 0.0666 3.44 cm2/s]
set DAs_M [arrhenius 12.8 4.05 cm2/s]
set DAs [func diffusivity $nni $DAs_0 $DAs_M 0 0]
op diff CAs $DAs $CAs_act

# Drift of arsenic
op diff CAs [func prod $DAs $CAs_act] [func log $nni]

# Segregate boron to the oxide
set mseg [arrhenius 1126 0.91]
set Ktr [func div [func sum $DB_0 $DB_P] $ASi]
op transfer Oxide CB CB [func div CB $mseg] CB $Ktr

# Run the simulator.
solver run 0 [unit 30 min]
```

Figure 4: DOPDEES input script for the solving the equation system for co-diffusion of boron and arsenic.

```
set TEMPK [unit 900 C]

# Define operators
region select Oxide

# Diffusion of boron in oxide
set DB_Ox [arrhenius 1.83e-2 3.82 cm2/s]
op diff CB $DB_Ox CB

region select Silicon

module fermi -sss -suprem -debug

# Segregate boron to the oxide
set mseg [arrhenius 1126 0.91]
set Ktr [func div [func sum $D(B,0) $D(B,P)] $ASi]
op transfer Oxide CB CB [func div CB $mseg] CB $Ktr

# Run the simulator.
solver run 0 [unit 30 min]
```

Figure 5: DOPDEES input script for the solving the same equation system, but with the Fermi model implemented using a PMM script.

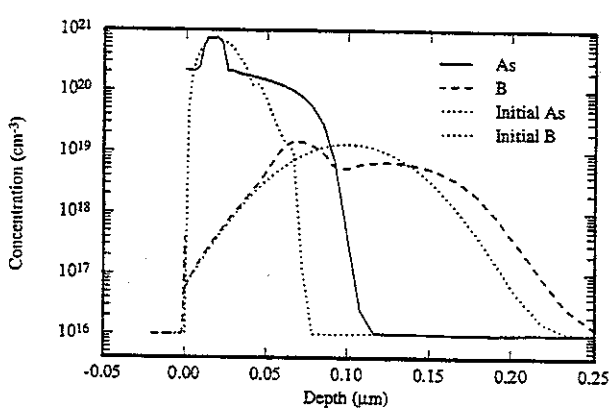

Figure 6: The initial dopant profiles and final profiles after a 30 min anneal at 900°C according to simulation using the Fermi model.