Process Modeling Modules

User's Guide and Reference Manual

Alp H. Gencer

Version: 97.221 (August 09, 1997)

Contents

Chapter 1

Introduction

The concept of Process Modeling Modules (PMM) came out when we realized 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 Alamode. Process modelers are not flexible enough to let you choose the equations you want to be solved for, they are hard-coded and you can only change some parameters. On the other hand, when using a PDE solver, you have to specify every single equation and parameter in all input decks, even if some equations and parameters are considered to be well-known.

Thus, one form of software gave you no flexibility, while the other required you to do the same work over and over, giving way to bugs. We attack this dilemma by writing reusable *modules* that can be incorporated into input decks of PDE solvers. This will keep the flexibility of a PDE solver, while providing reasonable defaults for models and parameters.

The resulting software, PMM, can be used both for DOPDEES and Alamode. Alamode users should adopt a few new syntactical rules when specifying their own equations on top of the ones provided by PMM. Other than that we hope that PMM will be easy to learn and use.

1.1 Installing PMM

All you need to do for installing PMM is to unpack the compressed archive file and add a line like the following to your `~/.dopdeesrc` or `~/.alamoderc` file:

```
source /home/user/PMM/scripts/init.tcl
```

Of course, you'll need to change the path of PMM to wherever you have installed it. In the current version of Alamode the `alamoderc` file is sourced only when the program is invoked interactively. Thus you may have to source it explicitly at top of your input deck (or ask Alamode supporters for a patch).

Included with PMM there is a separate package called UNITS, which has commands for handling units and unit systems in Tcl. Please read the documentations for UNITS in the directory `PMM/UNITS`.

1.2 Using PMM

In an input deck utilizing PMM, you should have the following structure:

1.2.1 Initialization

This is where you do your input grid specification, either using explicit commands or by reading from an external structure file. Note that this section *must* come before module inclusions, since the way the modules set up equations depend on the current structure. If no structure is loaded, no equations will be defined!

1.2.2 Parameter modification

If you want to modify some parameters in some modules, you should do it before including the modules. If you do not wish to modify any parameters, you may skip this section. The parameter you wish to modify is usually in a module the name of which starts with `params-`... To modify that parameter, first load the parameter module with a command like:

```
module params-... -opt1 -opt2 ...
```

and then set the value of the parameter to whatever value you wish. When you include a module that will use that parameter, it will use the modified value.

1.2.3 Module inclusion

The `module` command is used to include a module. A module defines equations, and includes modules that define the necessary parameters. The syntax of the `module` command is:

```
module name -opt1 -opt2 ...
```

Here `name` is the name of the module, and `-opt`s are options to the module. All modules expect certain fields to be defined, with certain names (see appendix). Options vary from module to module. A module defines equations that will be valid only in the *current region.*

1.2.4 Defining your own operators

As mentioned previously, you may define operators and equations in addition to the ones provided by the modules (but you don't have to). If you want to do so, you should do it at this point. DOPDEES users may simply specify new operators with the `op` command. For compatibility, Alamode users, too, may use DOPDEES' `op` command. The defined operators must have exactly the same syntax and order of arguments as in DOPDEES. The operator will be applied to, as in DOPDEES, the current region, and field names shouldn't have the "InRegion" tags, because all fields are assumed to be in the current region. Read the DOPDEES manual for more info about the operator syntax.

Alamode users who are not familiar with DOPDEES may use Alamode syntax for defining operators. But then, they must attach their new operator to an operator list. The syntax will look like:

```
set myop [operator .... ]
lappend OPLIST(FieldInRegion) $myop
```

Here, `myop` is set to the new operator, `Region` is the name of the region this operator is valid, and `Field` is the name of the field which stands on the left hand side of the equation in which this operator is found. That is, this operator is attached to the list of operators for the field `FieldInRegion`. `OPLIST` is a global variable keeping track of the list of operators. Please preserve the case of the word `In` between the field and the region.

Boundary operators should be attached to another list, BOPLIST, as follows:

```
set mybop [operator .... ]
lappend BOPLIST(Field:Region,Interface) $mybop
```

Here, `Field` and `Region` stand for the FieldInRegion which will be solved for in the equation this operator is attached to. `Interface` is the name of the interface this operator will be active (e.g. `Region1/Region2` or `ExposedRegion1`). Note the colon between `Field` and `Region` and the comma between `Region` and `Interface`.

1.2.5 Running the PDE solver

DOPDEES users may run the PDE solver with familiar `solver` commands. Alamode users first have to define a *model* before running the simulator. There is no need to use Alamode's `equation`, `systemInRegion` and `model` commands, since all operators have already been collected in lists. Instead PMM provides a `setmodel` command, which has the following syntax:

```
setmodel name
$name -stopTime 10 ...
$name -transientSolve
```

This command will make equations out of operators, systems out of equations and a model out of systems, and set `name` to the defined model. Solution parameters may be specified in the usual fashion using `name` as a key, as in the example above.

Chapter 2

Defined modules

Following is a list of defined modules. All modules accept certain options after the command name. Modules usually include other modules, and options are passed down the hierarchy. Thus, you may specify any valid options for the child modules when calling the parent module.

A common option for all modules is `-debug`, which results in extra debugging information, such as the values of used parameters, to be printed on the screen.

A module cannot be included twice in the same region. Trying to do so results in a warning if `-debug` is specified. Some modules may not be used concurrently, and produce an error is such a case.

2.1 Diffusion modules

2.1.1 The `fermi` module

The `fermi` module is used for diffusion of dopants where the diffusivity is a function of the Fermi level. It is analogous to the "fermi" model in SUPREM. Options specific to `fermi` module are:

- `-eff`: Use an electric field factor to approximate the drift term of the flux. The electric field factor is a factor multiplying the diffusivity with a number between 1 and 2 depending on the position of the Fermi level. It gives physically correct answers only when there is a single dopant.

Included modules: `carrierconc`

2.1.2 The `full-cpl` module

The `full-cpl` module is the implementation of fully coupled diffusion model of dopants and point defects. It is analogous to the "full.cpl" model in SUPREM. There are no options specific to `full-cpl` module.

Included modules: `carrierconc, point-defect`

2.1.3 The `five-stream` module

The `five-stream` module is the fully coupled diffusion model of dopants and point defects, using five species for diffusion (dopant, dopant-point defect pairs and point defects). It is safer than the `full-cpl` module, because `full-cpl` module assumes that the pairing reaction is fast and the number of pairs is always much smaller than the number of dopant atoms, which may not be true under circumstances like ion implant annealing. There are no options specific to `five-stream` module.

Included modules: `carrierconc`, `point-defect`

2.1.4 The `carrierconc` module

The `carrierconc` module is responsible for calculating the net active carrier concentration and diffusivities of dopants. It is usually used in conjunction with one of the diffusion modules above. Valid options are:

- `-sss`: Use a simple solid solubility cut-off model to calculate the active portion of the dopant profile. It is important NOT to specify `-sss` when using one of the dopant precipitation modules (e.g. `kpm-BIC` and `kpm-AsV`).
- `-ddp`: Use dopant-dopant pairing module, where dopants may become inactive by pairing with each other (e.g. B-As pairing).
- `-hcd`: Include high concentration diffusivity enhancement effects for As and P. Both dopants show a strong increase in diffusivity above a threshold concentration.

Included modules: `params-const`, `fields`, `params-B` (if boron is present), `params-As` (if arsenic is present), `params-P` (if phosphorus is present) and `params-ddp` (if `-dpp` is specified).

2.1.5 The `point-defect` module

The `point-defect` module is used for diffusion and recombination of point defects (interstitials and vacancies). Valid options are:

- `-surfrec`: Use a constant surface recombination velocity for point defects.

Included modules: `params-IV`, `fields`

2.2 Film modules

2.2.1 The `poly` module

The `poly` module implements two-stream model of diffusion in poly-silicon. Dopants are assumed to be diffusing within a grain and grain boundaries. Dopants segregate from grains to grain boundaries.

Included modules: `fermi`, `params-poly`, `fields`

2.2.2 The `oxide-I` module

The `oxide-I` module implements the segregation of interstitials to the oxide, and diffusion of them within the oxide. This model effectively results in an time-varying surface recombination velocity for interstitials. Note that the segregation operator must be specified explicitly in Alamode.

2.3 KPM modules

The following modules are based on the Kinetic Precipitation Model (KPM), which is described in detail in "KPM User's Guide". In order to use these modules, the copy of DOPDEES or Alamode you are using must have been compiled with the KPM library.

2.3.1 The `kpm-311` and `kpm-loop` modules

The `kpm-311` and `kpm-loop` modules implement the formation and growth of {311} defects and their transformation into dislocation loops. If `kpm-311` is used, it is assumed that only {311} defects form. If `kpm-loop` is used, it is assumed that after growing to a certain size {311} defects transform into dislocation loops. Thus, `kpm-loop` includes {311} defect kinetics, as well. The two modules cannot be used together. Valid options:

- `-fkpm`: Use the Full Kinetic Precipitation Model instead of the Reduced Kinetic Precipitation Model. May only be specified in DOPDEES.
- `-diff`: Use a small diffusivity for moments to avoid instabilities.

Included modules: `params-311`

2.3.2 The `kpm-BIC` module

The `kpm-BIC` module implements the formation of boron-interstitial clusters (BICs). It is assumed that boron incorporates interstitials while precipitating, resulting in an effectively lower solid solubility under interstitial supersaturation.

Included modules: `params-B`, `params-BIC`

2.3.3 The `kpm-AsV` module

The `kpm-AsV` module implements the formation of arsenic vacancy clusters (AsVs). It is assumed that arsenic injects interstitials while precipitating, resulting in an interstitial supersaturation. The model is similar to `kpm-BIC` model, with exchanged roles of point defects.

Included modules: `params-As`, `params-AsV`

2.4 Parameter modules

The `params-B`, `params-As` and `params-P` modules

The `params-B`, `params-As` and `params-P` modules define parameters for dopants, such as their charge, diffusivities, coupling coefficients with point defects, solid solubilities, etc. Valid options are:

- `-suprem`: Use diffusivity values from SUPREM's modelrc file, rather than using BU values.

The `params-IV` module

The `params-IV` module defines parameters for point defects such as their diffusivities, equilibrium concentrations and bulk and surface recombination velocities. Valid options are:

- `-suprem`: Use diffusivity and equilibrium concentration values from SUPREM's modelrc file, rather than using BU values.
- `-bracht`: Use diffusivity and equilibrium concentration values from Bracht *et al.*, rather than using BU values. `-bracht` overrides `-suprem`.

The `params-ddp` module

The `params-ddp` module has parameters for dopant-dopant pairing. Dopants of opposite charges may form immobile clusters. Currently, only (B-As) and (B-P) pairs are supported.

The `params-poly` module

The `params-poly` module defines parameters for the two-stream model of diffusion in poly. It is usually used in conjunction with the `poly` module.

The `params-311` module

The `params-311` module defines parameters for {311} defect and dislocation loop formation and growth. Used in conjunction with `kpm-311` and `kpm-loop` modules.

The `params-BIC` module

The `params-BIC` module defines parameters for boron interstitial cluster formation and growth. Used in conjunction with `kpm-BIC` module.

The `params-AsV` module

The `params-AsV` module defines parameters for arsenic vacancy cluster formation and growth. Used in conjunction with `kpm-AsV` module.

The `params-const` module

The `params-const` module has some fundamental constants for silicon.

Appendix A

Naming conventions in PMM

Most modules in PMM expect certain fields and regions with certain names. This allows the scripts to determine the type of a field and region. We will first describe the default naming convention and then explain how it can be changed.

A.1 Default names in PMM

Single species concentrations: All concentrations start with a capital `C` and is followed by the name of the chemical species it stands for. Thus `CB` is boron concentration and `CAs` is arsenic concentration. The species `I` is interstitials and `V` is vacancies, making `CI` interstitial concentration and `CV` vacancy concentration.

Pair concentrations: Concentrations of pairs of species is obtained in a similar way, by adding two species names to the capital `C`. The order is important here. If one of the species is a dopant and the other a point defect, the dopant comes first. If both species are dopants, the acceptor comes first. Thus `CBI` is the concentration of boron-interstitial pairs, `CAsV` is the concentration of the arsenic-vacancy pairs and `CBAs` is the concentration of boron-arsenic pairs.

Grain boundary concentrations: Concentrations of species within a poly silicon grain boundary are specified similar to above, but by appending a `_gb` to the end. Thus `CAs_gb` is the concentration of arsenic within a grain boundary. This is used only by the `poly` module.

KPM moments: KPM moments have names starting with `M_` and ending with the name of the precipitate. Thus, `M_P` stands for the moments of phosphorus precipitates and `M_BIC` stands for boron-interstitial clusters. The only exception is {311} defects and dislocation loops, which have the name `M_I`.

Region names: Region names are only used by PMM modules that have a segregation operator, and therefore need to check the type of the neighboring region. Only the first couple of letters are checked in the region names, thus the user has the freedom to use long names, or differentiate between, say, two silicon regions. Here is a list of default region names:

Name starts with	**Type of region**
`Si`	Silicon
`Ox`	Silicon Dioxide (SiO_2)
`Poly`	Poly-silicon
`Int`	Generic interface region

A.2 Changing default names

Since names are a matter of preference, different users might want to use different naming conventions. In such a case you'll need to follow a two simple steps to use your own convention:

Create a new names file: You should copy the module file `names-default.tcl` to a different name, and then make modifications on it for the names of your liking. Try to be as consistent as possible for your own sake.

Source the file: Call the file you created on top of every file you want to use it with a `module` command. If you really want to use it with every file you create, you may add the `module` command to your `~/.dopdeesrc` or `~/.alamoderc` file AFTER the call to PMM initialization.