Sentaurus™ Topography 3D User Guide

Version N-2017.09, September 2017

SYNOPSYS®

Copyright and Proprietary Information Notice

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.
All other product or company names may be trademarks of their respective owners.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
690 E. Middlefield Road
Mountain View, CA 94043
www.synopsys.com

Contents

Contents

About This Guide

The Synopsys Sentaurus™ Topography 3D tool is a three-dimensional simulator for evaluating and optimizing critical topography-processing steps such as etching and deposition.

Related Publications

For additional information, see:

- The TCAD Sentaurus release notes, available on the Synopsys SolvNet® support site (see Accessing SolvNet on page xii).
- Documentation available on SolvNet at https://solvnet.synopsys.com/DocsOnWeb.

Conventions

The following conventions are used in Synopsys documentation.

Convention	Description
Blue text	Identifies a cross-reference (only on the screen).
Bold text	Identifies a selectable icon, button, menu, or tab. It also indicates the name of a field or an option.
`Courier font`	Identifies text that is displayed on the screen or that the user must type. It identifies the names of files, directories, paths, parameters, keywords, and variables.
Italicized text	Used for emphasis, the titles of books and journals, and non-English words. It also identifies components of an equation or a formula, a placeholder, or an identifier.

Customer Support

Customer support is available through the Synopsys SolvNet customer support website and by contacting the Synopsys support center.

Accessing SolvNet

The SolvNet support site includes an electronic knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. The site also gives you access to a wide range of Synopsys online services, which include downloading software, viewing documentation, and entering a call to the Support Center.

To access the SolvNet site:

1. Go to the web page at https://solvnet.synopsys.com.
2. If prompted, enter your user name and password. (If you do not have a Synopsys user name and password, follow the instructions to register.)

If you need help using the site, click **Help** on the menu bar.

Contacting Synopsys Support

If you have problems, questions, or suggestions, you can contact Synopsys support in the following ways:

- Go to the Synopsys Global Support Centers site on synopsys.com. There you can find e-mail addresses and telephone numbers for Synopsys support centers throughout the world.
- Go to either the Synopsys SolvNet site or the Synopsys Global Support Centers site and open a case online (Synopsys user name and password required).

Contacting Your Local TCAD Support Team Directly

Send an e-mail message to:

- support-tcad-us@synopsys.com from within North America and South America.
- support-tcad-eu@synopsys.com from within Europe.
- support-tcad-ap@synopsys.com from within Asia Pacific (China, Taiwan, Singapore, Malaysia, India, Australia).
- support-tcad-kr@synopsys.com from Korea.
- support-tcad-jp@synopsys.com from Japan.

Acknowledgments

Third-party software forms part of Sentaurus Topography 3D:

- log4cxx – For more information about licensing, go to http://logging.apache.org/log4cxx/license.html.
- Trilinos Sacado – Go to https://trilinos.org/packages/sacado/.

  Project license: © 2006 Sandia Corporation. Distributed under the GNU Lesser General Public License, version 3.0.
- lp_solve 5.5 modified by Synopsys. The modified source is available from SolvNet.

  Go to https://solvnet.synopsys.com/redauthftp/cafe/lp_solve_5.5.

  Project license: Distributed under the GNU Lesser General Public License, version 2.1.

The Sentaurus Topography 3D software uses the MPICH message passing interface (MPI) package, which requires the following copyright notice:

Copyright Notice
+ 2002 University of Chicago

Permission is hereby granted to use, reproduce, prepare derivative works, and to redistribute to others. This software was authored by:

Mathematics and Computer Science Division
Argonne National Laboratory, Argonne IL 60439

(and)

Department of Computer Science
University of Illinois at Urbana-Champaign

GOVERNMENT LICENSE

Portions of this material resulted from work developed under a U.S. Government Contract and are subject to the following license: the Government is granted for itself and others acting on its behalf a paid-up, nonexclusive, irrevocable worldwide license in this computer software to reproduce, prepare derivative works, and perform publicly and display publicly.

DISCLAIMER

This computer code material was prepared, in part, as an account of work sponsored by an agency of the United States Government. Neither the United States, nor the University of Chicago, nor any of their employees, makes any warranty express or implied, or assumes any

legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights.

CHAPTER 1 Getting Started

This chapter provides an overview of the functionality of Sentaurus Topography 3D, the required input files, and the generated output files.

Introduction

Sentaurus Topography 3D is a three-dimensional simulator for evaluating and optimizing critical topography-processing steps such as etching and deposition. Two- and three-dimensional structures composed of an arbitrary number of different materials can be handled.

The initial structure can either be created using simple geometric etching and deposition operations, or be read in from TDR files. The resulting structures can be saved in different TDR file formats and can be used for further processing.

Starting Sentaurus Topography 3D

Sentaurus Topography 3D can be invoked and used in one of two modes.

From Sentaurus Workbench

Sentaurus Topography 3D is fully integrated in Sentaurus Workbench. On the command line, type the following command to start Sentaurus Workbench:

```
swb
```

Inside Sentaurus Workbench, you can add Sentaurus Topography 3D to any process flow.

From the Command Line

Sentaurus Topography 3D can be started directly from command line using:

```
sptopo3d commandfile.cmd
```

where `commandfile.cmd` is a file that contains all the commands for the simulation. See Chapter 4 on page 65 for detailed descriptions of commands.

Input Files

Sentaurus Topography 3D uses the following input files depending on the initial information available to start a simulation.

Command File

The command file is a text file that contains a sequence of commands that direct the simulation (see Chapter 4 on page 65). The command file can also contain Tcl commands.

TDR File

The initial structure for a simulation can be read from a TDR file containing a 2D or 3D boundary, or a 3D GC structure. The 3D boundary must have a rectangular base normal to the positive z-axis.

PMC File

The initial structure for a PMC simulation can be read from a PMC file.

Output Files

The names of output files created by Sentaurus Topography 3D follow a naming convention. The base name of the output files is defined by the base name of the command file. Different extensions are used, depending on the type of the output file.

Standard Output and Log File

All the commands specified by the input file as well as the messages and output indicating the progress of Sentaurus Topography 3D are, by default, displayed on-screen. Any error or warning condition encountered is also displayed.

The log file has the extension `.log` and contains the same information as the standard output. The file should be examined after completion of a Sentaurus Topography 3D simulation.

TDR Files

Different TDR file formats can be used to save information about structures modified by a Sentaurus Topography 3D simulation. You can use the `save` command (see save on page 199), for example, to:

- Save structure boundary information at any stage of a simulation.
- Save the resulting structure of a simulation that used the particle Monte Carlo method.

Furthermore, you can use the `extract` command to save 1D and 2D geometric information from a structure into TDR files (see extract on page 166).

PMC File

A PMC structure can be saved to a PMC file, which can be subsequently read in to define the initial structure of a new PMC simulation.

CHAPTER 2 Simulation Details

This chapter provides details about simulations performed using Sentaurus Topography 3D.

Sentaurus Topography 3D Computational Model

Sentaurus Topography 3D simulates deposition, etching, and simultaneous etching and deposition processes. Except for the spin-on-glass deposition model, two simulation methods are used to accomplish this task: the level-set method and the particle Monte Carlo (PMC) method.

Some physical models can be run only using the level-set method and others can be run only using the PMC method. To the former class belong the following models:

- Rate formula module (RFM)–based models (see Chapter 6 on page 215).
- Built-in models:
  - Etching and simultaneous etching and deposition models: `crystal`, `dry`, `etchdepo`, `etchdepo2`, `hdp`, `ion_enhanced`, `ionmill`, `rie`, `rie2`, `simple`, and `wet`.
  - Deposition models: `crystal`, `electrodeposition`, `electroplating`, `hdp`, `hdp2`, `lpcvd`, `pecvd`, `pvd`, and `simple`.

To the latter class belong the reaction models, that is, those defined in terms of reactions rather than rate formulas (see add_reaction on page 88 and define_model on page 124).

For the spin-on-glass deposition model (named `spin_on`), the surface profile evolution is determined by solving a partial differential equation describing the motion of a fluid driven by surface tension, centrifugal forces, and evaporation (see Spin-on-Glass Deposition Model on page 61).

Computational Model When Using the Level-Set Method

When using the level-set method to simulate a model, the exposed surface (see Boundary Types on page 13) is discretized into a set of surface elements of finite extents, and the processing time is divided into discrete time steps.

At each time step, the velocity at each surface element is determined according to the processing conditions, and the surface is updated. The velocity is computed using the *rate formula*, which is specific to each model, because it expresses with a mathematical relation the deposition or etching mechanisms relevant to the modeled process.

For example, the rate formula of the LPCVD model (Eq. 16, p. 46) states that the deposition rate at any surface point is proportional to the number of neutral particles hitting the surface per unit time and area, including both those reaching the surface directly from the reactor and those that are reemitted from other surface points.

At each time step, all the quantities involved in the rate formula are computed, and the velocities are determined accordingly at each surface point. Among such quantities, a crucial role is usually played by the number of particles hitting the surface per unit time and area, as described for the LPCVD model.

For level set–based models, Sentaurus Topography 3D uses the concept of *flux* to model the flow of particles and their interactions with the surface. In this context, a flux denotes an abstract continuum representation of one or more chemical species having a similar role in the modeled process. Therefore, *abstract* means that different chemical species (which are really involved in the process) can be included in the simulation as a single flux. The term *continuum* refers to the fact that species are not modeled as sets of discrete particles, but by a continuum flow that is characterized by statistical properties such as the angular distribution or the energy distribution. On this basis, a flux can be considered a continuum of abstract particles.

As explained in Modeling of Fluxes and Related Physical Effects on page 27, there are different kinds of flux for level set–based models:

- Fluxes with isotropic angular distribution, which are called neutrals.
- Fluxes with an anisotropic angular distribution, which are called ion fluxes.

The different interactions between a flux and the surface are *physical effects*. Sentaurus Topography 3D supports the physical effects of reemission for neutral fluxes, and reflection, sputtering, and deposition of sputtered material for ion fluxes.

As mentioned, a flux provides a continuum representation of a set of species. More precisely, every flux is characterized by its flux density, which describes the statistical distribution of the velocities of the particles as emitted from their source.

The scalar quantity called the *direct flux* denotes the integrated flux density on a point of the surface. The direct flux is the integral of the flux density over the visible range of the surface point, or the number of particles per unit area and time that reaches the surface at this point before having interacted with the surface.

NOTE When using the level-set method, flux densities are always normalized such that the direct flux is equal to 1 on a completely visible flat surface.

A secondary flux that results from an interaction with the surface is referred to as an *indirect flux*. For example, the number of sputtered particles per unit area and time is referred to as the *sputtered flux*.

Consequently, for each flux, there is one direct flux and possibly several indirect fluxes, according to the modeled physical effects. All these fluxes can be used in the rate formula.

At each time step, all the direct and indirect fluxes involved in the rate formula are computed at each surface point. The computation of all the direct and indirect fluxes requires knowledge about the flux densities and the shape of the surface.

Accordingly, when using the level-set method, the computational flow of Sentaurus Topography 3D can be described as a loop over the following three stages:

1. If the model depends on fluxes, compute the direct flux and all indirect fluxes for each flux at each surface point.
2. Compute the velocity at each surface point according to the rate formula.
3. Move the surface according to the velocities provided by the rate formula.

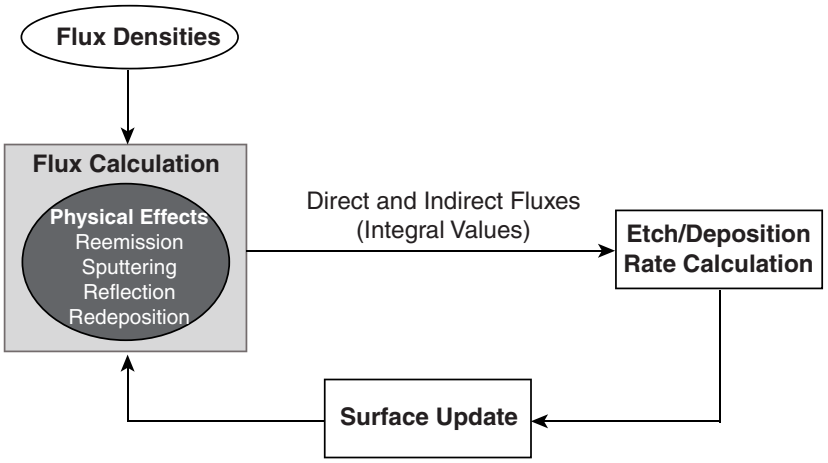


Figure 1 Conceptual model of Sentaurus Topography 3D when using the level-set method

Computational Model When Using the PMC Method

The PMC method uses a stochastic approach to compute the time evolution of a structure. You define interactions between the plasma species and the wafer species in terms of surface reactions (see add_reaction on page 88, add_source_species on page 98, and define_model on page 124).

Computational Model for Spin-on-Glass Deposition

The time evolution of the film profile is determined by solving a partial differential equation describing the motion of a fluid driven by surface tension, centrifugal forces, and evaporation (see Spin-on-Glass Deposition Model on page 61).

The processing time is divided into discrete time steps, and the surface is sampled at a set of points. At each time step, the height of the film is computed at each sampling point, and the film profile is updated until the total process time is reached.

Discretization Size and Accuracy

Numeric simulation methods, in general, only produce approximations to the exact results. Usually, the quality of the approximation can be improved by increasing the computational effort. For the level-set method and the method used to solve the spin-on-glass deposition model, the required memory and CPU time are determined mainly by the spatial and time discretization. For the PMC method, the required memory and the CPU time depend mainly on the spatial discretization, the number of simulated particles coming from the reactor, and the number of simulated reactions.

In Sentaurus Topography 3D, the `spacing` parameter of the `deposit` and `etch` commands sets the grid discretization used to compute the structure evolution (see deposit on page 147 and etch on page 154). The `flux` parameter of the `define_species_distribution` command (see define_species_distribution on page 133) as well as the parameter `time` of the `etch` command (see etch on page 154) determine the number of simulated particles.

Decreasing the size of the spatial discretization increases the required memory and the CPU time because there are more grid points to store and process. When using the level-set method, due to the Courant–Friedrichs–Lewy (CFL) condition [1], the maximum time-step size must be reduced. This causes a further increase of the CPU time.

Usually when setting up an etching or a deposition simulation, it is necessary to find a compromise between the required accuracy and the limited available computational resources. A good understanding of the relationship between the discretization sizes and the resulting accuracy is important not only when setting up a simulation, but also when interpreting the results of a simulation.

Sentaurus Topography 3D allows you to set the spatial discretization for each coordinate axis of the computational grid. For models using the level sets, the spatial discretization along each dimension of the structure can be set. For the spin-on-glass deposition model, since the vertical dimension does not need to be discretized, only the spatial discretizations along dimensions orthogonal to the vertical can be set. You might choose the discretization in a certain direction

to be much coarser than for the other directions to reduce simulation time. This is not recommended, however, for various implementation-specific reasons. It is recommended that the discretization for all axes does not vary by orders of magnitude. It has been found that a factor of three between the discretization of the axes produces good results. Currently, for models using the PMC method, the spatial discretization along all dimensions must be the same (see deposit on page 147 and etch on page 154).

Discretization Size and Accuracy When Using the Level-Set Method

In general, the spatial accuracy that can be achieved when using the level-set method is of the order of the spatial discretization. Therefore, you must decide which features of the input structure and the expected output structure are important and choose the spatial discretization accordingly.

For models using the level sets, subresolution accuracy can be achieved in those parts of the simulation domain where the surface shows little variation. For example, when using a deposition model for which the rate does not depend on any property of the surface, the thickness of a deposited layer can be determined with higher accuracy than the spatial discretization size in those areas of the surface that are at a distance from the corners of the surface.

For etching, the situation is more complicated, and some additional considerations must be taken into account. Generally, several materials are etched simultaneously with different etching rates. Sentaurus Topography 3D determines the material-dependent etching rate on the exposed surface.

To underetch masking materials properly and to avoid certain artifacts, special treatment is necessary at the interface of two different materials. At an interface, the highest etching rate of the materials next to the interface is used. This can lead to problems with very thin layers for which the etching rate is lower than for the neighboring materials, because these thin layers can be etched away in one time-step. To avoid this, it is suggested that the spatial discretization is set to slightly less than the thickness of the thin layer.

As described in Computational Model When Using the Level-Set Method on page 5, at each time step, the exposed surface (see Boundary Types on page 13) of the processed structure is discretized into surface elements of finite extents, whose sizes depend on the level-set grid spacing.

To determine the values of the material-dependent parameters involved in the rate formulas of etching models and simultaneous etching and deposition models, it is necessary to compute the material or the materials through which each element of the exposed surface passes.

Since the surface discretization depends on the level-set grid, but the material information comes from the input structure, it is possible that some surface elements pass through multiple materials of the structure. This might happen, for example, when an interface between two materials of a structure is not aligned with the level-set grid.

In such cases, the default behavior of Sentaurus Topography 3D is to assign to each surface element the material in which its centroid is contained. When better accuracy is needed, you can change this behavior by setting `region_query_accuracy=subresolution` in the `etch` command (see etch on page 154). When `region_query_accuracy=subresolution`, the etching or deposition rate of each surface element is computed by taking into account all materials through which a surface element passes.

Sentaurus Topography 3D Simulation Flow

In Sentaurus Topography 3D, independently of the used numeric method, the simulation of a physical process requires:

- A model describing the process of interest.
- A structure to be processed.

When a process step simulation is complete, you can write the resulting structure to a TDR file (see Output Files on page 2 and save on page 199) or perform measurements on it using the `extract` command (see extract on page 166).

A model and its parameter values are specified by defining a machine, as detailed in Machines on page 10. A machine can be applied to simulate a process running on any structure.

You can create a structure by either using Sentaurus Topography 3D commands or loading a TDR file (see Initial Structure Generation on page 11 for an overview of the commands available to define the initial structure of a simulation and the related concepts).

Machines

Sentaurus Topography 3D can simulate different topography processes: deposition, etch, and lithography. Each process is represented by a machine that groups all of the parameters necessary to perform a simulation (see define_deposit_machine on page 101, define_etch_machine on page 109, or define_litho_machine on page 120).

Machines must be defined before their use, and a unique name for a given machine type (either deposition, etch, or lithography) must be assigned. When only one machine is defined, the name definition can be omitted and the simulator assigns the default machine name. If more

than one machine of the same type is needed, a unique name must be set for each of the defined machines.

The definition of different machines with a corresponding unique name allows the creation of a machine library. A machine defined in such a library can be referenced at any time during the simulation.

Initial Structure Generation

The input structure to be processed in Sentaurus Topography 3D can be obtained in either of the following ways:

- Loading a TDR boundary file, for example, `define_structure_file=input.tdr`.
- Creating a structure directly in Sentaurus Topography 3D, using a combination of simple geometric etch and deposition steps.

The second option is useful to create simple input structures without using external tools, as discussed in the next sections.

Creating the Initial Structure

When a structure is created with Sentaurus Topography 3D, an initial cuboid structure must be defined. This structure constitutes the base for further etch or deposition processes. The following command creates a unit cube made of silicon (see define_structure on page 139):

```
define_structure material=Silicon point_min={0 0 0} point_max={1 1 1}
```

Modifying the Initial Structure

The initial cuboid region can be modified either by adding materials using the `deposit` command or by removing parts of the structure using the `etch` command.

The following commands etch a rectangular trench out of the initial cuboid defined above:

```
define_shape type=cube point_min={0 0.3 0.5} point_max={1 0.7 1.0} \
   name=etch_shape

etch shape=etch_shape
```

The first command defines a cuboid shape that overlaps the original unit cube (see define_shape on page 129). The second command uses the defined cube `etch_shape` to remove material from the unit cube (see etch on page 154).

Similarly, it is possible to deposit a cuboid region:

```
define_shape type=cube point_min={0.4 0 0.5} point_max={0.6 1 1.25} \
   name=depo_shape

deposit shape=depo_shape material=Tungsten
```

The above commands define a cube, which is deposited over the trench created earlier. In the `deposit` command, the deposited material must be specified (see deposit on page 147). Parts of `depo_shape` that overlap the existing structure will not be added.

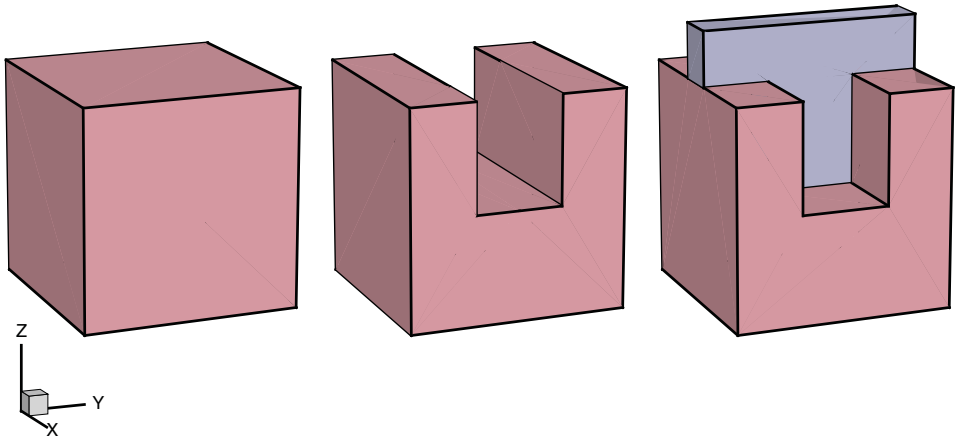


Figure 2 Initial structure generation: (*left*) initial cuboid structure, (*middle*) trench etched from the initial structure, and (*right*) cuboid deposited on the trench

The structure that has been created could now be saved for visualization by using the command `save` (see save on page 199). It can also be used as the initial structure for physics-based etch or deposition steps. Figure 2 on page 12 shows the results of the commands in this section.

NOTE When using level set–based models or the spin-on-glass deposition model, Sentaurus Topography 3D does not require the presence of any gas region on top of the initial structure. However, if the initial structure contains gas regions, gas will be present also in the final structure.

When using a reaction model based on the PMC method, where deposition is allowed, the computational domain must be sufficiently large to contain also the material that will be deposited during the simulation (see Boundary Types on page 13). This goal can be achieved by adding a gas region on top of the initial structure.

A gas region can be added to an existing structure using the `fill` command (see fill on page 185). This feature is useful also if a structure produced by Sentaurus Topography 3D must be read by other tools that require the presence of gas on top of their initial structure.

Boundary Types

As described in Input Files on page 2, Sentaurus Topography 3D loads two- or three-dimensional TDR boundary files containing all the geometric and material information of the structure that must be processed.

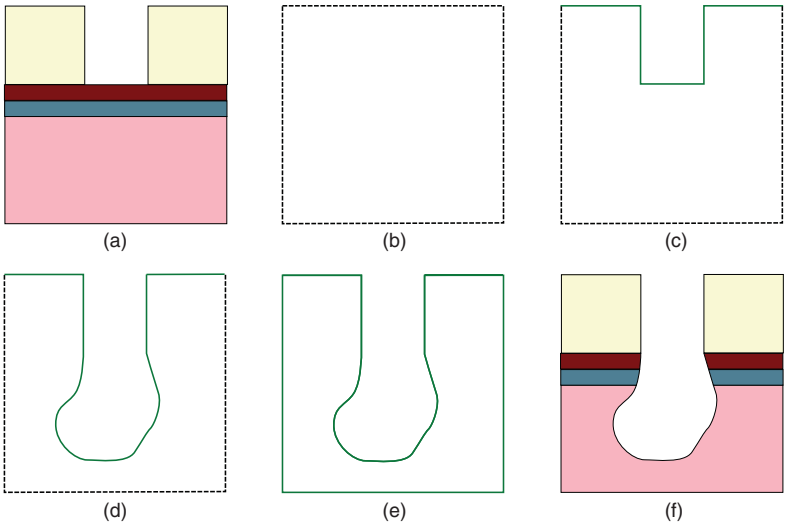


Figure 3 Cross section of a 3D structure: (a) initial structure, (b) initial computational domain, (c) initial exposed surface, (d) final exposed surface, (e) final closed surface, and (f) final structure

Figure 3 (a) shows a cross section of a typical 3D structure that can be used as the *initial structure* of a simulation. The computational domain is defined as the minimum bounding box that contains the structure (see Figure 3 (b)).

The upper part of the surface of the structure is called the *exposed surface* (see Figure 3 (c)). When using the level-set method or the spin-on-glass deposition model, the exposed surface evolves until the end of the simulation when a final profile of it is obtained (see Figure 3 (d)).

At the end of the simulation, the final exposed surface can be combined with the lateral and the bottom planes of the computational domain, resulting in the *closed surface* (see Figure 3 (e)).

Using Boolean operations, a boundary representation of the final structure, hereafter referred to as the *final boundary structure* (see Figure 3 (f)), which includes all material information, can be created from the initial structure (see Figure 3 (a)) and the closed surface (see Figure 3 (e)).

The boundary data structures – exposed surface, closed surface, and final boundary structure – can be saved in a TDR boundary file using the `save` command (see save on page 199).

The final boundary structure contains much more information than the exposed and closed surfaces, but the Boolean operations can be computationally expensive.

The final structure is necessary and must be saved when transferring the simulation results to another TCAD Sentaurus tool (for example, Sentaurus Process).

Simulating Process Steps on 2D and 3D Structures

When using a level set–based model or the spin-on-glass deposition model, Sentaurus Topography 3D can simulate process steps on both two-dimensional (2D) and three-dimensional (3D) structures. Two-dimensional and 3D structures can be simulated using the same physical models, with consistent results. When using the PMC simulation method, only process steps on 3D structures can be simulated.

Simulating 2D cuts of 3D structures typically is orders of magnitude faster than a full 3D simulation. When using level set–based models, compared to the 2D simulator Sentaurus Topography, Sentaurus Topography 3D is slower when processing 2D structures because it uses a more flexible 2D/3D engine.

Flux integration for 2D structures is performed using the radiosity method. No Monte Carlo implementation is provided.

For a full list of limitations, see Limitations When Processing 2D Structures on page 16 and Chapter 9 on page 243.

Sentaurus Topography 3D provides a command language that is mostly independent of the dimension of the structure to simulate.

A structure is defined by the `define_structure` command, and there are two use cases:

- `define_structure file=<c> [name=<c>]`

  Sentaurus Topography 3D reads the file specified by the parameter `file`. The last 2D or 3D TDR boundary contained in that file is used to create a structure with the name specified by the parameter `name` or with the default name `default_structure`.

- ```
  define_structure point_min=<v> point_max=<v> material=<c> \
    [name=<c>]
  ```

  Sentaurus Topography 3D creates a structure – a cube or a rectangle – depending on the length of the vectors used for the parameters `point_min` and `point_max`, with the name specified by the parameter `name` or with the default name `default_structure`.

See define_structure on page 139 for a complete description of this command.
  ```

Any other command either refers to an already defined structure (through its `structure` parameter) or does not operate on any structure. In this way, you can mix 2D and 3D simulations in the same input file, as shown in the following example:

```
# Define a 3D structure called 'structure_3d'. The dimension of the created
# structure is determined by the size of the 'point_min' and 'point_max'
# parameter values.
define_structure name=structure_3d material=Silicon \
   point_min={0 0 0} point_max={1 1 1}

# Define a 2D structure called 'structure_2d'. The dimension of the created
# structure is determined by the size of the 'point_min' and 'point_max'
# parameter values.
define_structure name=structure_2d material=Oxide \
   point_min={0 0} point_max={1 1}

# Define a deposition machine. The 'define_deposit_machine' command does
# not operate on any structure and the machine it defines can be used with
# both 2D and 3D structures.
define_deposit_machine anisotropy=0.8 curvature=0 material=Nitride \
   model=simple rate=1

# Simulate a deposition process for the 2D structure 'structure_2d' and
# save the result. Because the referred structure is 2D, a 2D simulation
# will be run.
deposit spacing=0.1 structure=structure_2d time=1

# Save the result of the deposition step for structure 'structure_2d'.
save structure=structure_2d

# Simulate a deposition process for the 3D structure 'structure_3d' and
# save the result. This command looks exactly like the one above. It only
# refers to a different structure. Because the referred structure is 3D,
# a 3D simulation will be run.
deposit spacing=0.1 structure=structure_3d time=1

# Save the result of the deposition step for structure 'structure_3d'.
save structure=structure_3d

# Simulate another deposition process for the 3D structure 'structure_3d' and
# save the result. Because a discretization that is not uniform across the
# coordinate directions is specified, the length of the value of the 'spacing'
# parameter must match the dimension of the structure this command operates on.
# If the structure had dimension 2, an error will be issued by the 'deposit'
# command.
deposit spacing={0.1 0.5 0.025} structure=structure_3d time=1

# Save the result of the second deposition step for structure 'structure_3d'.
save structure=structure_3d
```

NOTE A 2D structure can be obtained as a cut of a 3D structure using the `extract` command with `type=slice` (see extract on page 166).

Limitations When Processing 2D Structures

The commands listed in Table 1 are not supported or have some limitations when processing 2D structures.

Table 1 Commands not supported or not fully supported when processing 2D structures

Command	Functionality when processing 2D structures
`add_litho_command`	Sentaurus Lithography integration only works in three dimensions.
`define_boundary_conditions`	There is no support for nondefault boundary conditions for indirect flux computation for 2D structures.
`define_litho_machine`	Sentaurus Lithography integration only works in three dimensions.
`define_mask`	Masks are only supported to process 3D structures.
`etch`	Supported except for the `mask` parameter and except when using a reaction model.
`filter_structure`	Supported except for `type=smooth`.
`litho`	Sentaurus Lithography integration only works in three dimensions.
`pattern`	Patterning is only supported to process 3D structures.

Coordinate Systems

This section describes the wafer coordinate system and the simulation coordinate system.

Wafer Coordinate System

Similar to Sentaurus Process, Sentaurus Topography 3D uses a wafer coordinate system to describe the crystal orientation of the wafer. The wafer coordinate system is a right-handed coordinate system in which the x-axis is parallel to the wafer flat, and the y-axis is perpendicular to the wafer flat. The z-axis is perpendicular to the wafer surface (see Figure 4 on page 17).

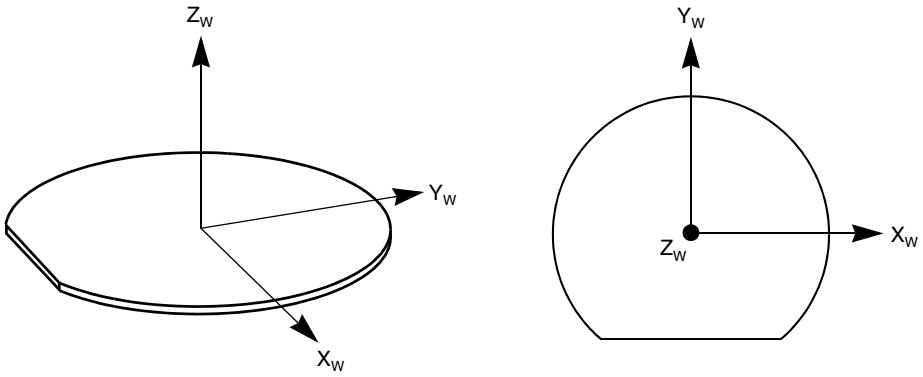


Figure 4 Wafer coordinate system

The tilt and rotation of the structure are defined with respect to the wafer coordinate system (see Structure Tilt on page 18).

Simulation Coordinate System

In contrast to the simulation coordinate system of Sentaurus Process, in the simulation coordinate system of Sentaurus Topography 3D, the x-axis and y-axis lie in the wafer plane, and the z-axis is perpendicular to the wafer surface. The x-axis of the simulation coordinate system is rotated with respect to the y-axis of the wafer coordinate system by the slice angle (see Figure 5).

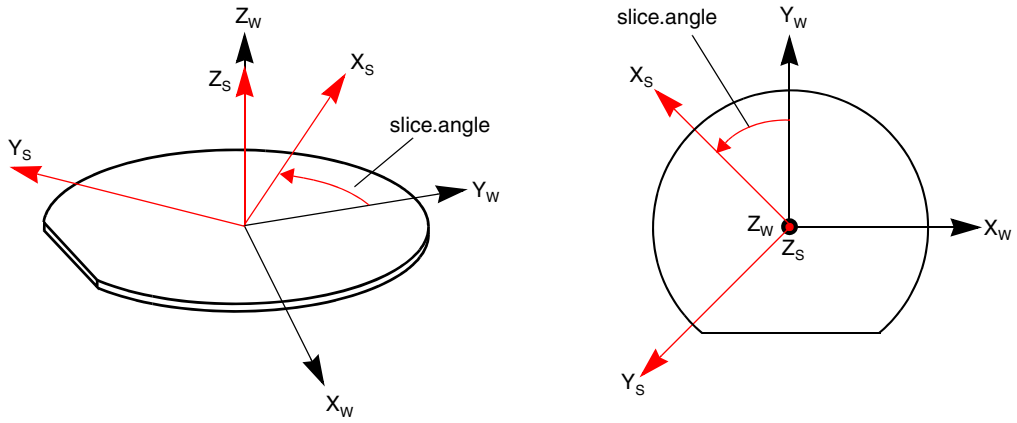


Figure 5 Simulation coordinate system with slice.angle = 45°

As in Sentaurus Process, the default value of the slice angle is –90° (see Figure 6).

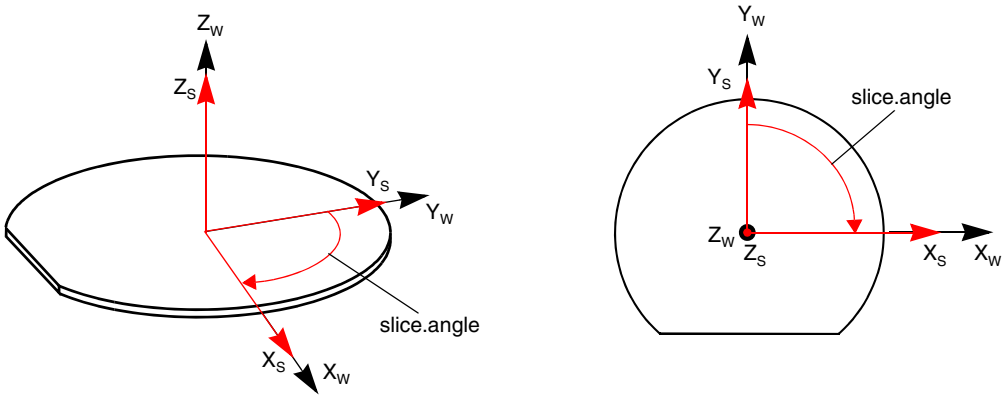


Figure 6 Simulation coordinate system when using default value of slice.angle (–90°)

The simulation coordinate system is used to specify the coordinates of the simulated structure and all input and output coordinates, except for the crystal orientation.

2D Coordinate System

The simulation coordinate system (x/y) used for 2D structures is oriented in such a way that the positive y-axis points upwards in a device. Therefore, the y-coordinate of the simulation coordinate system used for 2D structures corresponds to the z-coordinate of the simulation coordinate system used for 3D structures.

Structure Tilt

When a machine is defined using the `define_deposit_machine` command or the `define_etch_machine` command, you can specify the tilt of the structure by setting the parameters `tilt` and `rotation`, which represent the tilt and rotation (or twist) angles as shown in Figure 7 on page 19.

The tilt is the angle between the beam and the z-axis of the wafer coordinate system. The rotation is defined as the angle between the following two planes:

1. The plane that contains the beam and the z-axis of the wafer coordinate system.
2. The yz plane of the wafer coordinate system.

The tilt angle is positive when measured from the beam axis to the z-axis; while the rotation angle is positive when measured from plane 1 to plane 2 as shown in Figure 7.

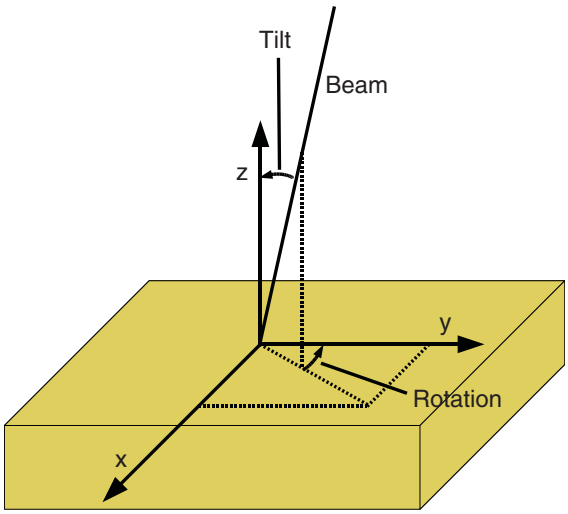


Figure 7 Definition of tilt and rotation angles

Boundary Conditions

Similar to other types of simulation, topography simulations usually handle only a small part of a much larger structure. Typical reasons for this are limited computational resources or a limited area of interest. When limiting the simulation domain to a small part of a larger structure, it is necessary to use boundary conditions to take into account how the parts outside the simulation domain influence the simulation domain.

In general, for some quantities, it is sufficient if the boundary conditions model the immediate vicinity of the simulation domain and, in general, highly accurate results can be obtained. For nonlocal effects, much larger parts outside the simulation domain have a significant influence. This makes the definition of boundary conditions that lead to highly accurate results much more difficult.

For a plasma etching or deposition process, the ion or neutral flux arriving at a point on the surface of the wafer is such a nonlocal effect and can depend on parts of the surface that do not belong to the actual area of interest. Depending on the particular structure, it might still be possible to define boundary conditions that allow you to restrict the simulation domain to the area of interest and to achieve highly accurate results. Unfortunately, this is not always possible.

In cases where limiting the simulation domain to the area of interest and achieving high accuracy is not possible, it is necessary to choose the simulation domain larger than the area of interest.

When increasing the simulation domain, you need to find a trade-off between the improved accuracy in the area of interest and the increase in the required computational resources.

When simulating etching or deposition processes, different boundary conditions are available in Sentaurus Topography 3D: `reflective`, `none`, and periodic. The default for the boundary conditions at the sidewalls of the simulation domain is `reflective`, which implements reflective boundary conditions.

The `define_boundary_conditions` command (see define_boundary_conditions on page 99) specifies the boundary conditions to use when processing a structure. However, currently, users can specify periodic boundary conditions only when processing structures using PMC-based models. Periodic boundary conditions are enabled automatically whenever you use a model that utilizes the RFM function `pad_pressure()` (see Data Available for Rate Calculation on page 225) or when processing tilted structures using PMC-based models, as clarified in the following.

For tilted structures, at sidewalls that are not parallel to the tilt vector, boundary conditions of type `reflective` cannot be used. Therefore, another type of boundary condition must be applied at those sidewalls, independently of the boundary conditions that were specified for the sidewalls using the `define_boundary_conditions` command. When using level set–based models, the boundary condition type `none` is always enforced on such sidewalls; whereas, periodic boundary conditions are applied when using PMC-based models.

Boundary Conditions: reflective

In general, with reflective boundary conditions, it is assumed that the sidewalls of the simulation domain are symmetry planes. When checking the visibility of the plasma source or other parts of the surface in a specific direction, a ray is started in this direction and the first intersection with the plasma source, the surface, or the sidewalls of the simulation domain is calculated. If the plasma source or the surface is hit, the process stops. If a sidewall of the simulation domain is hit, the ray is reflected and the next intersection is calculated.

In theory, this corresponds to an infinitely extended surface and gives the correct result if the sidewalls of the simulation domain are symmetry planes. In practice, it is necessary to limit the number of reflections and, thereby, to reduce the size of the extended surface that is taken into account. As long as this limit is high enough, it will not have a significant influence on the result.

When calculating the indirect flux using the radiosity method, it is necessary to solve an equation system. The elements of the system matrix depend on the material properties of the surface and on the geometry of the surface.

Radiosity Method

The geometric interaction between a pair of surface elements is described by the form factor. The form factor must be calculated for all pairs of elements of the original surface. However, it is also necessary to calculate the form factors between elements of the original surface and elements that are part of the extended surface, which was created by reflecting the original surface at the sidewall of the simulation domain where the boundary condition type is `reflective`. At corners where both adjacent sidewalls have the boundary type `reflective`, the surface must be extended for the corner area. This is performed by reflecting the original surface at each of these sidewalls, and the form factors between these elements and the elements of the original surface are calculated.

The complexity of calculating the form factors between elements of the original surface and elements of the extended surface increases rapidly with the number of reflections that were necessary to create the surface element. Therefore, the extended surface is limited to the direct neighbors at the sidewalls and at the corners of the simulation domain.

Therefore, the accuracy of the indirect flux calculated using the radiosity method strongly depends on the characteristic of the surface within the simulation domain. For example, if the surface inside the simulation domain represents a quarter or a half of a hole (see Figure 8), the interaction of a surface element inside the simulation domain will be limited to other surface elements inside the simulation domain and to elements of the extended surface that were created by one reflection. In this case, the reflective boundary conditions describe the real situation very well, and the result will be highly accurate.

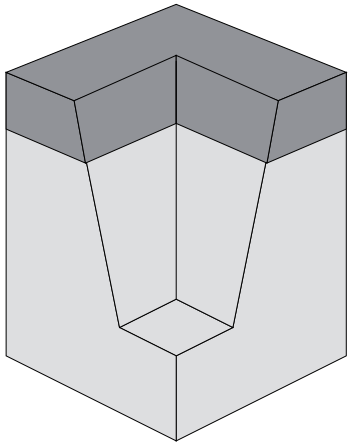


Figure 8 Quarter of a hole

If the surface inside the simulation domain represents the narrow cross section of a trench (see Figure 9 on page 22), surface elements much further away than one reflection can have a significant influence. In this case, the result can contain large errors, and it is recommended to use the 2D mode for this kind of structure.

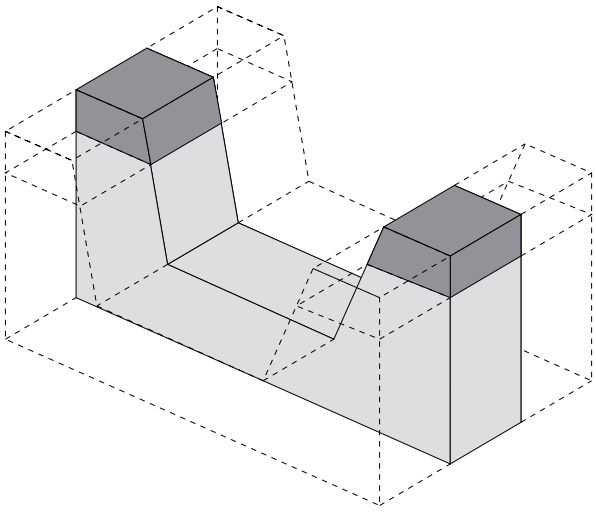


Figure 9 Trench with extended surface at the front and rear sidewalls; the surface is extended at the left and right sidewalls and at the corners but, for this structure, these extensions have no influence on the result and, therefore, are not shown

Boundary Conditions: none

When using the boundary condition type `none`, there is no reflection and no calculation of the next intersection when the ray hits the sidewall of the simulation domain. Currently, boundary conditions of type `none` are not supported when using the PMC method.

When calculating the direct flux, the rays are started from the surface. If a ray hits a sidewall for which the boundary condition type is `none`, the process stops and the plasma source is considered to be visible.

When calculating the indirect flux, the original surface is not extended at sidewalls with the boundary condition type `none`. Therefore, there is no contribution to the indirect flux from elements that would be located in this area.

Boundary Conditions: periodic

When using periodic boundary conditions, the processed structure is supposed to repeat infinitely many times along the direction where periodic boundary conditions are applied.

Periodic boundary conditions can be applied only to a pair of parallel sidewalls and not to a single sidewall. Currently, users can specify boundary conditions of type periodic only when simulating structures with the PMC method. Periodic boundary conditions are always used for

simulating tilted structures with the PMC method as well as for RFM models that utilize the RFM function `pad_pressure()`.

Units

Sentaurus Topography 3D supports a default unit set (second column in Table 2) that is used whenever no unit is specified for a parameter in the command file.

Different units can also be used by setting them between angle brackets (< >) after the numeric value. Table 2 lists the units supported by Sentaurus Topography 3D.

For example, the default unit for a rate is `<um/min>`. In the command file, it is also possible to specify any of the measurement units listed in Table 2 for velocity.

For more details, see Syntax of Commands and Parameters on page 65.

Table 2 Supported units in Sentaurus Topography 3D

Variable	Default		Other possible values	
	Unit	Symbol	Unit	Symbol
Angle	degree	`<deg>`	radian	`<rad>`
Angular velocity	revolution per minute	`<rpm>`	radian per second	`<rad/s>`
Density	gram per cubic centimeter	`<g/cm3>`	kilogram per cubic meter	`<kg/m3>`
Energy	electron volt	`<eV>`	joule	`<J>`
Length	micrometer	`<um>`	meter	`<m>`
			nanometer	`<nm>`
Surface tension	dyne per centimeter	`<dyn/cm>`	Newton per meter	`<N/m>`
Time	minute	`<min>`	second	`<s>`
Velocity	micrometer per minute	`<um/min>`	meter per second	`<m/s>`
			micrometer per second	`<um/s>`
			nanometer per minute	`<nm/min>`
			nanometer per second	`<nm/s>`
Viscosity	poise	`<poise>`	Pascal second	`<Pa*s>`

Parallelization

Sentaurus Topography 3D can use multiple CPU cores or CPUs to accelerate simulations on shared-memory computers.

Basic Parallelization

To activate multithreaded computation, use the command:

```
let parallel=true
```

The number of threads is determined automatically, depending on the hardware resources, the optimum number of threads for the algorithm used, and the number of parallel licenses available. This is the recommended parallelization mode.

Advanced Parallelization Options

Advanced users can request a specific number of threads using the command:

```
let num_threads=<n>
```

where `<n>` is an integer greater than zero. This forces Sentaurus Topography 3D to use the number of threads specified by the user. The engine will try to check out the necessary number of parallel licenses.

If insufficient licenses are available, the simulator will behave according to the user-specified fallback behavior, as set with the command:

```
let parallel_license=<c>
```

Table 3 lists the options of the `let parallel_license` command and the behavior of the simulator if insufficient licenses are available.

Table 3 Options if insufficient parallel licenses are available

Option	Description
`abort`	Aborts the simulation.
`available`	Checks out the maximum number of available licenses, and uses the maximum number of threads available.
`serial`	(Default) Continues the simulation in serial mode.
`wait`	Waits until enough parallel licenses become available.

References

[1] R. Courant, K. Friedrichs, and H. Lewy, “On the Partial Difference Equations of Mathematical Physics,” *IBM Journal*, vol. 11, no. 2, pp. 215–234, 1967.

CHAPTER 3 Model Descriptions

This chapter describes the fundamental theory and assumptions of the physical models used in Sentaurus Topography 3D. In particular, level set–based models, the spin-on-glass deposition model, and the reaction models are addressed.

Level Set–Based Models

This section discusses level set–based models.

Modeling of Fluxes and Related Physical Effects

Figure 10 on page 28 shows an overview of the physical processes relevant to the process models implemented in Sentaurus Topography 3D based on the level-set method.

The source of reactants (in most cases, a plasma) is considered to consist of two kinds of particle: neutrals and ions. Neutrals and ions are assumed to be independent and do not interact with each other. However, they can interact with the surface in various ways, as described in the following.

Moreover, for all plasma processes, the pressure in the reaction chamber is assumed to be very low. Therefore, the mean free path length of ions and neutrals is much larger than the feature size.

Fluxes in Sentaurus Topography 3D are normalized such that a flux integrated on an unshadowed flat surface is equal to one. There are two different kinds of flux:

- Energy-independent fluxes with an isotropic angular distribution.

  These fluxes are suitable for modeling electrically neutral species. Therefore, they will be called neutral fluxes in the following.

  It is assumed that neutrals travel in a straight path to the surface without interacting with other particles. On the surface, neutrals react with a certain probability, and deposit or etch material. If they do not react, they are isotropically reemitted from the surface. The reemitted particles can react elsewhere or can be reemitted again.

- Energy-dependent and energy-independent fluxes with an anisotropic angular distribution.

  These fluxes are suitable for modeling charged particles. Therefore, they will be called ion fluxes in the following.

  Ions are assumed to travel in a straight path to the surface. Depending on the impinging angle on the surface, ions can then react, or be reflected, or sputter away surface material.

  In the case of sputtering or reaction, it is assumed that the ion is consumed. The sputtered particles generate an indirect flux, which can be redeposited elsewhere on the surface. Depending on the material that is being sputtered, the sputtered particles can have an angular distribution with the symmetry axis either normal to the surface or in the direction of the reflected impacting ion.

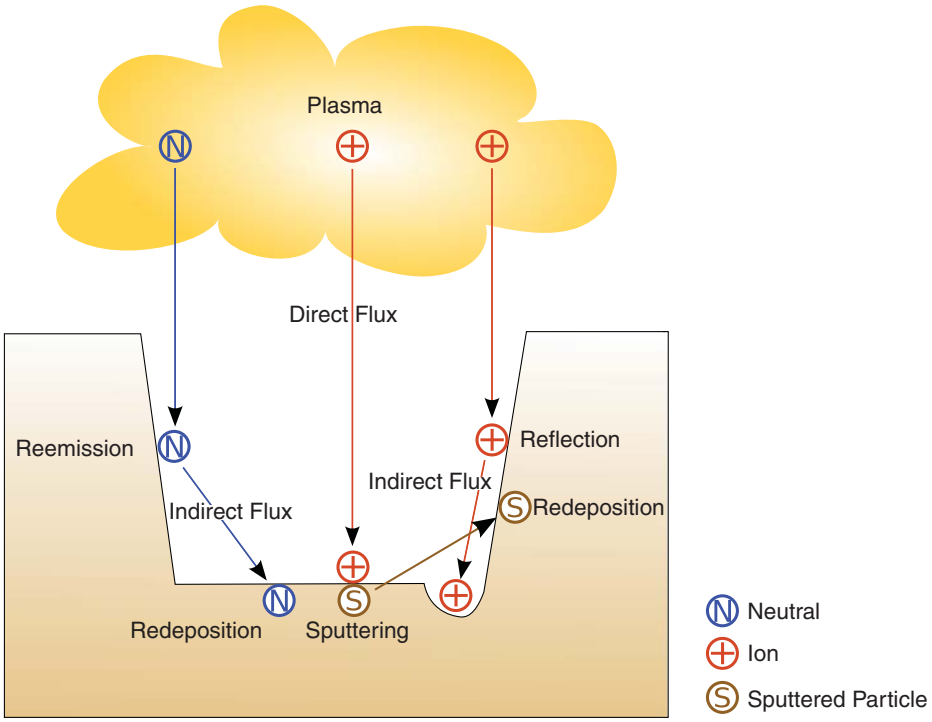


Figure 10 Physical effects available in Sentaurus Topography 3D for modeling plasma processes

Ion fluxes can have arbitrarily shaped and energy-dependent angular distributions. Accordingly, to use a model involving ion fluxes, the angular distributions of each ion flux must be specified (see define_iad on page 117). Ion angular distributions are properties only of the particles and the processing conditions, and they do not depend on the particular structure under process.

On the other hand, the distributions of neutral fluxes are always supposed to be uniform and energy independent. Therefore, no distribution needs to be explicitly defined for the neutral fluxes.

Neutral and ion fluxes are also different in terms of the physical effects they support.

For neutral fluxes, only reemission is taken into account, which means that a part of the neutrals reaching the surface reacts and sticks to the surface; whereas, the remainder is reemitted isotropically. This effect is characterized by the sticking coefficient according to Eq. 2, p. 31.

For each neutral flux, a sticking coefficient must be provided to enable the computation of the reemitted flux. The total number of neutral particles reaching a surface point per unit area and time, taking into account both those directly arriving from the plasma and those reemitted from the surface, is referred to as the *total flux*.

For ion fluxes, it is possible to take into account three physical effects: reflection, sputtering, and deposition of the sputtered material. If reflection is enabled, ions can be reflected specularly at the surface of the structure. The reflection probability depends on both the incident angle of the particles and the surface material. As previously mentioned, reflection probabilities are a property not only of the ion flux being reflected, but also of the target material.

Accordingly, to use a model involving reflection, the reflection probability of each ion flux must be known for each material involved in the simulation. The number of particles per unit area and time reaching a surface point after being reflected by all the other surface points is referred to as the *reflected flux*. For built-in models, only the first reflection of ions by the surface is taken into account in the definition of the reflected flux.

If sputtering is taken into account, the number of particles removed from the surface per unit area and time by ions, referred to as the *sputtered flux*, is computed by Sentaurus Topography 3D. The yield function, which gives the number of sputtered atoms per incoming ion as a function of the ion incident angle, is used to specify the process, and it depends on both the properties of the incoming ions and the target material. For this reason, the yield function of each ion flux included in the model must be provided by users for all the materials involved in the structure under process.

Finally, to model reemission of sputtered material, the sticking coefficient, the angular distribution of the sputtered material, and the type of the sputter emission must be provided. For the sputter type `diffuse`, the symmetry axis of the angular distribution of the sputtered material is given by the surface normal. For the sputter type `reflective`, the symmetry axis is defined by the direction of the specular reflection of the incoming ion.

Therefore, the computation of direct and indirect fluxes might require scalar values (such as the sticking coefficient and the sputter type) as well as some functions, for example, ion angular distribution, yield, and reflection functions. Angular distributions are assumed to be only plasma dependent; whereas, yield and reflection functions are specific to both the particles and the target material.

Figure 11 summarizes the indirect fluxes available in Sentaurus Topography 3D and the information you must provide to compute each of them.

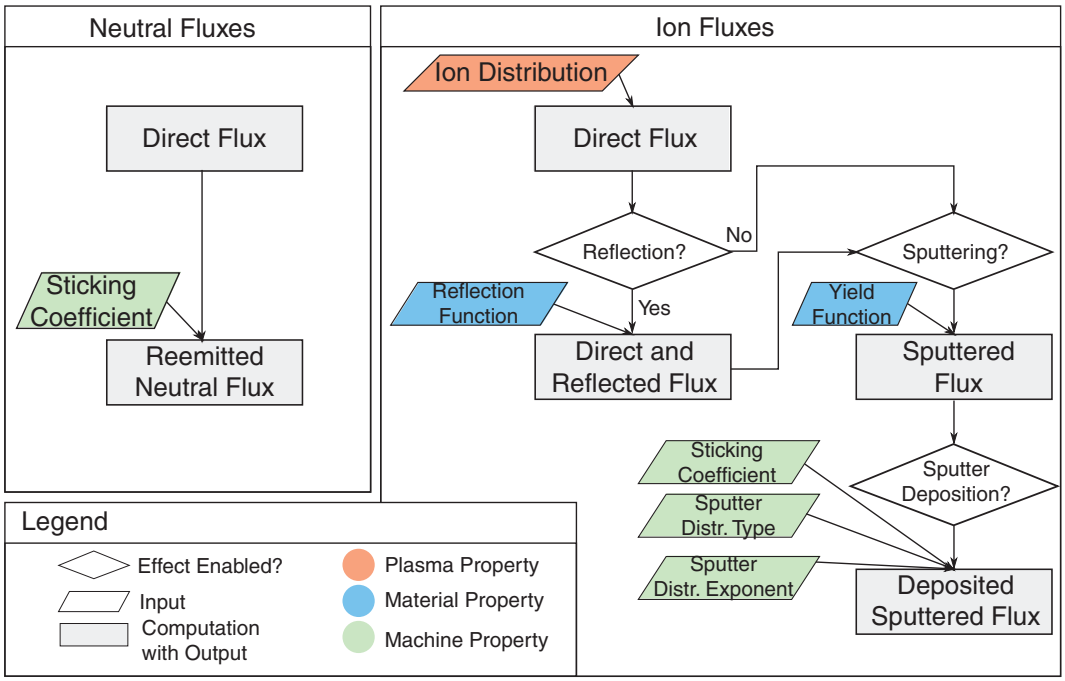


Figure 11 Direct and indirect fluxes and their relationships with physical effects

When a physical effect is enabled for a flux, Sentaurus Topography 3D computes the corresponding indirect flux. However, if an indirect flux does not appear in the rate formula, it will have no effect on the evolution of the surface, even if it is computed by Sentaurus Topography 3D. In other words, the definition and the configuration of a flux specify which direct and indirect fluxes are computed at each time step, but the way they are used is defined only by the rate formula. This is important for user-defined models using the rate formula module (RFM) because, for these kinds of model, users are responsible for adding the appropriate fluxes, activating the required physical effects, and defining the rate formula (see Chapter 6 on page 215).

For built-in models, neutral particles are lumped and modeled as belonging to one flux. The same assumption is made for ion particles in the built-in models. For RFM models, an arbitrary number of ion and neutral fluxes can be used. Each of them can have different properties. Neutrals on page 31 and Ions on page 32 discuss the physical effects in more detail.

Deposition or etching rates for built-in models are obtained by multiplying the normalized flux with the deposition or etching rate measured on an unshadowed flat surface.

NOTE Currently, level set–based models do not support energy-dependent fluxes.

Neutrals

A neutral flux is characterized by an isotropic source distribution that is modeled as $\cos\theta$. Neutral particles emitted from the source and arriving at a surface point j can directly react on the point j or can be reemitted several times from the surface and react at a surface point i (see Figure 12).

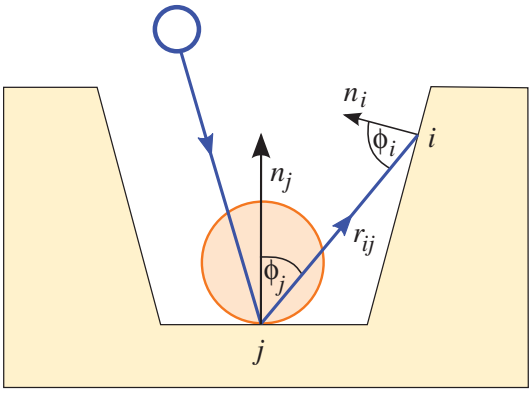


Figure 12 Reemission of neutrals

The sticking coefficient is defined as the reaction probability with the surface:

$$\sigma_j = \frac{\Gamma_{\text{reaction},j}}{\Gamma_{\text{neutral},j}} = 1 - \frac{\Gamma_{\text{re-emitted},j}}{\Gamma_{\text{neutral},j}} \quad (1)$$

where:

- $\Gamma_{\text{neutral},j}$ is the total incoming neutral flux at surface element j.
- $\Gamma_{\text{reaction},j}$ is the fraction of the incoming flux that reacts.
- $\Gamma_{\text{re-emitted},j}$ is the fraction of the incoming flux that is reemitted.

The sticking coefficient σ_j is not a constant on the whole surface. It depends on the material at the surface point j but not on the neutral species, and it varies between 0 and 1.

The total neutral incoming flux on a surface point i can be written as the summation of the direct neutral flux plus all the contributions due to reemissions of the surrounding points j [1]:

$$\Gamma_{\text{neutral},i} = \Gamma_{\text{direct},i} + \sum_{j \neq i} (1 - \sigma_j) g_{ij} \Gamma_{\text{neutral},j} \quad (2)$$

where:

- $\Gamma_{\text{direct},i}$ is the direct flux arriving from the source at the point i.
- g_{ij} is the *form factor* that accounts for how much of the reemitted flux from the point j arrives at the point i.

The form factor depends on the surface geometry and the reemission angular distribution. Assuming that neutrals are reemitted with an isotropic angular distribution [1] (see Figure 12 on page 31):

$$g_{ij} = \int_{A_j} \frac{\cos\phi_i \cos\phi_j}{\pi r_{ij}^2} V_{ij} dA_j \tag{3}$$

where:

- dA_j is the area of the infinitesimal emitter.
- ϕ_i is the angle between the incoming particle direction and the surface normal.
- ϕ_j is the angle between the emitted particle direction and the normal of the surface of the emitter.
- r_{ij} is the distance between the two surface elements i and j.
- V_{ij} is the mutual visibility matrix, defined as:

$$V_{ij} = \begin{cases} 0 & i \text{ and } j \text{ are not mutually visible} \\ 1 & i \text{ and } j \text{ are mutually visible} \end{cases} \tag{4}$$

Eq. 2 is one row of a system of equations that must be solved to find the unknown neutral fluxes $\Gamma_{\text{neutral},i}$ at each surface point.

Ions

The ion flux is characterized by a directional angular source distribution. The source distribution is either modeled as $\cos^m\theta$, where the exponent m controls the flux anisotropy, or specified as an arbitrary ion-angular distribution function (IADF), using the command `define_iad` (see define_iad on page 117) and the parameter `iad` in the respective etching or deposition models.

Ions can react on the surface and can etch or deposit. They also can sputter some material from the substrate or be reflected by vertical walls producing the microtrenching phenomenon at the bottom of a trench.

Analytic Ion-Angular Distribution Function

A good approximation to measure IADFs can be obtained by defining the IADF analytically as:

$$f(\theta) = A\cos^m(\theta) \tag{5}$$

where:

- θ is the angle between the vertical and the incoming ion direction.
- m is a user-defined parameter (exponent) that describes the anisotropy of the distribution.

- A is a constant that is determined by normalizing the integrated ion flux on a flat unshadowed surface.

The flux normalization implies that the normalized flux is dimensionless. Consequently, the total etching or deposition rate can be obtained by multiplying the integrated and normalized flux on a surface element by the etching or deposition rate of a flat unshadowed surface.

User-Defined Ion-Angular Distribution Function

The flux models of Sentaurus Topography 3D allow the definition of user-defined IADFs, which can have been obtained by measurement or from plasma simulations. Arbitrary IADFs can be defined or loaded using the `define_iad` command (see define_iad on page 117). The IADFs are defined in a tabular format, and linear interpolation is used between data points.

User-defined IADFs will be normalized internally when used in flux models.

Sputtering

High-energy ions can sputter some substrate material. The sputter etch rate depends greatly on the impact angle θ_{im} of the ions (the angle between the normal of the surface element and the incoming ion direction; see Figure 13). This dependency is expressed by the yield function:

$$\gamma(\theta_{im}) = s_1 \cos\theta_{im} + s_2 \cos^2\theta_{im} + s_4 \cos^4\theta_{im} \tag{6}$$

where s_1, s_2, and s_4 are the sputtering coefficients.

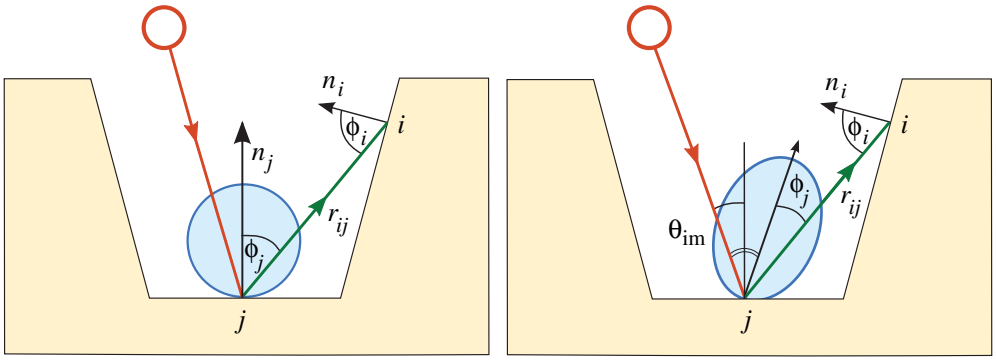


Figure 13 Sputtering with the symmetry axis of the angular distribution (*left*) along the surface normal and (*right*) along the reflected impacting ion direction

In Sentaurus Topography 3D, the sputtered flux is evaluated under the assumption of a narrow angular distribution (a large coefficient m in the angular distribution) for which it is possible to write:

$$\Gamma_{\text{sputter},j} = \gamma(\theta_{im})\Gamma_{\text{ion}} \tag{7}$$

where Γ_{ion} is the total direct ion flux, which is normalized. For the sputtered flux $\Gamma_{sputter}$ to be normalized such that the total sputtered flux from an unshadowed flat surface is unity, it is necessary that $\gamma(0) = 1$. Using this constraint, a relevant condition on the sputtering coefficients is obtained:

$$s_1 + s_2 + s_4 = 1 \tag{8}$$

This means that for the definition of the yield function $\gamma(\theta_{im})$, only two parameters are required, and Sentaurus Topography 3D uses the parameters s_1 and s_2.

The sputtered material can be redeposited as well. It is assumed that the redeposition process occurs with a probability $\sigma_{redeposit}$, and no further reemissions are considered, that is, the remaining $1 - \sigma_{redeposit}$ sputtered material is considered to be volatile. The redeposition flux is:

$$\Gamma_{redeposition,i} = \sigma_{redeposit} \sum_{j \neq i} \nu_{ij} \Gamma_{sputter,j} \tag{9}$$

where ν_{ij} is another form factor that accounts for how much of the sputtered material from the point j arrives at the point i:

$$\nu_{ij} = \int_{A_j} (m+1) \frac{\cos\phi_i \cos^m \phi_j}{\pi r_{ij}^2} V_{ij} dA_j \tag{10}$$

where ϕ_j now has a general meaning as the angle between the emitted particle direction and the symmetry axis of the angular distribution of the sputtered material (see Figure 13 on page 33).

The differences between g_{ij} and ν_{ij} are:

- g_{ij} is evaluated under the assumptions that the reemission of neutrals can be approximated with an isotropic angular distribution. The symmetry axis of the angular distribution is the surface normal, and the exponent of the cosine distribution is one.
- ν_{ij} is evaluated under more general assumptions. Depending on the surface, the angular distribution of the sputtered material can be either:
  - Diffuse – The sputtered particles have no *memory* of the impact direction, and the axis of the distribution is normal to the surface element (see Figure 13 (*left*)).
  - Reflective – The sputtered particles keep some momentum of the impacting ion, and the axis of the distribution has a preferential direction, which is the reflected incoming ion direction (see Figure 13 (*right*)).

In both cases (diffuse and reflective), an exponent can be assigned to the angular distribution.

Reflection

Low-energy ions with a large incident angle have a larger probability of being reflected by walls (see Figure 14). This is an important phenomenon that contributes to microtrenching at the bottom of sidewalls. The probability that an ion is reflected is modeled according to [2]:

$$P_{\text{reflection}}(\theta_{\text{im}}) = \min\left\{1, k\left[\frac{1}{2\pi} + \left(\frac{\pi}{4} - \frac{1}{3}\right)\frac{1}{\left(\frac{\pi}{2} - \theta_{\text{im}}\right)^2} + \frac{5}{\pi^3}\left(\frac{\pi}{2} - \theta_{\text{im}}\right)\right]\right\} \tag{11}$$

where k is a constant that depends on the mass and atomic number of the incoming ion and the wall material, and on the ion energy:

$$k \propto \frac{1}{E_{\text{ion}}^2} \tag{12}$$

The constant k can be set for each material in the structure using the parameter `reflection` in the `add_material` command (see add_material on page 84) and the `define_deposit_machine` command (see define_deposit_machine on page 101).

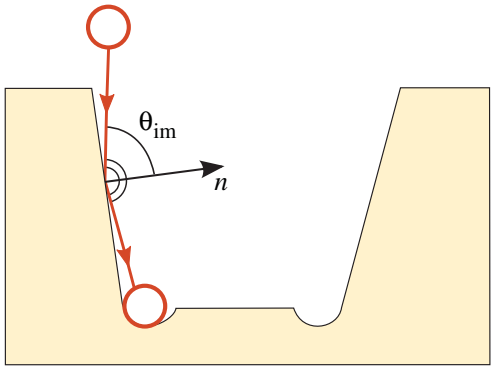


Figure 14 Ion reflection process: ions with a large impact angle can be reflected and react at the bottom of a trench producing microtrenching

Numeric Modeling

The number of incoming (neutral or ion) particles on a surface element is calculated by integrating the various contributions to the total flux. Sentaurus Topography 3D offers two different numeric methods to perform the flux integration: the radiosity method and the Monte Carlo method.

You set the integration method with the parameter `engine` in the `etch` and `deposit` commands. Some models do not support both methods. Furthermore, models that do not perform any flux integration do not need or support these two methods.

NOTE For 2D structures, the only supported flux integration method is the radiosity method.

Table 4 Deposition and etching models: Support for numeric integration methods

Model	Radiosity support	Monte Carlo support
Deposition		
`crystal`	No	No
`electrodeposition`	No	No
`electroplating`	No	No
`hdp`	Yes	Yes
`hdp2`	Yes	Yes
`lpcvd`	Yes	Yes
`pecvd`	Yes	Yes
`pvd`	Yes	Yes
`simple`	No	No
Etching		
`crystal`	No	No
`dry`	Yes	Yes
`etchdepo`	Yes	Yes
`etchdepo2`	Yes	Yes
`hdp`	Yes	Yes
`hdp2`	Yes	Yes
`ion_enhanced`	Yes	Yes
`ionmill`	No	No
`rie`	Yes	Yes
`rie2`	Yes	Yes
`simple`	No	No
`wet`	No	No

Radiosity Method

The radiosity method is a numeric method that solves Eq. 2, p. 31 by evaluating the direct flux Γ_{direct} and then by inverting the linear system matrix to find the unknown fluxes at each surface point.

The linear system matrix scales with the square of the number of surface elements and, therefore, it can become prohibitively large for simulations with small level-set grid spacing. In addition, inverting the system matrix does not lend itself well to parallelization and, therefore, simulations can use more wallclock time than the Monte Carlo method on multicore machines.

The radiosity method is used by default. It is activated explicitly using `engine=radiosity` in the `etch` and `deposit` commands.

NOTE The only supported flux integration method for 2D structures is the radiosity method.

NOTE The boundary condition type `reflective` is not supported for the indirect flux calculation for 2D structures.

NOTE The radiosity method does not provide accurate results for machines with a tilt angle greater than approximately $70°$ for 3D structures.

Monte Carlo Method

In the Monte Carlo method, neutral or ion particles are sent randomly from the top of the simulation domain to the structure. These particles interact with the surface and can be adsorbed or reemitted, and can sputter some material. The particles can be reemitted several times from the surface depending on the sticking coefficient.

The Monte Carlo method uses less memory than the radiosity method because there is no large system matrix to invert. The simulation of individual particles scales well on multiple cores. Therefore, the Monte Carlo method shows significant speedup on multicore machines when run in parallel (see Parallelization on page 24).

You activate the Monte Carlo method using `engine=monte_carlo` in the `etch` and `deposit` commands.

NOTE When using the Monte Carlo method, be careful with the following:

- Due to its stochastic nature, the Monte Carlo method introduces some numeric noise. Therefore, the results of the radiosity method and Monte Carlo method do not always match exactly. The parameter `integration_samples` can be used to increase the number of samples. This increases simulation accuracy at a higher computational cost.
- There is no Monte Carlo implementation for 2D structures. Only the radiosity method is available to perform flux integration with 2D structures.
- The Monte Carlo method can be activated only for models that perform flux integration. See Table 4 on page 36 for a list of supported models.

Orientation-Dependent Models

Sentaurus Topography 3D supports two methods for orientation-dependent etching or deposition modeling when using the level-set method: the built-in `crystal` models and the function `directional_value()` in the rate formula module (RFM) (see Chapter 6 on page 215).

The RFM function `directional_value()` is used to introduce an orientation dependency into the rate formula of an RFM model. This function has three arguments that are the values of an arbitrary quantity for the directions <100>, <110>, and <111>, respectively. Depending on the direction of the normal of a surface element, an appropriately interpolated value of this quantity is returned and can be used in the rate calculation (see Data Available for Rate Calculation on page 225).

For each region of a structure, the crystal orientation can be defined. When creating a new region with the `define_structure`, `deposit`, or `etch` command, the crystal orientation can be defined with the parameters `flat_orientation` and `vertical_orientation` (see define_structure on page 139, deposit on page 147, and etch on page 154).

For structures that were loaded from a TDR file and do not yet contain crystal orientations, or to change the orientation of a region, the `set_orientation` command can be used (see set_orientation on page 205).

Crystal orientations are always specified with respect to the wafer coordinate system (see Wafer Coordinate System on page 16).

Setting Crystallographic Orientations

When using a model with orientation-dependent rates, the crystallographic orientations must be defined with respect to the wafer coordinate system.

The slice angle, which defines how the simulated structure is oriented with respect to the wafer coordinate system, can be set only with the `define_structure` command (see Simulation Coordinate System on page 17). For a newly created structure, the slice angle is defined with the parameter `slice_angle`. If a structure is loaded from a TDR file, the slice angle is read from that file, but the value of the slice angle can be overwritten with the parameter `slice_angle` of the `define_structure` command.

The flat orientation and the vertical orientation of a region can be set with the following commands:

- `define_structure` (only if it is not used to load a structure from a TDR file; see define_structure on page 139)
- `deposit` (see deposit on page 147)
- `etch` (only when using a model for simultaneous etching and deposition; see etch on page 154)
- `set_orientation` (see set_orientation on page 205)

The `define_structure` command is used to set the crystallographic orientation of a newly created structure.

The `set_orientation` command is used to set the crystallographic orientation of any region of a structure.

NOTE The parameters `flat_orientation` and `vertical_orientation` of the `define_structure` command are optional (without a default value). Unless specified, no crystallographic orientation will be set for a newly created region.

NOTE The parameters `flat_orientation` and `vertical_orientation` of the `etch` and `deposit` commands are mandatory if the crystallographic orientation is required by the model, namely, for the `crystal` deposition model and RFM models that use the `directional_value()` function in the rate formula. Otherwise, these two parameters are optional (without a default value).

Continuously Rotating and Tilted Structure Modeling

Sentaurus Topography 3D allows you to model etching, deposition, and simultaneous etching and deposition processes for tilted structures with a continuously increasing value of their rotation angle (see Structure Tilt on page 18).

This feature is available only for rate formula module (RFM) models and reaction models, and is enabled by setting `rotation=continuous` in the `define_deposit_machine` or `define_etch_machine` command (see define_deposit_machine on page 101 and define_etch_machine on page 109).

The simulation of processes involving continuously rotating structures is based on the assumption that the structure rotation period is much smaller than the process time.

To simulate such processes, Sentaurus Topography 3D internally recasts the actual problem into an equivalent one, where the structure of interest is not tilted and does not rotate, and the species of the model (see add_ion_flux on page 81, add_neutral_flux on page 87, and add_source_species on page 98) have angular distributions different from those of the original problem. Each species of the equivalent problem has the same effect on the not tilted and nonrotating structure as the original species has on the tilted and continuously rotating structure, within a margin of error that depends only on the original angular distribution and on the tilt angle. In the following, such a discrepancy is referred to as the *equivalence error* for a species.

The equivalence error of each species is measured by a number in the range [0, 0.5]. You can set the maximum-tolerated equivalence error for all distributions of a model (parameter `maximum_error` of the `define_deposit_machine` or `define_etch_machine` command). The actual equivalence error of each species of the model is written to the log file, when the machine is used with the `deposit` or `etch` command. If the actual equivalence error for any species of the model is greater than the one specified with the `maximum_error` parameter, an error will be issued.

Figure 15 on page 41 shows the values of the equivalence error for species having an analytic angular distribution with different values of the exponent m (see Eq. 5, p. 32) for different tilt angles of the structure.

As can be seen from Figure 15, the equivalence error increases as the tilt angle increases; whereas, for a given tilt angle, the equivalence error decreases as the angular distribution becomes more focused around the source axis. The trend can be observed for species with angular distributions other than that of Eq. 5, p. 32.

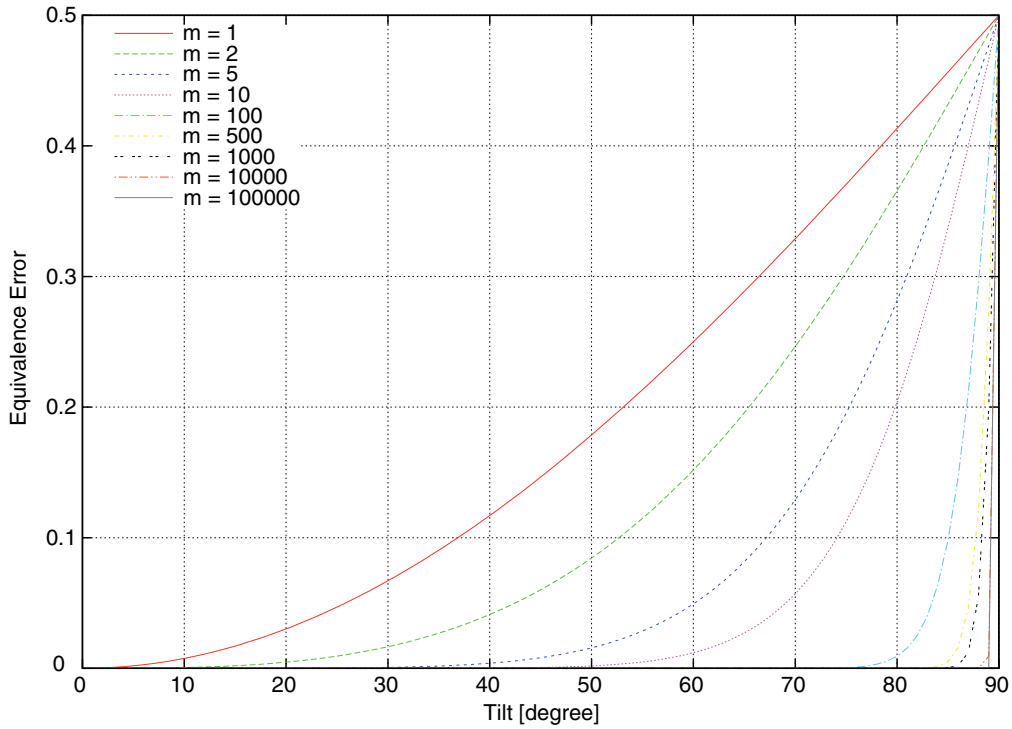


Figure 15 Equivalence error for species having an analytic angular distribution with different values of the exponent for different tilt angles of the structure

Modeling Chemical-Mechanical Polishing Processes

When using the level-set method, Sentaurus Topography 3D allows you to model chemical-mechanical polishing (CMP) processes at feature scale, using the RFM function `pad_pressure()` (see Data Available for Rate Calculation on page 225).

The RFM function `pad_pressure()` returns the pressure produced by a deformable pad in contact with the simulated structure using a contact mechanics–based model [3].

In the implemented model, the pad is characterized by its Young's modulus and Poisson ratio as well as by a Laplacian distribution of the heights of its asperities. The interaction between the pad asperities and the substrate is modeled according to the laws governing Hertz contacts [3].

The Young's modulus of the pad and its Poisson ratio are set with the parameters `pad_young_modulus` and `pad_poisson_ratio` of the `define_etch_machine` command, respectively. Whereas, the standard deviation of the height distribution of the pad asperities is determined from the value of the parameter `pad_roughness` of the

`define_etch_machine` command. The pressure applied to the pad is given by the value of the parameter `applied_pressure` of the `define_etch_machine` command (see define_etch_machine on page 109).

The pressure distribution over the substrate is computed using an iterative method to solve the nonlinear integral equation of the model.

The maximum number of iterations that the solver can perform as well as the solution accuracy below which the solver stops are set with the parameters `pad_pressure_max_iterations` and `pad_pressure_accuracy` of the `etch` command, respectively (see etch on page 154).

Flux and Flux-Emulating Models

This section presents an overview of the flux and flux-emulating level set–based models available in Sentaurus Topography 3D. Flux and flux-emulating models require flux computations or approximate flux quantities using purely geometric quantities, respectively. See Deposition Models on page 44 and Etching Models on page 52 for detailed descriptions of each model.

Flux and Flux-Emulating Deposition Models

Table 5 provides an overview of all flux and flux-emulating deposition models available in Sentaurus Topography 3D.

Table 5 Flux and flux-emulating deposition models

Model	simple	pvd	lpcvd	pecvd	hdp	hdp2

Machine parameters	`anisotropy`
	`curvature`
	`rate`
		`(exponent | iad)`
		`rate`
			`rate`
			`sticking`
				`anisotropy`
				`(exponent | iad)`
				`rate`
				`sticking`
					`anisotropy`
					`(exponent | iad)`
					`rate`
					`redeposition`
					`s1, s2`
					`sputter_rate`
					`sticking`
						`anisotropy`
						`(exponent | iad)`
						`rate`
						`redeposition`
						`reflection`
						`s1, s2`
						`sputter_exponent`
						`sputter_rate`
						`sputter_type`
						`sticking`

Neutral flux			Yes	Yes	Yes	Yes
Ion flux with exponent or IADF		Yes		Yes	Yes	Yes
Reemission and redeposition			Yes	Yes	Yes	Yes

Table 5 Flux and flux-emulating deposition models (Continued)

Model	simple	pvd	lpcvd	pecvd	hdp	hdp2
Sputtering and redeposition of sputtered material					Yes	Yes
Diffusive or reflective sputtering						Yes

Flux and Flux-Emulating Etching and Simultaneous Etching and Deposition Models

Table 6 and Table 7 on page 44 give an overview of etching, and simultaneous etching and deposition, flux and flux-emulating models available in Sentaurus Topography 3D.

Table 6 Flux and flux-emulating etching models

Model	simple	ionmill	rie	rie2	hdp	hdp2	ion_enhanced
Machine parameters			(exponent | iad)	(exponent | iad)	(exponent | iad)	(exponent | iad)	(exponent | iad)
Material parameters	anisotropy curvature rate	anisotropy rate s1, s2	anisotropy rate	anisotropy rate reflection sticking	anisotropy rate s1, s2	anisotropy rate reflection s1, s2 sputter_rate sticking	anisotropy desorption_ rate rate reflection s1, s2 sticking
Isotropic etch rate	Yes	Yes	Yes		Yes	Yes	
Unidirectional ion flux	Yes	Yes					
Ion flux with exponent or IADF			Yes	Yes	Yes	Yes	Yes
Sputtering		Yes			Yes	Yes	Yes
Reemission				Yes		Yes	Yes
Ion reflection				Yes		Yes	Yes

Table 7 Flux and flux-emulating simultaneous etching and deposition models

Model	dry	etchdepo	etchdepo2

Machine parameters	`deposit_material`
	`rate`
	`sticking`
		`deposit_material`
		`(exponent | iad)`
		`rate`
		`sticking`
			`deposit_material`
			`(exponent | iad)`
			`rate`
			`sticking`

Material parameters	`rate`
	`s1, s2`
		`rate`
		`reflection`
		`s1, s2`
		`sputter_type`
			`anisotropy`
			`desorption_rate`
			`rate`
			`reflection`
			`s1, s2`
			`sticking`

Isotropic etch rate			
Unidirectional ion flux			
Ion flux with exponent or IADF		Yes	Yes
Sputtering	Yes	Yes	Yes
Diffusive or reflective sputtering		Yes	Yes
Reemission	Yes		Yes
Redeposition		Yes	
Ion reflection		Yes	Yes

Deposition Models

Simple Deposition

Set `model=simple` in the `define_deposit_machine` command to define a simple deposition machine (see define_deposit_machine on page 101).

The deposition rate is evaluated as the contribution of the following components:

- An isotropic deposition
- A directional deposition
- A curvature-dependent term

The resulting rate at a surface point (x, y, z) can be written as:

$$R(x, y, z) = R_0[(1 - A) + H(x, y, z)A\vec{v} \cdot \vec{n}(x, y, z)](1 - k\kappa(x, y, z)) \quad (13)$$

where:

- A is the anisotropy factor, parameter `anisotropy`.
- $H(x, y, z)$ is a function that accounts for point shadowing in the vertical direction, defined as:

$$H(x, y, z) = \begin{cases} 0 & \text{vertically shadowed region} \\ 1 & \text{vertically unshadowed region} \end{cases} \quad (14)$$

- k is the curvature factor; parameter `curvature`.
- $\kappa(x, y, z)$ is the surface curvature at the point (x, y, z).
- $\vec{n}(x, y, z)$ is the unit vector normal to the surface at the point of coordinate (x, y, z).
- R_0 is the deposition rate on a completely flat surface; parameter `rate`.
- $\vec{v}$ is the unit vector normal to the horizontal plane.

The limiting values of $A = 0$ and $A = 1$ correspond to a pure isotropic and a pure directional deposition model.

The pure isotropic deposition model mimics a chemical vapor deposition (CVD) where reactants, or reactive intermediates, have a very low sticking coefficient, resulting in a uniform concentration of reactants along the surface irrespective of the geometric configuration.

The pure directional deposition model corresponds to a physical vapor deposition (PVD) process where deposition occurs in only one direction. Atoms in the vapor stream that impinge on the surface of the structure are always parallel to the vector $\vec{v}$. No material deposition occurs in the shadowed regions (see Eq. 13 and Eq. 14).

The curvature term has the effect of decreasing the deposition at convex surfaces and increasing the deposition at concave surfaces, smoothing the surface as it evolves.

Physical Vapor Deposition

Set `model=pvd` in the `define_deposit_machine` command to define a physical vapor deposition (PVD) machine (see define_deposit_machine on page 101).

The PVD model applies to processes where the deposition involves pure physical processes rather than chemical reactions. The particle flux is characterized by a $\cos^m\theta$ angular distribution, where m is set by the parameter `exponent` or a user-defined IADF (the parameter `iad`).

The sticking coefficient is equal to one, that is, each incoming particle deposits on the surface. The deposition rate R_0 on an unshadowed flat surface is set with the parameter `rate`.

The total deposition rate is evaluated as:

$$R = R_0\Gamma \tag{15}$$

where Γ is the integrated flux on the considered surface point.

Low-Pressure Chemical Vapor Deposition

Set `model=lpcvd` in the `define_deposit_machine` command to define a low-pressure chemical vapor deposition (LPCVD) machine (see define_deposit_machine on page 101).

In the LPCVD model, the vapor flux of chemical precursors is simulated by neutrals having an isotropic angular flux distribution. The incoming particles can either react on the surface and be deposited, with a probability given by the sticking coefficient (parameter `sticking`), or be reemitted as described in Neutrals on page 31. The reemitted flux is distributed isotropically. The deposition rate R_0 on an unshadowed flat surface is set with the parameter `rate`. The total deposition rate is evaluated as:

$$R = R_0\Gamma_{\text{neutral}} \tag{16}$$

with Γ_{neutral} evaluated from Eq. 2, p. 31.

Plasma-Enhanced Chemical Vapor Deposition

Set `model=pecvd` in the `define_deposit_machine` command to define a plasma-enhanced chemical vapor deposition (PECVD) machine (see define_deposit_machine on page 101).

The incoming fluxes to the surface are characterized by two components: a thermally driven LPCVD precursor and ion-induced deposition precursors. The parameter `anisotropy` sets the ratio between the anisotropic ion rate and the total deposition rate R_0 (parameter `rate`). The ion flux can be defined to have a $\cos^m\theta$ angular distribution, where m is set by the parameter `exponent`, or can be a user-defined IADF (the parameter `iad`).

The total deposition rate is evaluated as:

$$R = (1-A)R_0\Gamma_{\text{neutral}} + AR_0\Gamma_{\text{ion}} \tag{17}$$

High-Density Plasma Deposition

Set `model=hdp` in the `define_deposit_machine` command to define a high-density plasma (HDP) deposition machine (see define_deposit_machine on page 101).

In the HDP deposition model, two simultaneous mechanisms are competing: the deposition and the etching by physical sputtering due to high-energy ions. The neutral flux can either react and be deposited with a probability given by the sticking coefficient σ_j (see Eq. 1, p. 31), the parameter `sticking`, or be reemitted as described in Neutrals on page 31.

Ions are deposited on the surface and, at the same time, can induce a flux of sputtered particles. The sputtering etch rate is highly dependent on the impact angle, and it is described by a yield function (see Eq. 6, p. 33), in which s_1 and s_2 are set with the parameters `s1` and `s2`, respectively.

It is assumed that the sputtered material has an isotropic angular distribution. The amount of redeposition on the surface $\sigma_{\text{redeposition}}$ is set by the parameter `redeposition`.

The total rate on an unshadowed flat surface is given by:

$$R_{\text{flat}} = R_0 - R_{\text{sputter}} \tag{18}$$

where:

- R_0 is the deposition rate on an unshadowed flat surface, without sputtering; parameter `rate`.
- R_{sputter} is the sputtering rate; parameter `sputter_rate`.

NOTE The model assumes a net deposition, that is, $R_0 > R_{\text{sputter}}$ must hold for flat surfaces.

The total deposited material is the sum of the deposition due to neutrals (direct and indirect flux), plus the deposition due to ions, plus the sputtered material that redeposits, minus the amount of locally sputtered material. Finally, for nonzero anisotropy, the deposition rate is:

$$R = (1 - A)R_0\Gamma_{\text{neutral}} + AR_0\Gamma_{\text{ion}} + R_{\text{sputter}}(\Gamma_{\text{redeposition}} - \Gamma_{\text{sputter}}) \tag{19}$$

where A is the anisotropy coefficient; parameter `anisotropy`.

For anisotropy equal to zero, the deposition rate is:

$$R = R_0\Gamma_{\text{neutral}} \tag{20}$$

The fluxes Γ_{neutral}, $\Gamma_{\text{redeposition}}$, and Γ_{sputter} are evaluated as described in Neutrals on page 31 and Ions on page 32.

High-Density Plasma 2 Deposition

Set `model=hdp2` in the `define_deposit_machine` command to define a high-density plasma 2 (HDP2) deposition machine (see define_deposit_machine on page 101).

The HDP2 model extends the HDP model. It takes ion reflection into account and allows you to specify the sputter type and sputter exponent.

For nonzero anisotropy, the deposition rate is:

$$R = (1-A)R_0\Gamma_{\text{neutral}} + AR_0\tilde{\Gamma}_{\text{ion}} + R_{\text{sputter}}(\tilde{\Gamma}_{\text{redeposition}} - \Gamma_{\text{sputter}}) \tag{21}$$

For anisotropy equal to zero, the deposition rate is:

$$R = R_0\Gamma_{\text{neutral}} \tag{22}$$

Eq. 21 has the same structure as Eq. 19, but the factors $\tilde{\Gamma}_{\text{ion}}$ and $\tilde{\Gamma}_{\text{redeposition}}$ for the ion flux and the redeposition, respectively, are different.

$\tilde{\Gamma}_{\text{ion}}$ takes ion reflection into account as described in Reflection on page 35. When the parameter `reflection` is set to 0, $\tilde{\Gamma}_{\text{ion}}$ is equal to Γ_{ion} in Eq. 19.

$\tilde{\Gamma}_{\text{redeposition}}$ can take diffuse or reflective reemission of the sputtered material into account. By setting `sputter_type=diffuse` and `sputter_exponent=1`, $\Gamma_{\text{redeposition}}$ is equal to $\Gamma_{\text{redeposition}}$ in Eq. 19.

As in the HDP model, the HDP2 model considers the process with two competing simultaneous mechanisms: the deposition and the etching by physical sputtering due to high-energy ions. The neutral flux can either react and be deposited with a probability given by the parameter `sticking` or be reemitted. Ions are deposited on the surface and, at the same time, can induce a flux of sputtered particles. The angular dependency of the sputter rate is given by the yield function as defined in Eq. 6, p. 33 by the parameters `s1` and `s2`.

The main difference with respect to the HDP model is that the angular distribution of the sputtered material, diffuse or reflective, can be set by users (see Ions on page 32). The ion reflection index (see Reflection on page 35) is set with the parameter `reflection`.

The rate evaluation for the HDP2 model is the same as that described in Eq. 19.

Electrodeposition

Set `model=electrodeposition` in the `define_deposit_machine` command to define an electrodeposition machine (see define_deposit_machine on page 101).

The model takes into account the following species: plating species, inhibitors, and accelerators. They are assumed to diffuse from the bulk of the plating bath to the surface of the structure under processing, with their concentrations remaining constant at a specified distance above the top of the structure.

Inhibitors and accelerators competitively adsorb on the surface of the structure. Their adsorption kinetics are determined by their local concentration, their adsorption rates, and the local surface curvature. The additives surface interaction model also includes displacement of inhibitors by accelerators and surface saturation effects. The local surface coverage of inhibitors and accelerators, together with the local concentration of the species being plated, determine the deposition rate of the plating species.

You can set up an oscillating overpotential between two values with a given duty cycle to simulate pulse-plating conditions.

The deposition rate R of the plating species at each surface element is determined by the local surface coverage of inhibitors and accelerators, by the local concentration of the species being plated, and by the overpotential, as specified by the following equations [4][5][6]:

$$R = \frac{\Omega}{zF}\frac{C_{\text{depo}}}{C_{\text{depo}}^{\text{ref}}}J \tag{23}$$

where:

$$J = \omega[DC\ J_{\text{on}} + (1 - DC)J_{\text{off}}] \tag{24}$$

$$J_{\text{on}} = J_0(1 - \vartheta_{\text{accel}} - \vartheta_{\text{inhib}})g(\alpha, \eta_{\text{on}}) + J_0^{\text{accel}}\vartheta_{\text{accel}}g(\alpha_{\text{accel}}, \eta_{\text{on}}) + J_0^{\text{inhib}}\vartheta_{\text{inhib}}g(\alpha_{\text{inhib}}, \eta_{\text{on}}) \tag{25}$$

$$J_{\text{off}} = J_0(1 - \vartheta_{\text{accel}} - \vartheta_{\text{inhib}})g(\alpha, \eta_{\text{off}}) + J_0^{\text{accel}}\vartheta_{\text{accel}}g(\alpha_{\text{accel}}, \eta_{\text{off}}) + J_0^{\text{inhib}}\vartheta_{\text{inhib}}g(\alpha_{\text{inhib}}, \eta_{\text{off}}) \tag{26}$$

$$g(a, \eta) = \exp\left(-a\frac{z_\alpha F}{RT}\eta\right) - \exp\left((1 - a)\frac{z_\alpha F}{RT}\eta\right) \tag{27}$$

and:

- Ω is the molar volume of the plating species (parameter `depo_molar_volume`).
- DC is the duty cycle of the electrode overpotential (parameter `duty_cycle`). When its value is 1, pulse plating is disabled.
- z is the number of electrons transferred during the deposition of an ion of the plating species (parameter `z`).
- $F = 96485.33289\,\text{C/mol}$ is the Faraday constant.
- ω is equal to 1 if the current surface element does not belong to a void and is equal to 0 otherwise.

- C_{depo} is the local concentration of the plating species. Its value is computed by the simulator assuming that the concentration of the plating species remains constant to the value specified by the `depo_bulk_concentration` parameter at the distance from the top of the structure given by the `bulk_distance` parameter. The diffusivity of the plating species can be specified with the `depo_diffusivity` parameter. When `depo_diffusivity` is not specified, the concentration of the plating species is assumed to be equal to its bulk value all over the surface. C_{depo} depends on both the electrode overpotential and its duty cycle.
- C_{depo}^{ref} is the reference concentration of the plating species at which the exchange current densities are measured (parameter `depo_reference_concentration`).
- J_0 is the exchange current density for a surface where no additives were adsorbed, when the concentration of the plating species is C_{depo}^{ref} (parameter `exchange_current_density`).
- J_0^{accel} is the exchange current density for a surface fully covered by accelerators, when the concentration of the plating species is C_{depo}^{ref} (parameter `exchange_current_density_acc`).
- J_0^{inhib} is the exchange current density for a surface fully covered by inhibitors, when the concentration of the plating species is C_{depo}^{ref} (parameter `exchange_current_density_inh`).
- ϑ_{accel} and ϑ_{inhib} are the local accelerator and inhibitor surface coverages, respectively. Their values are computed by the simulator and depend on the following quantities (among others):
  - Accelerator adsorption rate (parameter `acc_adsorption_rate`).
  - Inhibitor adsorption rate (parameter `inh_adsorption_rate`).
  - Accelerator saturation surface concentration (parameter `acc_saturation_surface_concentration`).
  - Inhibitor saturation surface concentration (parameter `inh_saturation_surface_concentration`).
  - Inhibitor displacement rate by accelerators (parameter `inh_displacement_rate`).
  - Local concentration of accelerators. This value is computed assuming that the accelerator concentration remains constant to the value specified by the `acc_bulk_concentration` parameter at the distance from the top of the structure given by the `bulk_distance` parameter. The diffusivity of accelerators can be specified with the `acc_diffusivity` parameter. When `acc_diffusivity` is not specified, the concentration of accelerators is assumed to be equal to its bulk value all over the surface.
  - Local concentration of inhibitors. This value is computed assuming that the inhibitor concentration remains constant to the value specified by the `inh_bulk_concentration` parameter at the distance from the top of the structure

given by the `bulk_distance` parameter. The diffusivity of inhibitors can be specified with the `inh_diffusivity` parameter.

- Local surface curvature.

- α, α_{accel}, and α_{inhib} are the transfer coefficients (parameters `alpha`, `alpha_acc`, and `alpha_inh`, respectively).
- z_α is the number of electrons transferred during the rate-determining elementary reaction (parameter `z_alpha`).
- $R = 8.3144598\,\text{J/mol/K}$ is the gas constant.
- T is the absolute temperature of the electroplating cell (parameter `temperature`).
- η_{on} is the electrode overpotential during the *on* time of the pulse period when pulse plating is either activated or deactivated (parameter `overpotential`).
- η_{off} is the electrode overpotential during the *off* time of the pulse period when pulse plating is activated (parameter `overpotential_off`).

Electroplating Deposition

Set `model=electroplating` in the `define_deposit_machine` command to define an electroplating machine (see define_deposit_machine on page 101).

This model applies to the electrodeposition of copper. The deposition rate is proportional to the ratio of accelerators to inhibitors on the surface. The accelerators are conserved as the surface evolves. For trenches, accelerators in the trench accumulate while the trench closes, leading to an accelerated surface growth speed in the trench, an effect that is commonly referred to as *superfilling*.

It is assumed that, at the beginning of a simulation, there is a starting coverage of accelerators distributed on the exposed surface. The initial distribution of accelerators either can be uniform, which is the default, or can be set to be linearly dependent on the depth of the surface. The parameter `delta` describes the derivative of the accelerator coverage with respect to the vertical direction. This parameter is used to emulate an unevenly distributed initial accelerator coverage, where the coverage in the depths of a trench is lower than on the top surface.

In each time step, the surface evolves; whereas, the deposition rate R at each surface element is proportional to the ratio of accelerator coverage C_{accel} and inhibitor coverage C_{inhib} at the surface element:

$$R \propto R_0 \frac{C_{\text{accel}}}{C_{\text{inhib}}} \tag{28}$$

where:

- R_0 is the deposition rate (parameter `rate`).

- C_{accel} is the accelerator coverage on the surface. The total number of accelerators is conserved over time.
- C_{inhib} is the inhibitor coverage on the surface. It is assumed that inhibitors are constantly replenished from the electrolyte solution and, therefore, inhibitors are distributed uniformly in space, and the coverage remains constant in time.

Crystallographic Orientation–Dependent Deposition

Set `model=crystal` in the `define_deposit_machine` command to define a crystallographic orientation–dependent deposition machine (see define_deposit_machine on page 101).

The deposition rate is determined by the orientation of the exposed surface with respect to the lattice of its bulk material, which is assumed to be cubic. More precisely, the deposition rate at any point of the exposed surface is computed as the spherical barycentric interpolation [7], along its normal direction, of the deposition rates along the <100>, <110> and <111> crystallographic directions (parameters `rate_100`, `rate_110`, and `rate_111` of the `define_deposit_machine` command, respectively).

The crystallographic orientation of the deposited material must be defined with the parameters `flat_orientation` and `vertical_orientation` of the `deposit` command. The model does not require the other regions of the structure to have crystallographic orientations defined. See Orientation-Dependent Models on page 38, Setting Crystallographic Orientations on page 39, and set_orientation on page 205.

Moreover, unlike the other deposition models, this model makes it possible to simulate selective deposition processes, as the list of materials on which deposition will occur can be specified using the `selective_materials` parameter of the `define_deposit_machine` command.

Etching Models

Simple Etching

Set `model=simple` in the `define_etch_machine` command to define a simple etching machine (see define_etch_machine on page 109).

The etch rate is evaluated as the contribution of the following different components:

- An isotropic etch
- A directional etch
- A curvature-dependent term

The resulting rate at the surface point (x, y, z) for the material m can be written as:

$$R_m(x, y, z) = R_{0m}[(1 - A_m) + H(x, y, z)A_m \hat{\nu} \cdot \hat{n}(x, y, z)](1 + k_m \kappa(x, y, z)) \quad (29)$$

where:

- A_m is the anisotropy factor of material m, parameter `anisotropy`.
- $H(x, y, z)$ is a function that accounts for shadowing, defined in Eq. 14.
- k_m is the curvature factor for material m, parameter `curvature`.
- $\kappa(x, y, z)$ is the surface curvature in the point (x, y, z).
- $\hat{n}(x, y, z)$ is the unit vector normal to the surface at the point of coordinate (x, y, z).
- R_{0m} is the etching rate on a completely flat surface for material m, parameter `rate`.
- $\hat{\nu}$ is the unit vector normal to the horizontal plane.

Plasma-assisted processes from reactive or nonreactive ions impinging upon the surface of the structure usually result in a directional etching. The directional component is evaluated only for unshadowed regions (see Eq. 14 and Eq. 29).

The curvature term has the effect of increasing etching at convex surfaces and decreasing etching at concave surfaces, smoothing the surface as it evolves.

Ion-Milling

Set `model=ionmill` in the `define_etch_machine` command to define an ion-mill etching machine (see define_etch_machine on page 109).

In this model, the etch is due to high-energy (unidirectional) ions impacting the surface. The sputtering etch rate is evaluated using the yield function defined in Eq. 7, p. 33:

$$R_m(x, y, z) = R_{0m}[(1 - A_m) + H(x, y, z)A_m \gamma_m(\theta_{\text{im}})] \quad (30)$$

where:

- A_m is the anisotropy factor of material m, parameter `anisotropy`.
- $\gamma_m(\theta_{\text{im}})$ is the value of the yield function for the material m and the ion impact angle θ_{im}.
- $H(x, y, z)$ is the function that accounts for the surface point shadowing as defined in Eq. 14, p. 45.
- R_{0m} is the etching rate on a completely flat surface for material m; parameter `rate` in the command `add_material`.

Reactive Ion Etching

Set `model=rie` in the `define_etch_machine` command to define a reactive ion etch (RIE) machine (see define_etch_machine on page 109).

The etching process has two contributions: an isotropic etch and an anisotropic etch. The isotropic etch emulates the wet (chemical) etch process, while the anisotropic etch emulates the plasma etch process.

The anisotropic etch rate is proportional to the incoming ion flux. The ion flux Γ_{ion} either has an angular distribution $\cos^m\theta$, where m is set by the parameter `exponent`, or is a user-defined IADF (the parameter `iad`).

The etch rate R_{0m} on an unshadowed flat surface for the material m is set with the parameter `rate`.

The total etch rate is the sum of these two etch rates:

$$R_m = (1 - A_m)R_{0m} + A_m R_{0m}\Gamma_{\text{ion}} \tag{31}$$

where A is the anisotropy coefficient; parameter `anisotropy`.

Reactive Ion Etching 2

Set `model=rie2` in the `define_etch_machine` command to define a reactive ion etch 2 (RIE2) machine (see define_etch_machine on page 109).

The RIE2 model extends the RIE model by improving both the neutral and the ion modeling. The RIE2 model considers the etching process as the result of the following contributions:

- Chemical reactions induced by radicals
- Desorptions from the surface induced by ions

The evaluation of the neutral flux Γ_{neutral} is performed considering the structure shadowing and the multiple reemissions as described in Neutrals on page 31 (see Eq. 2). The sticking coefficient is material dependent and must be set during the material definition with the parameter `sticking`.

Ion reflection can be included by setting the parameter `reflection` as described in Reflection on page 35. The ion etch rate is proportional to the incoming ion flux. The ion flux has an angular distribution $\cos^m\theta$, where m is set by the parameter `exponent`, or is a user-defined IADF (the parameter `iad`).

The etch rate R_{0m} on an unshadowed flat surface for the material m is set with the parameter `rate`.

The etch contribution due to neutrals and ions is:

$$R_m = (1 - A_m)R_{0m}\Gamma_{\text{neutral}} + A_m R_{0m}\Gamma_{\text{ion}} \tag{32}$$

where A_m is the anisotropy coefficient for material m, parameter `anisotropy`. The flux Γ_{neutral} is evaluated as described in Neutrals on page 31.

High-Density Plasma Etching

Set `model=hdp` in the `define_etch_machine` command to define a high-density plasma (HDP) etching machine (see define_etch_machine on page 109).

As in the RIE process, the etch rate has two contributions: an isotropic etch and an anisotropic etch. The total etch rate is the sum of these two contributions; the parameter `anisotropy` defines the ratio between the anisotropic etch rate and the total rate. The isotropic etch emulates the wet (chemical) etch process. The anisotropic etch emulates the plasma etch process that, contrary to the RIE model, is due mainly to sputtering. The sputtering yield function is defined as in Eq. 6, p. 33. The anisotropic etch models the plasma etch process where the anisotropic etch rate is proportional to the incoming ion flux. The isotropic part consists of two terms: an isotropic etch term and a term that is proportional to the incoming ion flux. The latter is only active if the surface element is shadowed by another surface element in the vertical direction; whereas, the former is only active if, in the vertical direction, there is no shadowing (see Eq. 14, p. 45).

The total etch rate is:

$$R_m = (1 - A_m)R_{0m}[H(x, y, z) + (1 - H(x, y, z))\Gamma_{\text{ion}}] + A_m R_{0m}\gamma_m(\theta_{\text{im}})\Gamma_{\text{ion}}H(x, y, z) \tag{33}$$

where:

- A_m is the anisotropy factor of material m; parameter `anisotropy`.
- R_{0m} is the etching rate on a completely flat surface for the material m; parameter `rate` in the command `add_material`.
- $H(x, y, z)$ is the function that accounts for the surface point shadowing as defined in Eq. 14.

The ion flux has an angular distribution $\cos^m\theta$, where m is set by the parameter `exponent`, or is a user-defined IADF (the parameter `iad`).

High-Density Plasma 2 Etching

Set `model=hdp2` in the `define_etch_machine` command to define a high-density plasma 2 (HDP2) etching machine (see define_etch_machine on page 109).

The HDP2 model extends the HDP model by implementing the features already discussed in Reactive Ion Etching 2 on page 54 for the RIE2 model: neutral flux calculation including reemissions and ion reflections.

The HDP2 model considers etching as the result of the following contributions:

- Chemical reactions induced by radicals
- Desorptions from the surface induced by ions
- Sputtering

The total rate is given by:

$$R_m = (1 - A_m)R_{0m}\Gamma_{\text{neutral}} + A_m R_{0m}\Gamma_{\text{ion}} + A_m R_{\text{sputter},\,m}\gamma_m(\theta_{\text{im}})\Gamma_{\text{ion}} \tag{34}$$

where:

- A_m is the anisotropy coefficient for the material m; parameter `anisotropy`.
- $R_{\text{sputter},\,m}$ is the sputtering rate for the material m; parameter `sputter_rate`.
- R_0 is the etch rate on a flat unshadowed surface when there is no sputtering; parameter `rate`.
- Γ_{neutral} is the neutral flux that takes shadowing and multiple reemissions into account as described in Neutrals on page 31 (see Eq. 2, p. 31). The sticking coefficient σ_j is material dependent and must be set during the material definition with the parameter `sticking`.

Ion-Enhanced Etching

Set `model=ion_enhanced` in the `define_etch_machine` command to define an ion-enhanced etching machine. (see define_etch_machine on page 109)

The ion-enhanced etching model considers etching as the result of the combined effects of neutrals and ions. Neutrals are deposited on the surface (neutral flux Γ_{neutral}). Then, either these neutrals are removed by ions or they dissociate from the surface through thermal desorption. Where the layer of neutrals is removed, chemical etching occurs.

The etching due to ions is modeled as a sputter etching process, where the sputter yield function is defined as in Eq. 6, p. 33.

In this model, the total etch rate is not considered as the linear summation of neutrals and ions etch rates, but:

$$R_m = \frac{1}{\frac{1}{(1-A_m)\,R_{0m}\,\Gamma_{\text{neutral}}} + \frac{1}{A_m^2\,R_{0m}\gamma_m(\theta_{\text{im}})\,\Gamma_{\text{ion}} + R_{\text{des},m}}} \tag{35}$$

where:

- $R_{\text{des},m}$ is the thermal desorption rate for material `m`; parameter `desorption_rate` in the command `add_material`.
- A_m is the anisotropy of material `m`; parameter `anisotropy` in the command `add_material`.
- R_{0m} is the etch rate on a flat surface of material `m`; parameter `rate` in the command `add_material`.

This model has the following properties:

- If one of the fluxes is zero, the overall etch rate is zero since both neutrals and ions are required to etch the material.
- The etch rate becomes saturated when one component becomes too large relative to the other.

The neutral flux is evaluated as in the RIE2 and HDP2 models by considering the shadowing and the reemission probability on the surface. Ion reflections to the sidewalls can also be considered by setting the parameter `reflection`.

Wet Etching

Set `model=wet` in the `define_etch_machine` command to define a wet etching machine (see define_etch_machine on page 109).

This model assumes that the etchant concentration is constant at a specified distance above the top of the structure, and that the transport of the etchant to the surface of the structure occurs by diffusion. At the surface of the structure, the model assumes that the etchant is consumed by a first-order reaction.

Dry Etching

Set `model=dry` in the `define_etch_machine` command to define a dry etching machine (see define_etch_machine on page 109).

This model considers the combination of sputter etching and deposition processes. The etch process is induced by unidirectional high-energy ions that sputter the material of the structure

(see Ion-Milling on page 53). At the same time, an LPCVD process is considered (see Low-Pressure Chemical Vapor Deposition on page 46).

The total deposition etch rate is evaluated as:

$$R_m = R_{\text{deposition}}\Gamma_{\text{neutral}} - H(x, y, z)R_{\text{ion-mill},\, m}\gamma_m(\theta_{\text{im}}) \tag{36}$$

where:

- $R_{\text{deposition}}$ is the deposition rate on a completely flat surface; parameter `rate` in the command `define_etch_machine`.
- $R_{\text{ion-mill},\, m}$ is the etching rate on a completely flat surface for material m; parameter `rate` in the command `add_material`.
- $H(x, y, z)$ is the function that accounts for the surface point shadowing as defined in Eq. 14, p. 45.

Crystallographic Orientation–Dependent Etching

Set `model=crystal` in the `define_etch_machine` command to define a crystallographic orientation–dependent etching machine (see define_etch_machine on page 109).

The etching rate is determined by the orientation of the exposed surface with respect to the lattice of its bulk material, which is assumed to be cubic. More precisely, the etching rate at any point of the exposed surface is computed as the spherical barycentric interpolation [7], along its normal direction, of the deposition rates along the <100>, <110> and <111> crystallographic directions (parameters `rate_100`, `rate_110`, and `rate_111` of the `define_etch_machine` command, respectively).

Compared to other built-in etching models, only one material (specified with the parameter `etchable_material` of the `define_etch_machine` command) can be etched using this model.

The crystallographic orientation of the material to be etched is expected to be set. If there are several regions with the material to be etched, their crystallographic orientations must be all set and they must be identical. See Orientation-Dependent Models on page 38, Setting Crystallographic Orientations on page 39, and set_orientation on page 205.

Simultaneous Etching and Deposition

Set `model=etchdepo` in the `define_etch_machine` command to define a simultaneous etching and deposition machine (see define_etch_machine on page 109).

This model considers the etching process due to sputtering induced by high-energy ions and the redeposition of this sputtered material. At the bottom of the trench, the polymer is usually

removed by ion bombardment; while on the sidewalls, it accumulates forming a thin layer. This layer can inhibit the etching process.

The ion flux has an angular distribution $\cos^m\theta$, where m is set by the parameter `exponent`, or is a user-defined IADF (the parameter `iad`).

The deposition process is simulated considering that the incoming material flux $\Gamma_{\text{sputter deposition},i}$ at point i is the sum of the integrated sputtered flux $\Gamma_{\text{sputter},j}$, which is sputtered from point j and arriving at point i, plus the reemitted flux from all the visible points j:

$$\Gamma_{\text{sputter deposition},i} = \sum_{j \neq i} g_{ij}\Gamma_{\text{sputter},j} + (1-\sigma)\sum_{j \neq i} g_{ij}\Gamma_{\text{sputter deposition},j} \tag{37}$$

Here, $\Gamma_{\text{sputter},j}$ is induced by the ion flux Γ_{ion} (see Eq. 7, p. 33).

Ion reflection at the sidewalls also can be considered by setting the parameter `reflection`. The angular distribution of sputtered material can be set with the parameters `sputter_type` and `exponent`.

Finally, the total rate is evaluated as:

$$R_m = R_{\text{deposition}}\Gamma_{\text{sputter deposition}} - R_{\text{etch},\,m}\Gamma_{\text{sputter}} \tag{38}$$

where:

- $R_{\text{deposition}}$ is the deposition rate on a completely flat surface; parameter `rate` in the command `define_etch_machine`.
- $R_{\text{etch},\,m}$ is the etching rate on a completely flat surface for material m; parameter `rate` in the command `add_material`.
- $\Gamma_{\text{sputter deposition}}$ is the indirect flux of sputtered material, as described in Eq. 37.

A positive total rate R_m indicates a net deposition; whereas, a negative total rate indicates a net etching.

NOTE In contrast to other models, in the `etchdepo` model, the sputter flux emitted from a point is set to zero if the sputter rate for the corresponding material is exactly zero. The sputter rate for a material can be zero either because it has been specified explicitly with the `add_material` command or because the properties of the material have not been specified with the `add_material` command.

Simultaneous Etching and Deposition 2

Set `model=etchdepo2` in the `define_etch_machine` command to define a simultaneous etching and deposition 2 machine (see define_etch_machine on page 109).

This model considers the combined effect of etching in the ion-enhanced regime and the deposition due to polymerization. As in the ion-enhanced model, the ion flux removes the layer of atoms formed by the chemical reactions and damages the surface that favors chemical etching. The polymer deposits on the surface as in an LPCVD process and is removed with the ion-milling mechanism.

The etching due to ions is modeled as a sputter etching process, where the sputter yield function is defined as in Eq. 6, p. 33.

The polymer also dissociates thermally, with a rate specified by the parameter `desorption_rate`. At the bottom of the trench, the polymer is usually removed by ion bombardment; while on the sidewalls, it accumulates forming a thin layer. This layer can inhibit the etching process. The etch rates and fluxes are evaluated as in the ion-enhanced model; while the polymer deposition rate is evaluated as in an LPCVD process.

The total deposition rate for all materials except the deposited material is:

$$R_m = R_{\text{deposition}}\,\Gamma_{\text{neutral depo}} - \frac{1}{\frac{1}{(1-A_m)\,R_{\text{etch},m}\,\Gamma_{\text{neutral etch}}} + \frac{1}{A_m\,R_{\text{etch},m}\gamma_m(\theta_{\text{im}})\,\Gamma_{\text{ion}} + R_{\text{des},m}}} \tag{39}$$

The total deposition rate for the deposited material is:

$$R_m = R_{\text{deposition}}\,\Gamma_{\text{neutral depo}} - R_{\text{etch},\,m}\gamma_m(\theta_{\text{im}})\Gamma_{\text{ion}} \tag{40}$$

where:

- $R_{\text{des},m}$ is the thermal desorption rate for material `m`; parameter `desorption_rate` in the command `add_material`.
- A_m is the anisotropy for material `m`; parameter anisotropy in the command `add_material`.
- $\Gamma_{\text{neutral etch}}$ is the total flux of neutrals, computed with a material-dependent sticking coefficient (specified with the `add_material` command).
- $\Gamma_{\text{neutral depo}}$ is the total flux of neutrals, computed with a material-independent sticking coefficient (specified with the `define_etch_machine` command).
- Γ_{ion} is the ion flux.

Spin-on-Glass Deposition Model

The `spin_on` model is used to determine the spin-on material profile over a given topographic substrate. The profile evolution of the film is computed by solving the Navier–Stokes equations under the lubrication theory for Newtonian fluids [8][9][10].

Given the initial thickness of the spin-on material (parameter `initial_thickness`), the profile evolution of the film is computed taking into account the capillarity and centrifugal forces acting on it. In addition, the film is supposed to evaporate during the process (parameter `evaporation_rate`). The spin-on material is characterized by its viscosity (parameter `viscosity`), its density (parameter `density`), and its surface tension with the surrounding fluid (parameter `surface_tension`).

The centrifugal force acting on the simulated structure is supposed to be constant over the simulation domain, and it is determined by the angular velocity of the wafer (parameter `angular_velocity`), the position of the simulated structure on the wafer (parameters `angular_position` and `radial_distance`), and the film density.

Reaction Models

Reaction models provide a very powerful and flexible means of describing the physical and chemical effects that occur on a wafer surface due to interaction with plasma species. Basic effects such as adsorption, reemission, sputtering, and chemical etching are described by surface reactions, using a syntax that is similar to that used for chemical reactions. Each reaction defines the interaction of a single gas-phase species with one or more surface species, along with the products that might result from this interaction. Reaction products can be emitted from the surface and eventually interact with other parts of the wafer surface, or they can be defined as being volatile, meaning that they have no effect in the simulation.

In the definition of a reaction model, unlike the definition of a level set–based model, there is no distinction between ion and neutral species. Indeed, a reaction model consists of a set of source species and a set of reactions. However, different properties (for example, angular distributions) can be assigned to each source species to effectively model both the neutral and the charged particles present in the reactor.

In addition, whereas for the level set–based models, the fluxes of the source species are always assumed to be normalized, reaction models require the specification of the absolute fluxes of the source species.

A reaction model is defined using the `define_model` command (see define_model on page 124). A source species is defined using the `add_source_species` command (see

add_source_species on page 98); whereas, reactions are defined using the `add_reaction` command (see add_reaction on page 88).

The angular distribution of each source species can have an arbitrary shape and must be specified with the `define_species_distribution` command (see define_species_distribution on page 133). However, the angular distributions of the source species are not part of any model. Only when defining a machine with the `define_etch_machine` command (see define_etch_machine on page 109), a reaction model is bound to the angular distributions of its source species.

As detailed in add_reaction on page 88, a reaction states a rule to transform a set of reactants into a set of products. Each reaction must contain one reactant species coming from the reactor, which must be either a source species or a species produced as a result of another reaction.

When the reactants become available, reactions are executed with a probability specified with the `add_reaction_properties` command (see add_reaction_properties on page 92). In the general case, the reaction probability depends on the angle between the traveling direction of the incoming species and the normal to the surface where the incoming species collides with the structure as well as on the energy of the incoming particle. The `define_probability` command allows you to define an energy-dependent and angle-dependent reaction probability (see define_probability on page 125). Since the execution of a reaction depends on a random decision, it is possible that no reaction occurs even if the reactants are available. When the reactants are available, but no reaction is actually executed, the incoming species is either stopped or reemitted in the reactor, as specified by the `default_event` parameter of the `define_species_properties` command (see define_species_properties on page 138).

References

[1] J. Li, *Topography Simulation of Intermetal Dielectric Deposition and Interconnection Metal Deposition Processes*, Ph.D. thesis, Stanford University, Stanford, CA, USA, March 1996.

[2] T. Mizuno *et al.*, “Analytical Model for Oblique Ion Reflection at the Si Surface,” *IEEE* Transactions on Electron Devices, vol. 35, no. 12, pp. 2323–2327, 1988.

[3] J. J. Vlassak, “A model for chemical–mechanical polishing of a material surface based on contact mechanics,” *Journal of the Mechanics and Physics of Solids*, vol. 52, no. 4, pp. 847–873, 2004.

[4] J. A. Sethian and Y. Shan, “Solving partial differential equations on irregular domains with moving interfaces, with applications to superconformal electrodeposition in semiconductor manufacturing,” *Journal of Computational Physics*, vol. 227, no. 13, pp. 6411–6447, 2008.

[5] R. Akolkar and U. Landau, "Mechanistic Analysis of the "Bottom-Up" Fill in Copper Interconnect Metallization," *Journal of the Electrochemical Society*, vol. 156, no. 9, pp. D351–D359, 2009.

[6] T. P. Moffat *et al*., "Curvature enhanced adsorbate coverage mechanism for bottom-up superfilling and bump control in damascene processing," *Electrochimica Acta*, vol. 53, no. 1, pp. 145–154, 2007.

[7] T. Langer, A. Belyaev, and H.-P. Seidel, "Spherical Barycentric Coordinates," in *Eurographics Symposium on Geometry Processing*, Cagliari, Sardinia, Italy, pp. 81–88, June 2006.

[8] S. Hirasawa *et al*., "Analysis of Drying Shrinkage and Flow Due to Surface Tension of Spin-Coated Films on Topographic Substrates," *IEEE Transactions on Semiconductor Manufacturing*, vol. 10, no. 4, pp. 438–444, 1997.

[9] L. M. Peurrung and D. B. Graves, "Film Thickness Profiles over Topography in Spin Coating," *Journal of the Electrochemical Society*, vol. 138, no. 7, pp. 2115–2124, 1991.

[10] L. E. Stillwagon and R. G. Larson, "Leveling of thin films over uneven substrates during spin coating," *Physics of Fluids A*, vol. 2, no. 11, pp. 1937–1944, 1990.

CHAPTER 4 Input Commands

This chapter describes the input commands used by Sentaurus Topography 3D.

Syntax of Commands and Parameters

All commands in Sentaurus Topography 3D follow the general structure:

```
command parameter1=value1 [parameter2=value2]
```

where `parameter1` and `parameter2` are command parameters that are set to `value1` and `value2`, respectively. Brackets are used to denote optional parameters.

In the following, the parameter values are denoted by an italic character inside angle brackets depending on their type:

- `<c>` – character value
- `<l>` – list of strings
- `<n>` – numeric value
- `<v>` – vector of numbers

Vectors are denoted using the Tcl array syntax (braces). For example, {0 0 1} is a vector normal to the horizontal plane.

The vertical bar is used to denote that either one parameter or another is expected by a command, for example:

```
command parameter1=value1 | parameter2=value2
```

where `command` expects either the parameter `parameter1` or the parameter `parameter2`.

In Units on page 23, the supported units are described. Different units can be set by specifying them inside angle brackets:

```
command parameter1=value1<unit1> [parameter2=value2<unit2>]
```

Examples can be found in Appendix A on page 247.

Command Summary

Table 8 summarizes the supported Sentaurus Topography 3D commands.

Table 8 Commands supported by Sentaurus Topography 3D

Command	Function
add_float_parameter	Adds a new user-defined parameter for floating-point values to the specified rate formula module (RFM) model.
add_flux_properties	Specifies the values of the parameters for a flux associated with an RFM model.
add_formula	Defines a formula for calculating the deposition or etching rate for an RFM model.
add_int_parameter	Adds a new user-defined parameter for integer values to the specified RFM model.
add_interface_layer	Adds a layer across the interface between two materials, where a material-dependent parameter can take position-dependent values, and specifies the spatially dependent values of that parameter in that layer.
add_ion_flux	Adds a new ion flux to the specified RFM model.
add_litho_command	Adds a Sentaurus Lithography Tcl command.
add_material	Specifies the values of the parameters of an etch machine for a particular material.
add_neutral_flux	Adds a new neutral flux to the specified RFM model.
add_reaction	Adds a reaction to a reaction model.
add_reaction_properties	Specifies the values of the parameters for a reaction.
add_source_species	Adds a species to the source of a reaction model.
define_boundary_conditions	Specifies the boundary conditions to be used when processing a structure.
define_deposit_machine	Defines a machine for a built-in or an RFM deposition model.
define_etch_machine	Defines a machine for an etching model, or a simultaneous etching and deposition model, or a deposition reaction model.
define_extraction	Defines extractions that can be used during a deposition step or an etching step.
define_iad	Defines an ion-angular distribution function (IADF).
define_litho_machine	Defines a machine for a lithographic process.
define_mask	Defines a mask to be used in an etch step or a patterning step.
define_model	Starts the definition of a new RFM model or a new reaction model.

Table 8 Commands supported by Sentaurus Topography 3D (Continued)

Command	Function
define_probability	Defines an energy- and angle-dependent probability that can be used in a reaction model.
define_reflection	Defines the properties of a new reflection function.
define_shape	Defines a new shape for a geometric etch or deposit step.
define_species_distribution	Defines the angular distribution of a source species of a reaction model.
define_species_properties	Defines the properties of a species used in a reaction model.
define_structure	Defines the initial 2D or 3D boundary structure.
define_yield	Defines the properties of a new yield function.
deposit	Performs a deposition process step on a structure.
etch	Performs an etching process step or a simultaneous etching and deposition process step on a structure, or performs a deposition step based on a reaction model.
extend_structure	Extends a structure by mirroring it, or by copying and shifting it.
extract	Extracts properties from a structure.
fill	Fills up the structure with a material.
filter_structure	Executes Boolean operations, or creates a copy of an existing structure, or decimates the number of surface elements, or smoothes a structure, or removes disconnected parts from a structure.
finalize_model	Indicates that the definition of a model is completed.
let	Defines global program settings.
litho	Performs a lithography simulation and adds a resist region.
pattern	Performs a patterning step on a structure.
remove_material	Removes a specified material from a structure.
save	Saves a structure, or an ion-angular distribution (IAD), or a probability function, or a reflection function, or a yield function to a TDR file, or saves a PMC structure to a PMC file.
set_orientation	Sets or changes the crystal orientation of a region.
truncate	Truncates a structure.

add_float_parameter

The `add_float_parameter` adds a new user-defined parameter for floating-point values to the specified RFM model.

For deposition:

```
add_float_parameter default=<n> model=<c> name=<n> quantity=<c> \
   [description=<c>] [max=<n>] [min=<n>] [optional=<c>]
```

For etching and simultaneous etching and deposition:

```
add_float_parameter default=<n> model=<c> name=<n> quantity=<c> \
   scope=<c> [description=<c>] [max=<n>] [min=<n>] [optional=<c>]
```

Table 9 Parameters of add_float_parameter command

Parameter	Type	Description	Default value [Possible values]	Unit
`default`	number	The default value of the float parameter.	none	Default unit of quantity specified with `quantity` parameter.
`description`	character	A description of what the float parameter models.	empty string	none
`max`	number	The maximum value of the float parameter.	∞	Default unit of quantity specified with `quantity` parameter.
`min`	number	The minimum value of the float parameter.	$-\infty$	Default unit of quantity specified with `quantity` parameter.
`model`	character	The RFM model to which the float parameter is added.	none	none

Table 9 Parameters of add_float_parameter command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
name	character	The name of the float parameter.	none When scope=global, these values cannot be used: applied_pressure, deposit_material, iad, material, maximum_error, model, name, pad_poisson_ratio, pad_roughness, pad_young_modulus, reflection, rotation, tilt, yield When scope=material_dependent, the value material cannot be used.	none
optional	character	Specifies whether the float parameter is optional when the model is used.	false [false, true]	none
quantity	character	The physical quantity of the value.	none [angle, dimensionless, energy, length, time, velocity]	none
scope	character	Specifies whether the float parameter is defined globally or per material.	none [global, material_dependent]	none

NOTE The model parameter must specify an RFM model, that is, the type parameter must have been set in the define_model command for that model.

For global parameters, the parameter default must be specified when optional=true, and it must not be specified when optional=false. The value specified with the parameter default is used when optional=true and the parameter has not been set explicitly with the define_deposit_machine or define_etch_machine command.

For material-dependent parameters, the parameter default must be set independently of the value of the parameter optional. The value specified with the parameter default is used for materials for which the parameters have not been specified with the add_material command. It is also used if optional=true and the parameter has not been set explicitly with the add_material command.

In general, it is not recommended to make a parameter optional. Instead, a model should be simplified to the degree where all user-defined parameters are required.

The parameter `default` specifies which value to assign to the new added parameter for unknown materials as well as which value to assign if the parameter is optional.

The parameter `quantity` specifies the physical quantity of the new added parameter. The expression for the rate defined with the `add_formula` command must have the dimension of a velocity.

The parameters `max` and `min` can be used to limit the valid range of the newly defined parameter.

For etching and simultaneous etching and deposition, the parameter `scope` specifies whether the new parameter is valid globally or is material dependent.

add_flux_properties

The `add_flux_properties` command specifies the parameters of a flux associated with an RFM model. There are different forms of the command for neutral and ion fluxes.

For neutral fluxes, you must specify the value of the sticking coefficient. For deposition models, the sticking coefficient is material independent (otherwise, it is material dependent):

```
add_flux_properties flux=<c> sticking=<n> [machine=<c>]

add_flux_properties flux=<c> material=<c> sticking=<n> [machine=<c>]
```

If `sputter_deposition` has been activated for an ion flux during the model definition (see add_ion_flux on page 81), the values for the parameters `sputter_exponent` (exponent of the angular distribution of the emission of the sputtered material), `sputter_type` (the type of emission of the sputtered material), and `sticking` (the sticking coefficient of the reemitted material) must be specified. Otherwise, these parameters must not be specified.

For ion fluxes, in deposition models, the flux parameters are material independent; whereas, for etching and simultaneous etching and deposition, they are material dependent:

```
add_flux_properties flux=<c> [machine=<c>] [sputter_exponent=<n>] \
   [sputter_type=<c>] [sticking=<n>]

add_flux_properties flux=<c> material=<c> [machine=<c>] \
   [sputter_exponent=<n>] [sputter_type=<c>] [sticking=<n>]
```

The `flux` parameter takes the name that was given to a flux in the model definition with the `add_neutral_flux` or `add_ion_flux` command.

If `sputter_exponent`, `sputter_type`, or `sticking` has been fixed in the model definition (given a constant value), the parameter cannot be specified again with the `add_flux_properties` command.

Table 10 Parameters of add_flux_properties command

Parameter	Type	Description	Default value [Possible values]	Unit
`flux`	character	The name of the flux to configure.	none	none
`machine`	character	The name of the machine for which the flux is configured.	`default_machine`	none
`material`	character	The material for which the properties are specified.	none	none

Table 10 Parameters of add_flux_properties command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
sputter_exponent	number	Exponent used to characterize the angular distribution $\cos^m\theta$ of the sputtered material. The specified value must be an integer.	none [1, ∞ [	none
sputter_type	character	Sets the sputter reemission type of the model (see add_ion_flux on page 81).	none [diffuse, reflective]	none
sticking	number	The sticking coefficient.	none [0, 1]	none

NOTE The machine parameter must specify a machine using an RFM model, that is, the type parameter must have been set in the define_model command for that model.

Example

```
# Define a model with sputter deposition enabled that fixes
# the sputter exponent and type, but not the sticking.
define_model name=m type=deposit description=""

add_ion_flux model=m name=i energy=independent reflection=false \
   sputtering=true sputter_deposition=true sputter_exponent=1 \
   sputter_type=diffuse

add_formula model=m expression="sputter_depo_flux(i)-direct_flux(i)"

finalize_model model=m

# ...
# When model is used:
define_deposit_machine model=m material=Oxide ...

# OK: Parameter sticking is required because sputter deposition is switched on
# in the model definition.
add_flux_properties model=m flux=i sticking=0.9

# Error: sputter_exponent has a constant value of 1 and cannot be
# specified again here when model is used.
add_flux_properties model=m flux=i sputter_exponent=100 sticking=0.9
```

add_formula

The `add_formula` command defines the formula for calculating the deposition or etching rate. To make it easier to develop and maintain an RFM model, you can define subexpressions and use them in the definition of subsequent subexpressions and the main expression for the rate.

For deposition and default rate formulas (material independent):

```
add_formula model=<c> name=<c> subexpression=<c> [unit=<c>]
add_formula expression=<c> model=<c> [unit=<c>]
```

For material-dependent rate formulas:

```
add_formula material=<c> model=<c> name=<c> subexpression=<c> [unit=<c>]
add_formula expression=<c> material=<c> model=<c> [unit=<c>]
```

Table 11 Parameters of add_formula command

Parameter	Type	Description	Default value [Possible values]	Unit
`expression`	character	The formula used to calculate the rate.	none	none
`material`	character	The material for which the formula is used.	none	none
`model`	character	The RFM model to which the formula is added.	none	none
`name`	character	The name of the subexpression.	none	none
`subexpression`	character	The formula used to calculate the subexpression.	none	none
`unit`	character	The unit of the expression or subexpression.	`um min^-1` `[nm min^-1, nm s^-1, um min^-1, um s^-1]`	none

NOTE The `model` parameter must specify an RFM model, that is, the `type` parameter must have been set in the `define_model` command for that model.

For deposition, the material-independent form of the `add_formula` command is used.

For etching and simultaneous etching and deposition, there is a material independent form and material dependent form of the `add_formula` command:

- The material-independent form is used to define a default rate formula. The default rate formula is used for all materials for which no specific rate formula has been defined.
- The material-dependent form is used to define a rate formula for a specific material.

Rate Calculation on page 223 describes the data and functions available for defining the formula for the rate. In addition, previously defined subexpressions can be used as part of the new rate formula.

add_int_parameter

The `add_int_parameter` adds a new user-defined parameter for integer values to the specified RFM model.

For deposition:

```
add_int_parameter default=<n> model=<c> name=<c> \
   [description=<c>] [max=<n>] [min=<n>] [optional=<c>]
```

For etching and simultaneous etching and deposition:

```
add_int_parameter default=<n> model=<c> name=<c> scope=<c> \
   [description=<c>] [max=<n>] [min=<n>] [optional=<c>]
```

Table 12 Parameters of add_int_parameter command

Parameter	Type	Description	Default value [Possible values]	Unit
`default`	number	The default value of the integer parameter.	none	none
`description`	character	A description of what the parameter models.	empty string	none
`max`	number	The maximum value of the integer parameter.	`2147483647` `[-2147483648, 2147483647]`	none
`min`	number	The minimum value of the integer parameter.	`-2147483648` `[-2147483648, 2147483647]`	none
`model`	character	The RFM model to which the integer parameter is added.	none	none
`name`	character	The name of the integer parameter.	none When `scope=global`, these values cannot be used: `applied_pressure`, `deposit_material`, `iad`, `material`, `maximum_error`, `model`, `name`, `pad_poisson_ratio`, `pad_roughness`, `pad_young_modulus`, `reflection`, `rotation`, `tilt`, `yield` When `scope=material_dependent`, the `material` value cannot be used.	none

Table 12 Parameters of add_int_parameter command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
optional	character	Specifies whether the integer parameter is optional when the model is used.	false [false, true]	none
scope	character	Specifies whether the integer parameter is defined globally or per material.	none [global, material_dependent]	none

NOTE The model parameter must specify an RFM model, that is, the type parameter must have been set in the define_model command for that model.

For global parameters, the parameter default must be specified when optional=true, and it must not be specified when optional=false. The value specified with the parameter default is used when optional=true and the parameter has not been set explicitly with the define_deposit_machine or define_etch_machine command.

For material-dependent parameters, the parameter default must be set independently of the value of the parameter optional. The value specified with the parameter default is used for materials for which the parameters have not been specified with the add_material command. It is also used if optional=true and the parameter has not been set explicitly with the add_material command.

In general, it is not recommended to make a parameter optional. Instead, a model should be simplified to the degree where all user-defined parameters are required.

The parameter default specifies which value to assign to the new added parameter for unknown materials as well as which value to assign if the parameter is optional.

The parameters max and min can be used to limit the valid range of the newly defined parameter.

For etching and simultaneous etching and deposition, the parameter scope specifies whether the new parameter is valid globally or is material dependent.

add_interface_layer

The `add_interface_layer` command adds a layer across the interface between two different materials, where a numeric material-dependent parameter can take position-dependent values, and specifies the spatially dependent values of that parameter in that layer.

NOTE The `add_interface_layer` command can be used only for machines using level set–based models. This command cannot be used for deposition machines or for machines using simultaneous etching and deposition models.

This command uses the initial structure to determine the spatially dependent parameter values and has two versions.

The first version allows you to specify a linear grading of the specified parameter across the interface between its bulk values, which are specified with the `add_material` command:

```
add_interface_layer material1=<c> material2=<c> parameter=<c> \
   thickness1=<n> thickness2=<n> [machine=<c>]
```

The second version allows you to specify a piecewise linear grading of the specified parameter across the interface between its bulk values, which are specified with the `add_material` command:

```
add_interface_layer material1=<c> material2=<c> parameter=<c> \
   table1=<v> table2=<v> [machine=<c>] \
   [table1_distance_unit=<c>] [table2_distance_unit=<c>]
```

Table 13 Parameters of add_interface_layer command

Parameter	Type	Description	Default value [Possible values]	Unit
`machine`	character	The name of the machine to which an interface layer is added.	`default_machine`	none
`material1`	character	The name of the first material of the interface. It must be a material different from the material specified with `material2`.	none	none
`material2`	character	The name of the second material of the interface. It must be a material different from the material specified with `material1`.	none	none

Table 13 Parameters of add_interface_layer command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
parameter	character	The name of the material-dependent parameter for which an interface layer is specified.	none One of the material-dependent parameter names of the machine specified by machine. For RFM models, it must be one of the parameters defined using the add_float_parameter command.	none
table1	vector	Tabular format of the piecewise linear scaling of the parameter value in the region of space occupied by the material specified by material1 (see below for details).	none	none
table1_distance_unit	character	The unit of the distances listed in table1.	um [nm, m, um]	none
table2	vector	Tabular format of the piecewise linear scaling of the parameter value in the region of space occupied by the material specified by material2 (see below for details).	none	none
table2_distance_unit	character	The unit of the distances listed in table2.	um [nm, m, um]	none
thickness1	number	The thickness of the interface layer in the region of space occupied by the material specified by material1.	none [0, ∞ [	µm
thickness2	number	The thickness of the interface layer in the region of space occupied by the material specified by material2.	none [0, ∞ [	µm

NOTE Currently, when using a built-in model, the add_interface_layer command is not supported for the sticking and reflection parameters.

The parameters table1 and table2 require a vector of numbers that fulfill the following conditions (the convention that the first vector element has index 0 is understood):

- The number of elements of the vector must be even and not less than 4.
- Vector elements with an even index represent a distance from the interface and must be nonnegative, unique, and sorted in ascending order.

- Vector elements with an odd index represent a scaling factor for the corresponding bulk value of the parameter specified with the `add_material` command and must be nonnegative.
- The first vector element (that is, the first distance) must be 0.
- The last vector element (that is, the last scaling factor) must be 1.

Examples

The following commands define an etch machine using the `simple` model, where the `rate` parameter values vary linearly across the interfaces between silicon and oxide:

```
define_etch_machine model=simple
add_material material=Silicon rate=1. anisotropy=0.5 curvature=0.
add_material material=Oxide rate=0.1 anisotropy=0.5 curvature=0.
add_material material=Nitride rate=0.5 anisotropy=0.5 curvature=0.
add_interface_layer material1=Silicon material2=Oxide parameter=rate \
   thickness1=0.1 thickness2=0.2
```

In particular:

- The value of the `rate` parameter is $1\ \mu m/min$ for points that are located in a region occupied by material `Silicon` and with a distance from a silicon–oxide interface greater than $0.1\ \mu m$.
- The value of the `rate` parameter is $0.1\ \mu m/min$ for points that are located in a region occupied by material `Oxide` and with a distance from a silicon–oxide interface greater than $0.2\ \mu m$.
- The value of the `rate` parameter decreases linearly from $1\ \mu m/min$ to $0.1\ \mu m/min$ as you move along a segment that crosses the silicon–oxide interface orthogonally, starting from a point located in the silicon region at $0.1\ \mu m$ from that interface and up to a point located in the oxide region at $0.2\ \mu m$ from that interface.
- The value of the `rate` parameter is $0.5\ \mu m/min$ for points located in a region occupied by material `Nitride`.

The following commands define an etch machine using an RFM model, where the `R` parameter values vary linearly across the interfaces between silicon and oxide:

```
define_model name=m type=etch description="isotropic RFM etch model"
add_float_parameter name=R model=m quantity=velocity \
   scope=material_dependent default=0
add_formula model=m expression="-R()"
finalize_model model=m

define_etch_machine model=m
add_material material=Silicon R=1
add_material material=Oxide R=2
```

```
add_interface_layer parameter=R material1=Silicon material2=Oxide \
   table1={0 0.1 1 0.5 2 1} table1_distance_unit=nm \
   table2={0 0.1 1 0.2 1.5 1} table2_distance_unit=nm
```

In particular:

- The value of the `R` parameter is 1 µm/min for points that are located in a region occupied by material `Silicon` and with a distance from a silicon–oxide interface greater than 2 nm.
- The value of the `R` parameter is 2 µm/min for points that are located in a region occupied by material `Oxide` and with a distance from a silicon–oxide interface greater than 1.5 nm.
- The value of the `R` parameter changes in a piecewise linear way from 1 µm/min to 2 µm/min as you move along a segment that crosses the silicon–oxide interface orthogonally, starting from a point located in the silicon region at 2 nm from that interface and up to a point located in the oxide region at 1.5 nm from that interface.
- The values of the `R` parameter on the silicon side of the interface layer are determined by multiplying the bulk value of the `R` parameter (that is, 1 µm/min) by the piecewise linear scaling factor specified by the parameter `table1`.
- The values of the `R` parameter on the oxide side of the interface layer are determined by multiplying the bulk value of the `R` parameter (that is, 2 µm/min) by the piecewise linear scaling factor specified by the parameter `table2`.

NOTE When using the `etch` command with a machine having interface layers defined, it is recommended to set the `spacing` parameter such that the spatial changes of the parameter values can be properly resolved.

add_ion_flux

The `add_ion_flux` command adds a new ion flux to the specified RFM model.

```
add_ion_flux energy=<c> model=<c> name=<c> reflection=<c> \
   sputter_deposition=<c> sputtering=<c> \
   [sputter_exponent=<n>] [sputter_type=<c>] [sticking=<n>]
```

Table 14 Parameters of add_ion_flux command

Parameter	Type	Description	Default value [Possible values]	Unit
`energy`	character	Specifies whether to perform flux integration depending on the kinetic energy of the ion species.	`independent` `[dependent, independent]`	none
`model`	character	The RFM model to which the ion flux is added.	none	none
`name`	character	The name to be given to the ion flux.	none	none
`reflection`	character	Specifies whether ion reflection is taken into account.	`false` `[false, true]`	none
`sputter_ deposition`	character	Indicates whether redeposition of sputtered material is calculated. **NOTE** This parameter must be specified if `sputtering=true`. Otherwise, it must not be specified.	none `[false, true]`	none
`sputter_exponent`	number	The distribution exponent of the sputtered material.	none `[1, ∞ [`	none
`sputter_type`	character	Sets the sputter reemission type of the model.	none `[diffuse, reflective]`	none
`sputtering`	character	Specifies whether sputtering is taken into account.	`false` `[false, true]`	none
`sticking`	number	The sticking coefficient.	none `[0, 1]`	none

NOTE The `model` parameter must specify an RFM model, that is, the `type` parameter must have been set in the `define_model` command for that model.

The name of the ion flux is used to reference it when configuring the flux properties using the `add_flux_properties` command and when accessing the flux values in the formula for calculating the etching and deposition rates.

The parameter `energy` specifies whether reflection and sputtering are energy dependent.

If redeposition of the sputtered material is activated, the optional parameters `sputter_exponent`, `sputter_type`, and `sticking` can be used to define constant values for the exponent of the angular distribution of the emission of the sputtered material, the type of emission of the sputtered material, and the sticking coefficient of the reemitted material, respectively. If any of these parameters is defined with the `add_ion_flux` command, its value cannot be changed with a subsequent `add_flux_properties` command.

add_litho_command

The `add_litho_command` command adds a Sentaurus Lithography Tcl command to a lithography machine identified by its name. A lithography machine must be created with the `define_litho_machine` command before using the `add_litho_command` command. The `machine` parameter specifies the name of the machine to which the Tcl command is added.

In general, material and simulation parameters should be specified in the SLO file or the Sentaurus Lithography material database. The `add_litho_command` command should be used mainly for varying a limited number of parameters, for example, in a parameterized Sentaurus Workbench project.

```
add_litho_command machine=<c> <slitho command> <slitho arguments>
```

Table 15 Parameters of add_litho_command command

Parameter	Type	Description	Default value [Possible values]	Unit
`machine`	character	Name of the lithography machine. This name can be used in subsequent `litho` commands for creating a resist region by a lithography simulation.	none	none
`<slitho command>`	character	Sentaurus Lithography Tcl command.	none	none
`<slitho arguments>`	character	Arguments to a Sentaurus Lithography Tcl command, without the Sentaurus Lithography connection handle.	none	none

NOTE Users are responsible for specifying the correct parameters depending on the specified Sentaurus Lithography Tcl command.

add_material

The `add_material` command defines the material-dependent properties of an etch machine (`define_etch_machine`). The parameters for different models are defined as follows:

Simple etching:

```
add_material anisotropy=<n> curvature=<n> material=<c> rate=<n> [machine=<c>]
```

Dry etching:

```
add_material material=<c> rate=<n> s1=<n> s2=<n> [machine=<c>]
```

Wet etching:

```
add_material material=<c> rate=<n> [deactivation_rate=<n>] [density=<n>] \
   [machine=<c>]
```

Simultaneous etching and deposition (etchdepo):

```
add_material material=<c> rate=<n> s1=<n> s2=<n> [machine=<c>] \
   [sputter_type=<c>] [reflection=<n>]
```

Simultaneous etching and deposition 2 (etchdepo2):

```
add_material anisotropy=<n> material=<c> rate=<n> s1=<n> s2=<n> \
   sticking=<n> [desorption_rate=<n>] [machine=<c>] [reflection=<n>]
```

High-density plasma etching:

```
add_material anisotropy=<n> material=<c> rate=<n> s1=<n> s2=<n> [machine=<c>]
```

High-density plasma 2 etching:

```
add_material anisotropy=<n> material=<c> rate=<n> s1=<n> s2=<n> \
   sputter_rate=<n> sticking=<n> [machine=<c>] [reflection=<n>]
```

Ion-enhanced etching:

```
add_material anisotropy=<n> material=<c> rate=<n> s1=<n> s2=<n> \
   sticking=<n> [desorption_rate=<n>] [machine=<c>] [reflection=<n>]
```

Ion-milling:

```
add_material anisotropy=<n> material=<c> rate=<n> s1=<n> s2=<n> [machine=<c>]
```

Reactive ion etching:

```
add_material anisotropy=<n> material=<c> rate=<n> [machine=<c>]
```

Reactive ion etching 2:

```
add_material anisotropy=<n> material=<c> rate=<n> sticking=<n> [machine=<c>] \
   [reflection=<n>]
```

RFM model:

```
add_material material=<c> [machine=<c>] ...
```

NOTE The `add_material` command is not supported by etch machines having `model=crystal` because it is not needed. The properties of the material etched by an etching machine having `model=crystal` are set using the `define_etch_machine` command. The crystallographic orientation of the regions containing the material etched by an etching machine having `model=crystal` can be set with the command `set_orientation`. See Orientation-Dependent Models on page 38, Crystallographic Orientation–Dependent Etching on page 58, define_etch_machine on page 109, and set_orientation on page 205.

NOTE The `add_material` command is not supported by etching machines using reaction models because it is not needed. In fact, reaction models use reactions to specify material-dependent behavior. The properties of reactions are set using the `add_reaction_properties` command (see add_reaction_properties on page 92).

Table 16 Parameters of add_material command

Parameter	Type	Description	Default value [Possible values]	Unit
`anisotropy`	number	Anisotropy coefficient for the current material.	none `[0, 1]`	none
`curvature`	number	Curvature coefficient for the current material.	none `[0, 0.1]`	µm
`deactivation_rate`	number	The rate at which etchants are deactivated when reaching the surface.	`0` `]0, ∞[`	µm min^{-1}
`density`	number	The volume density of the current material.	`1` `]0, ∞[`	mol cm^{-3}
`desorption_rate`	number	Thermal desorption rate for the material.	`0` `[0, ∞[`	µm min^{-1}
`machine`	character	The machine to which the command is applied.	`default_machine`	none
`material`	character	Sets the material.	none	none
`rate`	number	Sets the etching rate for the current material.	none	µm min^{-1}

Table 16 Parameters of add_material command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
reflection	number	Specifies the reflection parameter used to evaluate the reflection probability [1].	0 [0, 1]	none
s1	number	The first sputter coefficient.	none	none
s2	number	The second sputter coefficient.	none	none
sputter_rate	number	Sets the sputter rate.	none [0,∞ [	$\mu m\ min^{-1}$
sputter_type	character	Sets the angular distribution type of the sputtered material. The value reflective is only supported for the radiosity method for flux integration.	diffuse [diffuse, reflective]	none
sticking	number	The sticking coefficient.	none [0,1]	none

The add_material command can be used several times for the same etch machine to configure the properties of different materials. A material cannot be configured with the add_material command more than once.

The machine to be configured is referenced with the machine parameter that must match one of the names of the previously defined etch machines.

Examples

This command configures the material-dependent parameters of the previously defined machine etchmachine for the material Silicon, with rate of 0.5 µm/minute , an anisotropy coefficient of 0.3, and a curvature coefficient of 0.05 µm :

```
add_material machine=etchmachine material=Silicon anisotropy=0.3 \
   curvature=0.05 rate=0.5
```

NOTE For materials that are contained in the initial TDR structure, whose properties have not been defined with the add_material command, the etch rate is zero.

add_neutral_flux

The `add_neutral_flux` command adds a new neutral flux to the specified RFM model.

```
add_neutral_flux model=<c> name=<c> [sticking=<n>]
```

Table 17 Parameters of add_neutral_flux command

Parameter	Type	Description	Default value [Possible values]	Unit
`model`	character	The RFM model to which the neutral flux is added.	none	none
`name`	character	The name to be given to the neutral flux.	none	none
`sticking`	number	The sticking coefficient.	none `[0, 1]`	none

NOTE The `model` parameter must specify an RFM model, that is, the `type` parameter must have been set in the `define_model` command for that model.

The name of the neutral flux is used to reference it when configuring the flux properties using the `add_flux_properties` command and when accessing the flux values in the formula for calculating the etching and deposition rates.

The optional parameter `sticking` can be used to define a constant for the sticking coefficient. If this parameter is used, the sticking coefficient for this flux cannot be changed with a subsequent `add_flux_properties` command.

add_reaction

The `add_reaction` command defines a reaction of a reaction model.

```
add_reaction expression=<c> model=<c> name=<c>
```

Table 18 Parameters of add_reaction command

Parameter	Type	Description	Default value [Possible values]	Unit
`expression`	character	The expression of the reaction added to the model.	none	none
`model`	character	The name of the model to which the reaction is added.	none	none
`name`	character	The name of the reaction.	none	none

NOTE The `model` parameter must specify a reaction model, that is, the `type` parameter must not have been set in the `define_model` command for that model.

A reaction specifies a rule to transform a set of reactants into a set of products. The transformation specified by the `add_reaction` command is irreversible.

A reaction expression consists of two reaction states:

- The first reaction state describes the state *before* the reaction (specifying the *reactants*).
- The second reaction state describes the state *after* the reaction (specifying the *products*).

The reaction states are separated by the equal sign (=).

Each reaction state consists of one or more chemical species terms. The chemical species terms are separated by the plus sign (+).

A chemical species term consists of a chemical symbol and a species-type modifier (see Table 19).

Table 19 Valid species-type modifiers

Modifier	Description
`<b>`	Bulk species.
`<g>`	Gaseous species.
`<p>`	Sputtered product species that is tracked by the simulator.
`<q>`	Sputtered product species that is not tracked by the simulator.

Table 19 Valid species-type modifiers (Continued)

Modifier	Description
<r>	Reflected product.
<s>	Surface species.
<v>	Product that is not tracked by the simulator.

Limitations

The following limitations apply:

- Each reaction must have exactly two reactants: one with the <g> species-type modifier and one with the <s> species-type modifier.
- Each reaction is allowed to have, at most, one product that goes into the reactor and that is tracked. The species-type modifiers that denote products that go into the reactor and that are tracked are <g>, <p>, and <r>.

Example of Adsorption Reaction

The following statement adds a reaction named r1 to the model reaction_model:

```
add_reaction model=reaction_model expression="F<g> + Silicon<s> = SiF<s>" \
   name=r1
```

According to the specified expression, when an atom of species F coming from the reactor reacts with an atom of Silicon on the surface of the processed structure, a SiF molecule is produced. The effect of such a reaction is that a SiF molecule is adsorbed.

Example of Etching Reactions

The following statements add two reactions r2 and r3 to the model reaction_model:

```
add_reaction model=reaction_model \
   expression="Ar<g> + Silicon<s> = Silicon<v>" name=r2
add_reaction model=reaction_model expression="Cl<g> + Silicon<s> = SiCl<g>" \
   name=r3
```

According to reaction r2, when an atom of species Ar coming from the reactor reacts with an atom of Silicon on the surface of the processed structure, that silicon atom is removed without being tracked by the simulator.

According to reaction r3, when an atom of species Cl coming from the reactor reacts with an atom of Silicon on the surface of the processed structure, a SiCl molecule is produced and it goes into the reactor and is tracked by the simulator. The net effect of this reaction is that a silicon atom is etched from the structure.

Example of Deposition Reactions

The following statement adds a reaction named `r4` to the model `reaction_model`:

```
add_reaction model=reaction_model \
   expression="SiH2<g> + Silicon<s> = Silicon<s> + Silicon<b> + H2<v>" \
   name=r4
```

According to the specified expression, when a molecule of species `SiH2` coming from the reactor reacts with an atom of `Silicon` on the surface of the processed structure, a `Silicon` atom goes into the bulk and a new `Silicon` atom is deposited on the surface of the structure. Moreover, a `H2` molecule is produced, but it is not tracked by the simulator. Therefore, from the perspective of the simulation, the following command is equivalent to the previous one:

```
add_reaction model=reaction_model \
   expression="SiH2<g> + Silicon<s> = Silicon<s> + Silicon<b>" name=r4
```

However, including the chemical species term `H2<v>` makes the reaction clearer and more readable.

Example of Sputtering Reactions

The following statements add two reactions `r5` and `r6` to the model `reaction_model`:

```
add_reaction model=reaction_model \
   expression="I<g> + Silicon<s> = Silicon<p>" name=r5
add_reaction model=reaction_model \
   expression="Ar<g> + Photoresist<s>= Photoresist<q>" name=r6
```

According to reaction `r5`, when an atom of species `I` coming from the reactor reacts with an atom of `Silicon` on the surface of the processed structure, a `Silicon` atom is sputtered and tracked by the simulator. Since the sputtered atom is tracked, it can be deposited somewhere if the model contains a proper reaction for silicon deposition.

According to reaction `r6`, when an atom of species `Ar` coming from the reactor reacts with an atom of `Photoresist` on the surface of the processed structure, a `Photoresist` atom is sputtered off the structure. However, the removed atom is not tracked by the simulator. As a consequence, this reaction models the sputtering of Photoresist by Ar, but it cannot be used if the deposition of the sputtered Photoresist must be modeled.

Example of Reflection Reaction

The following statement adds a reaction named `r7` to the model `reaction_model`:

```
add_reaction model=reaction_model \
   expression="I<g> + Nitride<s> = I<r> + Nitride<s>" name=r7
```

According to reaction `r7`, when an atom of species `I` coming from the reactor reacts with an atom of `Nitride` on the surface of the processed structure, the `Nitride` atom is left untouched; whereas, the `I` atom is reflected.

add_reaction_properties

The `add_reaction_properties` command specifies the parameters of a reaction.

For reflection reactions (see add_reaction on page 88):

```
add_reaction_properties machine=<c> (p=<n> | probability=<c>) reaction=<c> \
   [activation_energy=<n> | (energy_exponent=<n> energy_reference=<n> \
    energy_threshold=<n>)] \
   [product_energy_factor=<n> | (product_energy_max=<n> \
    product_energy_min=<n>)] \
   [reflection_exponent=<n>]
```

For sputtering reactions (see add_reaction on page 88):

```
add_reaction_properties machine=<c> (p=<n> | probability=<c>) reaction=<c> \
   [activation_energy=<n> | (energy_exponent=<n> energy_reference=<n> \
    energy_threshold=<n>)] \
   [product_energy_factor=<n> | (product_energy_max=<n> \
    product_energy_min=<n>)] \
   [sputter_exponent=<n>] (sputter_gamma=<n> | [sputter_type=<c>])
```

For the other reactions:

```
add_reaction_properties machine=<c> (p=<n> | probability=<c>) reaction=<c> \
   [activation_energy=<n> | (energy_exponent=<n> energy_reference=<n> \
    energy_threshold=<n>)] \
   [product_energy_factor=<n> | (product_energy_max=<n> \
    product_energy_min=<n>)]
```

Table 20 Parameters of add_reaction_properties command

Parameter	Type	Description	Default value [Possible values]	Unit
`activation_energy`	number	Specifies the value of the activation energy parameter E_a in Eq. 41.	`0` `[0, ∞ [`	eV
`energy_exponent`	number	Specifies the value of the exponent parameter m in Eq. 42.	none `]0, ∞ [`	none
`energy_reference`	number	Specifies the value of the parameter E_{ref} in Eq. 42.	none `]0, ∞ [`	eV
`energy_threshold`	number	Specifies the value of the parameter E_{th} in Eq. 42.	none `]0, ∞ [`	eV
`machine`	character	The name of the machine for which the reaction is configured.	none	none

Table 20 Parameters of add_reaction_properties command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
p	number	The energy- and angle-independent value of the reaction probability.	none [0, 1]	none
probability	character	The name of the probability function defined with the define_probability command.	none	none
product_energy_factor	number	Sets the ratio between the energy of any reaction products with the <g>, <p>, or <r> species-type modifier and the energy of the gaseous reactant of the reaction being specified.	1 [0, 1]	none
product_energy_max	number	Sets the maximum energy of the products with the <g>, <p>, or <r> species-type modifier of the reaction being specified, when they are assumed to have a uniform energy distribution.	none]0, ∞[	eV
product_energy_min	number	Sets the minimum energy of the products with the <g>, <p>, or <r> species-type modifier of the reaction being specified, when they are assumed to have a uniform energy distribution.	none]0, ∞[	eV
reaction	character	The name of the reaction to configure.	none	none
reflection_exponent	number	The exponent of the angular distribution $\cos^m\theta$ of the reflected species. If omitted, all particles are reflected along the direction specular to the incoming one with respect to the surface normal.	none [1, 2147483647]	none
sputter_exponent	number	The exponent of the angular distribution $\cos^m\theta$ of the sputtered species.	1]0, ∞[	none
sputter_gamma	number	The value of γ in Eq. 43, p. 95 that controls the main direction of the emission of the sputtered material.	none]−∞, ∞[	none
sputter_type	character	The sputter reemission type of the model.	diffuse [diffuse, reflective]	none

When the `activation_energy` parameter is specified, the following energy-dependent reaction probability $p(E, \theta)$ is assumed for the reaction at hand:

$$p(E, \theta) = p(\theta)\exp\left(-\frac{E_a}{E}\right) \tag{41}$$

In Eq. 41:

- $p(\theta)$ denotes the probability specified with the `p` parameter or the angle-dependent probability function specified with the `probability` parameter.
- E_a denotes the value given to the `activation_energy` parameter. It is worth noting that, according to Eq. 41, the probability is not zero for energies smaller than the value of the `activation_energy` parameter, but it decays rapidly as the energy falls below such a value.

If the `probability` parameter specifies an energy-dependent probability, an error will be issued in this case.

When the parameters `energy_exponent`, `energy_reference`, and `energy_threshold` are specified, the following energy-dependent reaction probability $p(E, \theta)$ is assumed for the reaction at hand [2]:

$$p(E, \theta) = p(\theta) \cdot \begin{cases} 0 & E \le E_{th} \\ \dfrac{E^m - E_{th}^m}{E_{ref} - E_{th}} & E_{th} < E < E_{ref} \\ 1 & E \ge E_{ref} \end{cases} \tag{42}$$

In Eq. 42:

- $p(\theta)$ denotes the probability specified with the `p` parameter or the angle-dependent probability function specified with the `probability` parameter. It is the probability when the energy is greater than or equal to the value specified by the `energy_reference` parameter.
- m denotes the value given to the `energy_exponent` parameter.
- E_{ref} denotes the value given to the `energy_reference` parameter given in electron volts.
- E_{th} denotes the value given to the `energy_threshold` parameter given in electron volts.

If the `probability` parameter specifies an energy-dependent probability, an error will be issued in this case.

The `product_energy_factor`, `product_energy_max`, and `product_energy_min` parameters can be specified only for reactions that have at least one product with the `<g>`, `<p>`, or `<r>` species-type modifier.

When parameter `product_energy_factor` is specified, the energy of the product having the `<g>`, `<p>`, or `<r>` species-type modifier will be proportional to the energy of the reactant with the `<g>` species-type modifier, and the proportionality constant will be the value of the `product_energy_factor` parameter.

When parameters `product_energy_max` and `product_energy_min` are specified, the energy of the product having the `<g>`, `<p>`, or `<r>` species-type modifier will be uniformly distributed over the energy window defined by the values of these two parameters.

NOTE The following parameters are available only when the machine specified with the `machine` parameter is energy dependent: `activation_energy`, `energy_exponent`, `energy_reference`, `energy_threshold`, `product_energy_factor`, `product_energy_max`, and `product_energy_min`.

The main direction in which sputtered particles are emitted can be controlled with the parameter `sputter_gamma`. It is used to specify the value of the parameter γ in the function $f(\theta, \gamma)$. This function is used to determine the angle between the surface normal and the symmetry axis of the sputter emission distribution.

The symmetry axis of the sputter emission distribution lies in the plane defined by the direction of the incoming particles and the surface normal. The angle between the symmetry axis of the sputter emission distribution and the surface normal is considered to be negative if the symmetry axis of the sputter emission distribution is on the same side of the surface normal as the direction of the incoming particles. Otherwise, it is considered to be positive.

$$f(\theta,\gamma) = \begin{cases} \gamma\theta & \text{for } 0 \le |\gamma| \le 1 \\ \operatorname{sgn}(\gamma)\dfrac{\pi}{2}\left(1-\left(1-\dfrac{\theta}{\frac{\pi}{2}}\right)^{|\gamma|}\right) & \text{for } 1 \le |\gamma| \end{cases} \tag{43}$$

where:

$$\operatorname{sgn}(x) = \begin{cases} +1 & \text{for } x > 0 \\ 0 & \text{for } x = 0 \\ -1 & \text{for } x < 0 \end{cases} \tag{44}$$

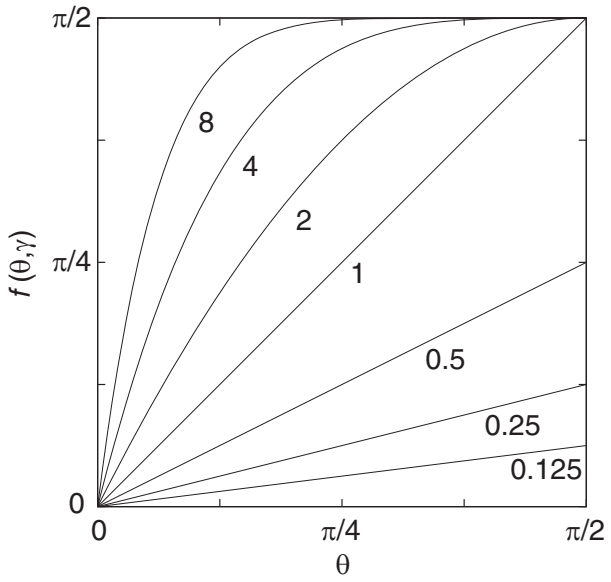


Figure 16 Function $f(\theta, \gamma)$ for the values of γ = 0.125, 0.25, 0.5, 1, 2, 4, and 8: The horizontal axis represents the angle θ between the surface normal and the incoming particle. The vertical axis represents the angle between the surface normal and the symmetry axis of the sputter emission distribution.

Figure 16 shows $f(\theta, \gamma)$ for different values of γ. The horizontal axis represents the angle θ between the surface normal and the incoming particle. The vertical axis represents the angle between the surface normal and the symmetry axis of the sputter emission distribution.

For $\gamma = 0$, the symmetry axis of the sputter emission distribution is perpendicular to the surface. Therefore, `sputter_gamma=0` gives the same result as `sputter_type=diffuse`.

For $\gamma = 1$, the symmetry axis of the sputter emission distribution is the reflected direction of the incoming particle. Therefore, `sputter_gamma=1` gives the same result as `sputter_type=reflective`.

For $\gamma < 0$, the sputtered particles are not emitted forward as for positive values of γ, but back in the direction of the incoming particle.

Examples

```
# Reaction model definition
define_model name=m description=""

add_source_species model=m name=I
add_source_species model=m name=N
```

```
# Adsorption of species N on species Silicon
add_reaction model=m name=adsorption_reaction \
   expression="N<g> + Silicon<s> = SiliconN<s>"

# Reflection of species I by species Silicon
add_reaction model=m name=reflection_reaction \
   expression="I<g> + Photoresist<s> = I<r> + Photoresist<s>"

# Etching of species SiliconN by species I
add_reaction model=m name=ion_etch_reaction \
   expression="I<g> + SiliconN<s> = SiliconN<v>"

# Sputtering of species Silicon by species I
add_reaction model=m name=sputtering_reaction \
   expression="I<g> + Silicon<s> = Silicon<p>"

finalize_model model=m

# ...

# Define the yield function for sputtering of Silicon by species I
define_yield name=my_yield energy=0 species=I material=Silicon theta_max=60 \
   yield_max=1.4

# Define the probability function for reflection of species I from Photoresist
define_probability name=my_probability energy=0 mizuno_k=0.05

# Define a machine using the reaction model 'm'. Since the used model contains
# sputtering reactions, yield functions must be provided when defining the
# machine.
define_etch_machine model=m yield=my_yield ...

# Reactions 'adsorption_reaction' and 'ion_etch_reaction' are neither
# reflection nor sputtering reactions. Therefore, the parameter 'p' is used to
# set their angle-independent probabilities.
add_reaction_properties reaction=adsorption_reaction p=0.9
add_reaction_properties reaction=ion_etch_reaction p=0.7

# Reaction 'reflection_reaction' is a reflection reaction. Therefore, it is
# possible to set parameter 'reflection_exponent' for it. Moreover, the
# probability function defined with the 'define_probability' command
# will be used as the reaction probability.
add_reaction_properties reaction=reflection_reaction \
   reflection_exponent=1000 probability=my_probability

# Reaction 'sputtering_reaction' is a sputtering reaction. Therefore, it is
# possible to set parameter 'sputter_exponent' for it. The default value of
# parameter 'sputter_type' is used here.
add_reaction_properties reaction=sputtering_reaction p=0.8 \
   sputter_exponent=100
```

add_source_species

The `add_source_species` command adds a new source species to the specified reaction model.

```
add_source_species model=<c> name=<c>
```

Table 21 Parameters of add_source_species command

Parameter	Type	Description	Default value [Possible values]	Unit
`model`	character	The name of the model to which the source species is added.	none	none
`name`	character	The name of the source species to add.	none	none

NOTE The `model` parameter must specify a reaction model, that is, the `type` parameter must not have been set in the `define_model` command for that model.

define_boundary_conditions

The `define_boundary_conditions` command defines the boundary conditions to apply when using a structure in a simulation.

```
define_boundary_conditions [structure=<c>] [x=<c>] [y=<c>] [xmin=<c>] \
   [xmax=<c>] [ymin=<c>] [ymax=<c>]
```

Table 22 Parameters of define_boundary_conditions command

Parameter	Type	Description	Default value [Possible values]	Unit
`structure`	character	Sets the name of the structure for which boundary conditions must be defined.	`default_structure`	none
`x`	character	Sets the boundary conditions for both planes x = x_min and x = x_max, where x_min and x_max are the minimum and the maximum x-value in the simulation domain, respectively.	`none` `[none, periodic, reflective]`	none
`y`	character	Sets the boundary conditions for both planes y = y_min and y = y_max, where y_min and y_max are the minimum and the maximum y-value in the simulation domain, respectively.	`none` `[none, periodic, reflective]`	none
`xmax`	character	Sets the boundary conditions for the plane x = x_max, where x_max is the maximum x-value in the simulation domain.	`reflective` `[none, reflective]`	none
`xmin`	character	Sets the boundary conditions for the plane x = x_min, where x_min is the minimum x-value in the simulation domain.	`reflective` `[none, reflective]`	none
`ymax`	character	Sets the boundary conditions for the plane y = y_max, where y_max is the maximum y-value in the simulation domain.	`reflective` `[none, reflective]`	none
`ymin`	character	Sets the boundary conditions for the plane y = y_min, where y_min is the minimum y-value in the simulation domain.	`reflective` `[none, reflective]`	none

Examples

This command sets the boundary conditions on the two x-bounding planes of the structure called `default_structure` and uses the default ones on the two y-bounding planes. The plane with the minimum x will be considered as a reflective boundary; while in the plane of maximum x, the boundary condition type `none` will be applied:

```
define_boundary_conditions xmin=reflective xmax=none
```

NOTE Currently:

- When using a level set–based model, the boundary conditions are set automatically to `none` in the planes in which a user-defined tilt breaks the symmetry. When using the PMC method, periodic boundary conditions are applied automatically to the planes in which a user-defined tilt breaks the symmetry (see Structure Tilt on page 18 and Boundary Conditions on page 19).
- The structure referred to by the `structure` parameter must be already defined before the `define_boundary_conditions` command is called.
- The `define_boundary_conditions` command affects not only flux computation, but also surface evolution. Therefore, this command is also relevant for non-flux models.
- The `define_boundary_conditions` command has no effect on the built-in deposition model `crystal` and the built-in etch model `crystal`. Boundary conditions are always reflective when using those models.
- The boundary conditions specified by the command `define_boundary_conditions` have no effect on the indirect flux computations for 2D structures.
- Only `reflective` boundary conditions are supported when using the `electrodeposition`, `spin_on`, or `wet` built-in deposition model.
- The boundary conditions specified by the command `define_boundary_conditions` are ignored when using RFM models that use the RFM function `pad_pressure()`, and periodic boundary conditions are always applied.
- When using the PMC method, boundary conditions of type `none` are not available.
- You can specify periodic boundary conditions only for the PMC method.
- The `x` parameter cannot be simultaneously specified with either the `xmin` parameter or the `xmax` parameter. The `y` parameter cannot be simultaneously specified with either the `ymin` parameter or the `ymax` parameter.

define_deposit_machine

The `define_deposit_machine` command defines a new machine for a deposition process. A deposition model can be specified with the parameter `model`. Parameters for different models are defined as follows.

Crystal orientation–dependent deposition:

```
define_deposit_machine material=<c> model=crystal rate_100=<n> rate_110=<n> \
   rate_111=<n> [name=<c>] [selective_materials=<l>]
```

Electrodeposition:

```
define_deposit_machine acc_adsorption_rate=<n> acc_bulk_concentration=<n> \
   bulk_distance=<n> depo_bulk_concentration=<n> \
   exchange_current_density=<n> \
   exchange_current_density_acc=<n> exchange_current_density_inh=<n> \
   inh_adsorption_rate=<n> inh_bulk_concentration=<n> inh_diffusivity=<n> \
   inh_displacement_rate=<n> \
   material=<c> model=electrodeposition overpotential=<n> temperature=<n> \
   [acc_alpha=<n>] [acc_diffusivity=<n>] \
   [acc_saturation_surface_concentration=<n>] [alpha=<n>] \
   [curvature_scaling=<n>] [depo_diffusivity=<n>] [depo_molar_volume=<n>] \
   [depo_reference_concentration=<n>] [duty_cycle=<n>] \
   [inh_alpha=<n>] [inh_saturation_surface_concentration=<n>] [name=<c>] \
   [overpotential_off=<n>] [z=<n>] [z_alpha=<n>]
```

Electroplating deposition:

```
define_deposit_machine delta=<n> material=<c> model=electroplating rate=<n> \
   [name=<c>]
```

High-density plasma deposition:

```
define_deposit_machine anisotropy=<n> (exponent=<n> | iad=<c>) material=<c> \
   model=hdp rate=<n> redeposition=<n> s1=<n> s2=<n> sputter_rate=<n> \
   sticking=<n> [name=<c>] [rotation=<n>] [tilt=<n>]
```

High-density plasma 2 deposition:

```
define_deposit_machine anisotropy=<n> (exponent=<n> | iad=<c>) material=<c> \
   model=hdp2 rate=<n> reflection=<n> redeposition=<n> s1=<n> s2=<n> \
   sputter_exponent=<n> sputter_rate=<n> sputter_type=<c> sticking=<n> \
   [name=<c>] [rotation=<n>] [tilt=<n>]
```

Low-pressure chemical vapor deposition:

```
define_deposit_machine material=<c> model=lpcvd rate=<n> sticking=<n> \
   [name=<c>] [rotation=<n>] [tilt=<n>]
```

Plasma-enhanced chemical vapor deposition:

```
define_deposit_machine anisotropy=<n> (exponent=<n> | iad=<c>) material=<c> \
   model=pecvd rate=<n> sticking=<n> [name=<c>] [rotation=<n>] [tilt=<n>]
```

Physical vapor deposition:

```
define_deposit_machine (exponent=<n> | iad=<c>) material=<c> model=pvd \
   rate=<n> [name=<c>] [rotation=<n>] [tilt=<n>]
```

Simple deposition:

```
define_deposit_machine anisotropy=<n> curvature=<n> material=<c> \
   model=simple rate=<n> [name=<c>] [rotation=<n>] [tilt=<n>]
```

Spin-on-glass deposition:

```
define_deposit_machine angular_position=<n> angular_velocity=<n> density=<n> \
   evaporation_rate=<n> initial_thickness=<n> material=<c> model=spin_on \
   radial_distance=<n> surface_tension=<n> viscosity=<n> [name=<c>]
```

RFM model:

```
define_deposit_machine material=<c> model=<c> [iad=<c>] [maximum_error=<n>] \
   [name=<c>] [reflection=<c>] [rotation=<n> | <c>] [tilt=<n>] [yield=<c>] ...
```

Table 23 Parameters of define_deposit_machine command

Parameter	Type	Description	Default value [Possible values]	Unit
acc_adsorption_ rate	number	The rate constant of the accelerator adsorption reaction.	none [0, ∞ [	µm/min
acc_alpha	number	The transfer coefficient for a surface fully covered by accelerators.	0.4 (from [3]) [0,1]	none
acc_bulk_ concentration	number	The bulk concentration of accelerators.	none [0, ∞ [	mol/cm^3
acc_diffusivity	number	The diffusivity of accelerators. If this parameter is omitted, the concentration of accelerators will be assumed to equal the value provided by the parameter acc_bulk_concentration everywhere.	none]0, ∞ [	cm^2/s

Table 23 Parameters of define_deposit_machine command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
acc_saturation_ surface_ concentration	number	The accelerator surface concentration of a surface fully covered by accelerators.	8e-10 (from [3])]0, ∞[	mol/cm^2
alpha	number	The transfer coefficient for a clean surface, that is, for a surface where no additives are adsorbed.	0.5 (from [3]) [0, 1]	none
angular_position	number	The angle between the plane that contains the yz plane of the wafer coordinate system and the plane that contains the z-axis of the wafer coordinate system and the center of the structure on which the process occurs.	none [-180, 180]	degree
angular_velocity	number	The angular velocity of the wafer.	none [0, ∞[	rpm
anisotropy	number	Anisotropy coefficient.	none [0, 1]	none
bulk_distance	number	The distance from the topmost point of the exposed surface of the plane where the species concentrations are assumed to equal their bulk values.	none]0, ∞[	µm
curvature	number	Curvature coefficient.	none [0, 0.1]	µm
curvature_scaling	number	The factor by which the surface mean curvature is scaled to update the additive surface coverage.	1 [0, ∞[	none
delta	number	The gradient of the accelerator surface concentration for the electrodeposition model.	none [0, ∞[	μm^{-1}
density	number	The density of the film to be deposited.	none]0, ∞[	g/cm^3
depo_bulk_ concentration	number	The bulk concentration of the deposited material.	none]0, ∞[	mol/cm^3
depo_diffusivity	number	The diffusivity of the deposited material. If this parameter is omitted, the concentration of the material being deposited is assumed to equal its bulk value.	none]0, ∞[	cm^2/s
depo_molar_volume	number	The molar volume of the deposited material.	7.1e-6 (from [4]) [0, ∞[	m^3/mol

Table 23 Parameters of define_deposit_machine command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
depo_reference_concentration	number	The concentration at which exchange current densities are measured. If this parameter is omitted, the value of the parameter depo_bulk_concentration is assumed as the reference concentration.	none]0, ∞ [	mol/cm^3
duty_cycle	number	The duty cycle of the overpotential, that is, a fraction of a pulse period. The overpotential takes the value given by the overpotential parameter. In the remaining time of a pulse period, the overpotential is assumed to equal the value given by the overpotential_off parameter.	1 (0,1]	none
evaporation_rate	number	The evaporation rate of the film to be deposited.	none [0, ∞ [	$\mu m\ min^{-1}$
exchange_current_density	number	The exchange current density on a free surface, measured at the reference concentration set by the depo_reference_concentration parameter.	none [0, ∞ [	mA/cm^2
exchange_current_density_acc	number	The exchange current density on a surface fully covered by accelerators, measured at the reference concentration set by the depo_reference_concentration parameter.	none [0, ∞ [	mA/cm^2
exchange_current_density_inh	number	The exchange current density on a surface fully covered by inhibitors, measured at the reference concentration set by the depo_reference_concentration parameter.	none [0, ∞ [	mA/cm^2
exponent	number	Exponent used to characterize the ion angular distribution $\cos^m\theta$. The specified value must be an integer.	none [1, ∞ [	none
iad	character	The name of the IADF to be used.	none	none
inh_adsorption_rate	number	The rate constant of the inhibitor adsorption reaction.	none [0, ∞ [	µm/min
inh_alpha	number	The transfer coefficient for a surface fully covered by inhibitors.	0.5 (from [3]) [0, 1]	none
inh_bulk_concentration	number	The bulk concentration of inhibitors.	none [0, ∞ [	mol/cm^3

Table 23 Parameters of define_deposit_machine command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
inh_diffusivity	number	The diffusivity of inhibitors.	none]0, ∞[	cm^2/s
inh_displacement_rate	number	The rate constant of the reaction by which inhibitors are displaced by accelerators.	none [0, ∞[	µm/min
inh_saturation_surface_concentration	number	The inhibitor surface concentration of a surface fully covered by inhibitors.	6e-11 (from [3])]0, ∞[	mol/cm^2
initial_thickness	number	The thickness of the film over the substrate when the spin-on simulation starts.	none]0, ∞[	µm
material	character	Sets the material to be deposited.	none	none
maximum_error	number	Sets the maximum tolerated equivalence error for all species of the model. This parameter can be used only if rotation=continuous. See Continuously Rotating and Tilted Structure Modeling on page 40.	0.5 [0, 0.5]	none
model	character	Sets the model used to represent the physical machine.	none	none
name	character	Sets the name of the machine.	default_machine	none
overpotential	number	The overpotential of the electroplating cell during the on-cycle of the overpotential pulse.	none]−∞, 0]	V
overpotential_off	number	The overpotential of the electroplating cell during the off-cycle of the overpotential pulse. This parameter can be specified only when duty_cycle is less than 1.	0]−∞, 0]	V
radial_distance	number	The distance of the center of the feature from the axis of rotation of the wafer.	none [0, ∞[	µm
rate	number	Sets the deposition rate.	none	$\mu m\ min^{-1}$
rate_100	number	The deposition rate for the <100> direction.	none [0, ∞[	$\mu m\ min^{-1}$
rate_110	number	The deposition rate for the <110> direction.	none [0, ∞[	$\mu m\ min^{-1}$
rate_111	number	The deposition rate for the <111> direction.	none [0, ∞[	$\mu m\ min^{-1}$

Table 23 Parameters of define_deposit_machine command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
redeposition	number	Ratio of the amount of material redeposited to the amount of material sputtered from the surface.	none [0, 1]	none
reflection	number/ character	For built-in models, it specifies the reflection parameter used to evaluate the reflection probability (parameter k in Eq. 11, p. 35). For RFM models, it specifies the name of the reflection function to use.	none [0, 1] for built-in models	none
rotation	number/ character	Either the rotation angle in the wafer coordinate system or whether the wafer rotates continuously.	0 [-180, 180] / [continuous]	degree / none
s1	number	The first sputter coefficient.	none	none
s2	number	The second sputter coefficient.	none	none
selective_ materials	list	A list of materials on which the deposition can occur. If this list is empty, deposition occurs on all materials of the structure.	empty	none
sputter_exponent	number	Exponent used to characterize the angular distribution $\cos^m\theta$ of the sputtered material. The specified value must be an integer.	none [1, ∞[	none
sputter_rate	number	Sets the sputtering rate. Note that the net deposition rate is equal to rate – sputter_rate.	none [0, 2]	$\mu m\ min^{-1}$
sputter_type	character	Sets the angular distribution type of the sputtered material.	none [diffuse, reflective]	none
sticking	number	Sticking probability of the deposition precursors.	none [0, 1]	none
surface_tension	number	The surface tension of the film to be deposited with respect to the surrounding fluid.	none [0, ∞[	dyn/cm
temperature	number	The absolute temperature of the electroplating process.	none [0, ∞[	K
tilt	number	The tilt angle.	0 [0, 90[	degree
viscosity	number	The dynamic viscosity of the film to be deposited.	none]0, ∞[	poise
yield	character	Specifies the name of the yield function to use for the deposition machine.	none	none

Table 23 Parameters of define_deposit_machine command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
z	number	The number of electrons involved in deposition reactions.	2 [1, 2147483647]	none
z_alpha	number	The number of electrons transferred in the rate-determining elementary reaction.	1 [1, 2147483647]	none

NOTE The `rotation` and `tilt` parameters do not support the radian measurement unit and are not supported when `model=crystal`, `model=electrodeposition`, or `model=spin_on`.

NOTE There are only two possible numeric values of the parameter `rotation` when defining a machine to be used with a 2D structure. The numeric values are obtained by converting, if needed, the angles given by:

`-slice_angle` [deg] + 90°
`-slice_angle` [deg] – 90°

into the range `[-180, 180]` degrees. In the above formulas, `slice_angle` denotes the slice angle of the structure, set by the `slice_angle` parameter of the `define_structure` command or stored in the loaded TDR file. When the default value of the slice angle is used (`-90`), the possible values of `rotation` are `0` and `180` degrees, respectively, for a machine to be used with a 2D structure (see Simulation Coordinate System on page 17 and define_structure on page 139).

You can use `rotation=continuous` when defining a machine to be used with a 2D structure that uses a rate formula module (RFM) model supporting continuous rotation, independently of the value of the slice angle (see Continuously Rotating and Tilted Structure Modeling on page 40, Data Available for Rate Calculation on page 225, and Chapter 9 on page 243).

Examples

This command defines a machine with the name `simpledepo` for a silicon deposition process that uses the model `simple`. It sets the anisotropy coefficient to 0.5, the curvature coefficient to 0.05 µm , and the deposition rate to 1 µm/minute :

```
define_deposit_machine anisotropy=0.5 curvature=0.05 material=Silicon \
   model=simple name=simpledepo rate=1
```

The wafer plane tilt can be defined by using the tilt and rotation angles as described in Structure Tilt on page 18. The same example with a tilted structure is:

```
define_deposit_machine anisotropy=0.5 curvature=0.05 material=Silicon \
   model=simple name=simpledepo rate=1 rotation=20 tilt=45
```

define_etch_machine

The `define_etch_machine` command defines a new machine for an etching model, or a simultaneous etching and deposition model, or a deposition reaction model. The parameters defined with the `define_etch_machine` command are material independent except when setting `model=crystal`.

The material-dependent parameters are defined with the command `add_material`. When setting `model=crystal`, the specified etching rates refer to the single etchable material. The `add_material` command cannot be used with a machine having `model=crystal`.

Parameters for different models are defined as follows.

Crystal orientation–dependent etching:

```
define_etch_machine etchable_material=<c> model=crystal rate_100=<n> \
   rate_110=<n> rate_111=<n> [name=<c>]
```

Dry etching:

```
define_etch_machine deposit_material=<c> model=dry rate=<n> sticking=<n> \
   [name=<c>] [rotation=<n>] [tilt=<n>]
```

Simultaneous etching and deposition:

```
define_etch_machine (exponent=<n> | iad=<c>) deposit_material=<c> \
   model=etchdepo rate=<n> sticking=<n> [name=<c>] [rotation=<n>] [tilt=<n>]
```

Simultaneous etching and deposition 2:

```
define_etch_machine (exponent=<n> | iad=<c>) deposit_material=<c> \
   model=etchdepo2 rate=<n> sticking=<n> [name=<c>] [rotation=<n>] [tilt=<n>]
```

High-density plasma etching:

```
define_etch_machine (exponent=<n> | iad=<c>) model=hdp [name=<c>] \
   [rotation=<n>] [tilt=<n>]
```

High-density plasma 2 etching:

```
define_etch_machine (exponent=<n> | iad=<c>) model=hdp2 [name=<c>] \
   [rotation=<n>] [tilt=<n>]
```

Ion-enhanced etching:

```
define_etch_machine model=ion_enhanced (exponent=<n> | iad=<c>) [name=<c>] \
   [rotation=<n>] [tilt=<n>]
```

Ion-milling:

```
define_etch_machine model=ionmill [name=<c>] [rotation=<n>] [tilt=<n>]
```

Reactive ion etching:

```
define_etch_machine (exponent=<n> | iad=<c>) model=rie [name=<c>] \
   [rotation=<n>] [tilt=<n>]
```

Reactive ion etching 2:

```
define_etch_machine (exponent=<n> | iad=<c>) model=rie2 [name=<c>] \
   [rotation=<n>] [tilt=<n>]
```

Simple etching:

```
define_etch_machine model=simple [name=<c>] [rotation=<n>] [tilt=<n>]
```

Wet etching:

```
define_etch_machine model=wet diffusivity=<n> source_distance=<n> \
   [name=<c>] [source_concentration=<n>]
```

RFM model:

```
define_etch_machine model=<c> [applied_pressure=<n>] [deposit_material=<c>] \
   [iad=<c>] [maximum_error=<n>] [name=<c>] [pad_poisson_ratio=<n>] \
   [pad_roughness=<n>] [pad_young_modulus=<n>] [reflection=<c>] \
   [rotation=<n> | <c>] [tilt=<n>] [yield=<c>] ...
```

Reaction model:

```
define_etch_machine model=<c> species_distribution=<c> [name=<c>] \
   [rotation=<n> | <c>] [species_properties=<c>] [tilt=<n>] [yield=<c>]
```

Table 24 Parameters of define_etch_machine command

Parameter	Type	Description	Default value [Possible values]	Unit
applied_pressure	number	The external pressure applied to the pad. It is mandatory for RFM models using the RFM function pad_pressure().	none [0, ∞[	Pa
deposit_material	character	Sets the deposition material for simultaneous etching and deposition models.	none	none
diffusivity	number	The diffusivity of the etchant species.	none]0, ∞[	cm^2/s
etchable_material	character	The material to be etched.	none	none

Table 24 Parameters of define_etch_machine command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
exponent	number	Exponent used to characterize the ion angular distribution $\cos^m\theta$. The specified value must be an integer.	none [1, ∞ [	none
iad	character	The name of the IADF to be used.	none	none
maximum_error	number	Sets the maximum tolerated equivalence error for all species of the model. This parameter can be used only if rotation=continuous. See Continuously Rotating and Tilted Structure Modeling on page 40.	0.5 [0, 0.5]	none
model	character	Sets the model used to represent the physical machine.	none	none
name	character	Sets the name of the etch machine.	default_machine	none
pad_poisson_ratio	number	The Poisson ratio of the pad. It is mandatory for RFM models using the RFM function pad_pressure().	none]-1, 0.5]	none
pad_roughness	number	A parameter related to the standard deviation of the distribution of the pad asperities. It is mandatory for RFM models using the RFM function pad_pressure().	none]0, ∞ [	µm
pad_young_modulus	number	The Young's modulus of the pad. It is mandatory for RFM models using the RFM function pad_pressure().	none]0, ∞ [	Pa
rate	number	Sets the rate of deposition for the material that was specified with the parameter deposit_material.	none [0, ∞ [	µm min^{-1}
rate_100	number	The etching rate for the <100> direction.	none [0, ∞ [	µm min^{-1}
rate_110	number	The etching rate for the <110> direction.	none [0, ∞ [	µm min^{-1}
rate_111	number	The etching rate for the <111> direction.	none [0, ∞ [	µm min^{-1}
reflection	character	Specifies the name of the reflection function to be used.	none	none
rotation	number/ character	Either the rotation angle in the wafer coordinate system or whether the wafer rotates continuously.	0 [-180, 180] / [continuous]	degree / none

Table 24 Parameters of define_etch_machine command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
source_ concentration	number	The concentration of the etchant species on the source plane.	1]0, ∞[	mol/cm^3
source_distance	number	The distance of the plane where the etchant concentration is assumed to be fixed from the topmost point of the exposed surface.	none]0, ∞[	µm
species_ distribution	character	Specifies the name of the species distribution to be used.	none	none
species_ properties	character	Specifies the name of the species properties to be used.	none	none
sticking	number	Sets the sticking coefficient for the deposition material that was specified with the parameter deposit_material.	none [0, 1]	none
tilt	number	The tilt angle.	0 [0, 90[	degree
yield	character	Specifies the name of the yield function to be used.	none	none

NOTE The rotation and tilt parameters do not support the radian measurement unit and are not supported when model=crystal or model=wet.

NOTE There are only two possible numeric values of the parameter rotation when defining a machine to be used with a 2D structure. The numeric values are obtained by converting, if needed, the angles given by:

-slice_angle [deg] + 90°
-slice_angle [deg] – 90°

into the range [-180, 180] degrees. In the above formulas, slice_angle denotes the slice angle of the structure, set by the slice_angle parameter of the define_structure command or stored in the loaded TDR file. When the default value of the slice angle is used (-90), the possible values of rotation are 0 and 180 degrees for a machine to be used with a 2D structure (see Simulation Coordinate System on page 17 and define_structure on page 139).

You can use rotation=continuous when defining a machine to be used with a 2D structure that uses a rate formula module (RFM) model supporting continuous rotation, independently of the value of the slice angle (see Continuously Rotating and Tilted Structure Modeling on

page 40, Data Available for Rate Calculation on page 225, and Chapter 9 on page 243).

When defining a machine that uses a reaction model, `rotation=continuous` can be used only if the reaction model is energy independent and no source species of the reaction model has a unidirectional distribution.

NOTE When defining an etch machine that uses a reaction model, only two cases are supported:

- The species distributions specified for all the source species of the used reaction models are energy independent. In this case, also the yield functions required by the reaction model and its reaction probabilities must be energy independent.
- The species distributions specified for all the source species of the used reaction models are energy dependent. In this case, the yield functions required by the reaction model and its reaction probabilities can be either energy independent or energy dependent.

In all other cases, an error will be issued.

Examples

This command defines a machine with the name `etchmachine` for an etching process that uses the model `simple`:

```
define_etch_machine name=etchmachine model=simple
```

The wafer plane tilt can be defined by using the tilt and rotation angles as described in Structure Tilt on page 18.

The same example with a tilted structure is:

```
define_etch_machine name=etchmachine model=simple rotation=45 tilt=30
```

define_extraction

The `define_extraction` command allows you to define extractions that can be used during a deposition step or an etching step.

```
define_extraction name=<c> type=interface <extraction_line> \
   <extraction_pair> tcl_output_variable=<c> [description=<c>] [output=<c>]

define_extraction name=<c> type=probe property=length <extraction_line> \
   (materials=<l> | [probe_empty_space=<c>]) [description=<c>] [file=<c>] \
   [output=<c>] [tcl_output_variable=<c>]

define_extraction name=<c> type=probe property=<c> <extraction_line> \
   tcl_output_variable=<c> [description=<c>] [output=<c>]
```

Here, an extraction line specification `<extraction_line>`, as defined in the `extract` command, is used to specify the location at which the extraction is performed.

Here, an extraction pair specification `<extraction_pair>`, as defined in the `extract` command, is used to specify a pair of materials, or parts, or regions for which the extraction is performed.

Table 25 Parameters of define_extraction command

Parameter	Type	Description	Default value [Possible values]	Unit
`axis`[1]	character	The direction of one of the coordinate axes.	none `[x, y, z]`	none
`description`	character	A description of the extraction.	none	none
`direction`[1]	vector	The direction of the extraction line.	none	none
`file`	character	File name or path to a file where the extraction results will be saved.	none	none
`material1`[2]	character	The name of the material.	none	none
`material2`[2]	character	The name of the material.	none	none
`materials`	list	The materials to take into account. **NOTE** This parameter can be specified only if `property=length`.	none	none
`name`	character	The name of the extraction group to which the extraction is added.	none	none
`output`	character	A filtering parameter.	`all` `[all, first, inside, last, outside]`	none
`point`[1]	vector	A point on the extraction line.	none	µm

Table 25 Parameters of define_extraction command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
point1[1]	vector	The first point on the extraction line.	none	µm
point2[1]	vector	The second point on the extraction line.	none	µm
probe_empty_space	character	Specifies whether to probe empty space. **NOTE** This parameter can be specified only if property=length.	false [false, true]	none
property	character	A property to be associated with the returned segments.	none [length, material, region, region_part]	none
region_part1[2]	list	The names of the first region and part in an extraction pair specification.	none	none
region_part2[2]	list	The names of the second region and part in an extraction pair specification.	none	none
region1[2]	character	The name of the first region in an extraction pair specification.	none	none
region2[2]	character	The name of the second region in an extraction pair specification.	none	none
tcl_output_variable	character	The name of the Tcl variable to which the extraction results will be written.	none	none
type	character	Specifies the extraction type.	none [interface, probe]	none

1. These parameters are used to specify an extraction line (see Specification of Extraction Lines, Planes, and Pairs on page 166).
2. These parameters are used to specify an extraction pair (see Specification of Extraction Lines, Planes, and Pairs on page 166).

The extractions defined with the define_extraction command work in a similar way as those defined with the extract command except that they are used during deposition and etching. However, there are some important differences:

- Each extraction defined with the define_extraction command is added to the group of extractions specified with the parameter name.
- To use a group of extractions defined with the define_extraction command in the deposit or etch command, the name of the extraction group is specified with the extraction parameter, and the time interval at which the extractions are performed is specified with the extraction_interval parameter. See deposit on page 147 and etch on page 154.
- Each extraction defined with the define_extraction command can write its output to a Tcl variable, whose name is provided by the parameter tcl_output_variable.

- The Tcl variable into which the extraction results are written is a list of values with the following format:

  ```
  t_0 extraction_result_0 t_1 extraction_result_1 ...
  ```

  where `t_i` is the *i*-th extraction time in minutes, and `extraction_result_i` is the result that the corresponding `extract` command would produce if run on the structure at time `t_i`.
- When specifying `type=probe property=length`, the extraction output can also be written to a TDR file such that it can be visualized in Sentaurus Visual.

NOTE When using `model=pmc` in the `etch` command, if both the `extraction_interval` and `plot_interval` parameters are specified, their values must be the same.

Examples

```
define_extraction name=extr axis=z point={0.2 0.2 0} type=probe \
   property=length materials=Silicon file=output.tdr

define_extraction name=extr axis=z point={0.2 0.2 1} type=interface \
   tcl_output_variable=res

etch time=1 spacing=0.1 extraction_interval=0.1 extraction=extr

puts "res = $res"
```

In the first command, an extraction of type `probe` is added to the extraction group named `extr`. This will extract the vertical thickness of the material `Silicon` at the specified point and write the value to the specified TDR file.

In the second command, an extraction of type `interface` is added to the extraction group named `extr`. This will extract the position of an interface, and the result is assigned to the variable `res`.

The extraction group named `extr` is used every 0.1 minutes.

In the fourth line, the Tcl command `puts` is used to output the result of the interface extraction.

define_iad

The `define_iad` command defines a new ion-angular distribution function (IADF) that can be used in a flux model. IADFs can be defined by either using a tabular format, reading them from a TDR file, or specifying the exponent of an analytic distribution function (Eq. 5, p. 32).

For built-in models:

```
define_iad exponent=<n> name=<c>

define_iad file=<c> name=<c>

define_iad name=<c> table=<v> [angle_unit=<c>]
```

For RFM models:

```
define_iad energy=<n> exponent=<n> name=<c> species=<c>

define_iad energy=<n> file=<c> name=<c> species=<c>

define_iad energy=<n> name=<c> species=<c> table=<v> [angle_unit=<c>]
```

Then, the defined IADF can be referred to under the name set by the `name` parameter.

Table 26 Parameters of define_iad command

Parameter	Type	Description	Default value [Possible values]	Unit
`angle_unit`	character	The unit of the angles listed in the `table` parameter.	`deg` `[deg, rad]`	none
`energy`	number	The ion energy for which the angular distribution is defined. **NOTE** When this parameter is set to 0, the defined distribution is energy independent.	`0` `[0, ∞ [`	eV
`exponent`	number	Exponent used to characterize the ion angular distribution $\cos^m\theta$. The specified value must be an integer.	none `[1, 2147483647]`	none
`file`	character	The name of the TDR file containing an IADF.	none	none
`name`	character	The name of the IADF. If the IADF is loaded from a file, this parameter refers to the TDR geometry name, which must exist. Otherwise, the name can be chosen freely. This name can be used to reference the IADF in either a `define_deposit_machine` or a `define_etch_machine` command.	none	none

Table 26 Parameters of define_iad command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
`species`	character	The name of the flux species for which the angular distribution is defined.	none	none
`table`	vector	Tabular format of the IADF (see Examples on page 119).	none	none

The format of a table for an IADF is as follows:

- Each line consists of a pair of numbers: The first number represents the angle and the second number represents the value of the flux distribution for this angle.
- The lines are sorted in increasing order by angle.
- The first entry is for angle $0°$.
- The last entry is for angle $90°$ and the flux distribution value is zero.
- Angles must be unique.
- All flux distribution values must be nonnegative and finite.

NOTE A violation of any of these conditions will cause an error.

NOTE When using spherical coordinates, the ion flux on a flat unshadowed surface is defined as:

$$\Gamma = \int_0^{2\pi} \int_0^{\pi/2} f(\theta) \sin\theta \, d\theta \, d\phi$$

where $f(\theta)$ is the IADF. Therefore, when using a tabular IADF, the provided data must not contain the factor $\sin\theta$.

The definition of ion distributions is not part of any model. In this way, it is easy to reuse data that has been measured or simulated under certain conditions. In fact, data about many ion fluxes can be stored in a file, which is sourced into the command file when needed.

For example, the following statement sets the distribution of the ion flux `I` according to Eq. 5, p. 32 with $m = 100$:

```
define_iad name=my_iad species=I exponent=100
```

The parameter `species` states the name of the flux to which the ion distribution refers.

Sentaurus Topography 3D supports collections of ion distributions, so that data can be reused easily. For example, a collection of ion distributions named `my_iad` is defined with the commands:

```
define_iad name=my_iad species=I1 table=$table1

define_iad name=my_iad species=I2 exponent=1000

define_iad name=my_iad species=I3 table=$table3
```

In this way, the entire collection can be referred to using one name.

NOTE Currently, there is no support for energy-dependent flux integration. Therefore, only energy-independent IADFs can be used.

Examples

This command loads an IADF with the TDR geometry name `iad_1` from the file `iad.tdr`:

```
define_iad name=iad_1 file=iad.tdr
```

The following commands first define a Tcl variable called `table` that contains a very simple IADF consisting of eleven angles and the corresponding flux distribution values:

```
# Define an IADF table using a Tcl list. The first column represents angles
# in degree, and the second column represents absolute flux values.
set table {
   0.0     1.0
   2.0     0.95
   4.0     0.8
   6.0     0.6
   8.0     0.4
   10.0    0.2
   15.0    0.1
   20.0    0.05
   30.0    0.02
   40.0    0.0
   90.0    0.0
}

# Use the Tcl variable in the definition of the IADF.
define_iad name=my_iad table=$table
```

NOTE The user-defined IADFs are normalized internally so that the direct flux on a flat unshadowed surface of the species they refer to equals one.

define_litho_machine

The `define_litho_machine` command defines a new machine for a lithography process. A lithography machine and its properties are used in a lithography simulation when the name of the machine is specified with the `machine` parameter in the `litho` command.

NOTE A machine defined with the `define_litho_machine` command can be used only to process a 3D structure.

```
define_litho_machine [name=<c>]
```

Table 27 Parameters of define_litho_machine command

Parameter	Type	Description	Default value [Possible values]	Unit
`name`	character	The name of the litho machine that is defined.	`default_machine`	none

Properties are added to the machine using the `add_litho_command` command.

define_mask

The `define_mask` command defines a new mask that can be in either an etch step, using the `etch` command (see etch on page 154), or a patterning step, using the `pattern` command (see pattern on page 196).

NOTE A mask defined with the `define_mask` command can be used only to process a 3D structure.

```
define_mask domain_min=<v> domain_max=<v> file=<c> layer=<c> \
   [gds_cell=<c>] [name=<c>] [translation=<v>] [type=<c>]
```

The mask is read from a file, which can be in CIF format or GDSII format. When read in, a specific mask layer and the 2D simulation domain must be specified. The origin of the (x,y) coordinate system is shifted so that (0, 0) is on the lower-left corner of the domain. An optional translation vector can be specified to move the mask relative to the new origin.

The relationship of the parameters `domain_min`, `domain_max`, and `translation` is illustrated in Figure 17.

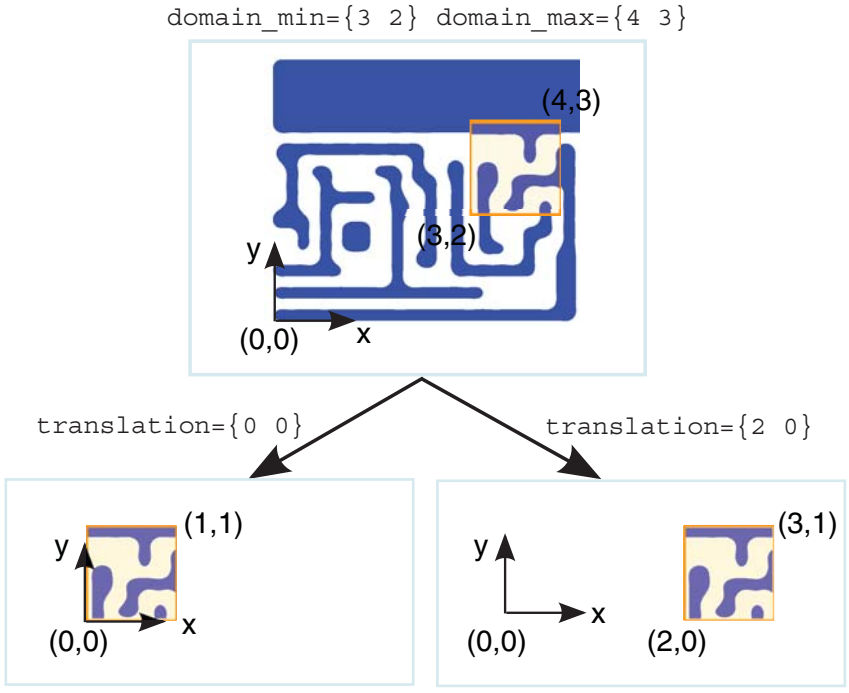


Figure 17 Illustration of the parameters domain_min, domain_max, and translation in the define_mask command

Table 28 Parameters of define_mask command

Parameter	Type	Description	Default value [Possible values]	Unit
domain_max	vector	The 2D maximum point of the mask domain.	none	µm
domain_min	vector	The 2D minimum point of the mask domain.	none	µm
file	character	The layout file from which the mask should be extracted.	none	none
gds_cell	character	The cell in a hierarchical GDSII file to be loaded. All other cells are not loaded. Only valid if file is a GDSII file.	empty string	none
layer	character	The mask layer within the layout file that is used to extract the mask.	none	none
name	character	The name of the mask.	default_mask	none
translation	vector	The 2D translation vector.	{0 0}	µm
type	character	The type of the mask: • dark inverts the polygons. • light creates a mask using the mask polygons in the file.	light [dark, light]	none

Examples

This command reads a CIF layout file mask.cif. From the layer INIT, a mask is extracted within the domain {0.0 0.0} {1.0 1.0}. The mask is given the name mask1, which can be used in a subsequent pattern command:

```
define_mask file=mask.cif name=mask1 domain_min={0.0 0.0} \
   domain_max={1.0 1.0} layer=INIT
```

This command reads the layer 15:0 in cell CELL_1 of the hierarchical GDSII file mask.gds. The domain is {0.0 0.0} {1.0 1.0}. The mask is given the name mask1, which can be used in a subsequent etch or pattern command:

```
define_mask file=mask.gds name=mask1 domain_min={0.0 0.0} \
   domain_max={1.0 1.0} gds_cell=CELL_1 layer=15:0
```

NOTE If the parameter gds_cell is specified when using a GDSII file, only the specified cell (and its subcells) will be loaded. This speeds up the mask definition process for large GDSII layouts.

NOTE When reading GDSII files that have been created using Ligament Layout Editor, an optional layer name translation file (with the same base name as the mask, ending in `.lyn`) is loaded automatically along with the mask. The layer names specified in the `.lyn` file then must be used in the `layer` parameter. If the `.lyn` file is not found, the GDSII layer names will be used.

define_model

The `define_model` command starts the definition of a new RFM model or a reaction model.

```
define_model description=<c> name=<c> [type=<c>]
```

Table 29 Parameters of define_model command

Parameter	Type	Description	Default value [Possible values]	Unit
`description`	character	A description of the model.	none	none
`name`	character	The name to be given to the model.	none	none
`type`	character	Specifies the type of the model. If omitted, the model being defined is assumed to be a reaction model.	none `[deposit, etch, etchdepo]`	none

The unique name of an RFM model is specified with the parameter `name`. It is used to reference the model in all commands for configuring a model and in the `define_deposit_machine` and `define_etch_machine` commands, for using the model in a machine.

If the parameter `type` is specified, the model being defined is assumed to be specified using rate formulas (see add_flux_properties on page 71, add_formula on page 73, add_int_parameter on page 75, add_ion_flux on page 81, and add_neutral_flux on page 87).

Three different types are supported: deposition, etching, and simultaneous etching and deposition. If the parameter `type` is omitted, the model being defined is assumed to be specified using reactions (see add_reaction on page 88 and add_source_species on page 98).

define_probability

The `define_probability` command defines a new energy- and angle-dependent probability (hereafter, referred to as the *probability function*).

You can specify the angle-dependent part of the probability function for a given energy level either as a table with the syntax:

```
define_probability energy=<n> name=<c> table=<v> [angle_unit<c>]
```

or according to the Mizuno model (see Eq. 11, p. 35) with the syntax:

```
define_probability energy=<n> mizuno_k=<n> name=<c>
```

or with an analytic expression with the syntax:

```
define_probability energy=<n> expression=<c> name=<c>
```

An analytic energy-dependent probability function can be specified with the syntax:

```
define_probability expression=<c> name=<c>
```

Table 30 Parameters of define_probability command

Parameter	Type	Description	Default value [Possible values]	Unit
`angle_unit`	character	The unit of the angles listed in the `table` parameter.	`deg` `[deg, rad]`	none
`energy`	number	The energy for which the probability function is defined.	none `[0, ∞ [`	eV
`expression`	character	The formula used to calculate the probability function.	none	none
`mizuno_k`	number	Species of the parameter k of Eq. 11, p. 35.	none `[0, 1]`	none
`name`	character	The name used to refer to the energy- and angle-dependent probability in an `add_reaction_properties` command.	none	none
`table`	vector	Tabular format of the probability function.	none	none

When using the parameter `table`, the same restrictions on the format of the `table` apply as described in define_iad on page 117, except that the value specified for 90° does not have to be zero. In addition, all of the probability function values must be in the range [0, 1].

Energy-independent probability functions can be specified by setting `energy=0`. In this case, only one specification for the given `name` is allowed.

The syntax of the supported expressions is the same as for the `define_yield` command and it is described in define_yield on page 143.

define_reflection

The `define_reflection` command specifies the properties of a new reflection function that depends on the ion species, the ion energy, and the surface material.

This reflection function can be used in RFM models that take reflection into account.

```
define_reflection energy=<n> material=<c> name=<c> species=<c> table=<v> \
   [angle_unit=<c>]

define_reflection energy=<n> material=<c> name=<c> species=<c> reflection=<n>
```

Table 31 Parameters of define_reflection command

Parameter	Type	Description	Default value [Possible values]	Unit
`angle_unit`	character	The unit of the angles listed in the `table` parameter.	`deg` `[deg, rad]`	none
`energy`	number	The energy for which the reflection function is defined.	none `[0, ∞ [`	eV
`material`	character	The name of the surface material for which the reflection function is defined.	none	none
`name`	character	The name used to reference the reflection function in a `define_deposit_machine` or `define_etch_machine` command.	none	none
`reflection`	number	Specifies the reflection parameter used to evaluate the reflection probability according to Eq. 11, p. 35.	none `[0, 1]`	none
`species`	character	The name of the flux species for which the reflection function is defined.	none	none
`table`	vector	Tabular format of the reflection function.	none	none

When the parameter `reflection` is specified, the reflection function described in Eq. 11, p. 35 is used.

When using the parameter `table`, the same restrictions on the format of the table apply as described in define_iad on page 117, except that the value specified for 90° does not have to be zero. In addition, all reflection probability values must be in the range [0, 1].

For energy-dependent reflection functions, for each combination of ion species and surface material, the properties of the reflection function for at least two different energy levels must be specified.

For energy values that have not been specified, the value of the reflection function is calculated by linearly interpolating the values of the parameter `reflection` for the analytic reflection function or the specified values for the tabular reflection functions.

Energy-independent reflection functions can be specified by setting the `energy` to zero. In this case, only one specification for each combination of ion species and surface material is allowed.

Sentaurus Topography 3D supports collections of reflection functions, as shown by the following commands, which define a collection of energy-independent reflection functions called `my_reflection` for flux `I` and several target materials:

```
define_reflection name=my_reflection species=I energy=0 material=Silicon \
   reflection=0.4

define_reflection name=my_reflection species=I energy=0 material=Oxide \
   reflection=0.1

define_reflection name=my_reflection species=I energy=0 material=Nitride \
   table=$reflection_table

define_reflection name=my_reflection species=I energy=0 material=Photoresist \
   reflection=0.9
```

NOTE Currently, there is no support for energy-dependent flux integration. Therefore, only energy-independent reflection functions can be used.

define_shape

The `define_shape` command defines a new shape for a geometric etch or deposit step.

The following 3D shapes are supported:

- Conical shape:

```
define_shape type=cylinder name=<c> center1=<v> center2=<v> radius1=<n> \
   radius2=<n> [refinement=<n>]
```

- Cuboid shape:

```
define_shape type=cube name=<c> point_max=<v> point_min=<v> \
   ( [scale_bottom=<n>] | ( [scale_bottom_x=<n>] [scale_bottom_y=<n>] ) ) \
   ( [scale_top=<n>] | ( [scale_top_x=<n>] [scale_top_y=<n>] ) )
```

- Cylindrical shape:

```
define_shape type=cylinder name=<c> center1=<v> center2=<v> radius=<n> \
   [refinement=<n>]
```

- Spherical shape:

```
define_shape type=sphere name=<c> center=<v> radius=<n> [refinement=<n>]
```

The following 2D shapes are supported:

- Rectangular shape:

```
define_shape type=rectangle point_max=<v> point_min=<v> \
   [scale_bottom=<n>] [scale_top=<n>]
```

- Circular shape:

```
define_shape type=circle center=<v> radius=<n> [refinement=<n>]
```

Table 32 Parameters of define_shape command

Parameter	Type	Description	Default value [Possible values]	Unit
`center`	vector	The center point of a sphere or circle.	none	µm
`center1`	vector	The first axis point of a cylinder.	none	µm
`center2`	vector	The second axis point of a cylinder.	none	µm
`name`	character	The unique name of the shape.	none	none
`point_max`	vector	The maximum corner point of a cube or rectangle.	none	µm
`point_min`	vector	The minimum corner point of a cube or rectangle.	none	µm

Table 32 Parameters of define_shape command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
`radius`	number	The radius of a circle, a sphere, or a cylinder base circle.	none `[0, ∞ [`	µm
`radius1`	number	The radius of the circle with the center at `center1` of a truncated cone.	none `[0, ∞ [`	µm
`radius2`	number	The radius of the circle with the center at `center2` of a truncated cone.	none `[0, ∞ [`	µm
`refinement`	number	Circular, cylindrical, or spherical refinement: • If a circle, this is the number of circle vertices. • If a cylinder, this is the number of base circle points of the cylinder. • If a sphere, this is the number of icosahedron faces: $20 \cdot 4^{\text{refinement}}$.	`2` (sphere) `20` (circle) `20` (cylinder) `[1, ∞ [`	none
`scale_bottom`	number	Scale factor for the bottom face or edge of the initial cube or rectangle. A truncated pyramid or a symmetric trapezoid is created instead of a cube or a rectangle, respectively.	`1` `[0, ∞ [`	none
`scale_bottom_x`	number	Scale factor for the two sides parallel to the x-axis of the bottom face of a cube.	`1` `[0, ∞ [`	none
`scale_bottom_y`	number	Scale factor for the two sides parallel to the y-axis of the bottom face of a cube.	`1` `[0, ∞ [`	none
`scale_top`	number	Scale factor for the top face or edge of the initial cube or rectangle. A truncated pyramid or a symmetric trapezoid is created instead of a cube or a rectangle, respectively.	`1` `[0, ∞ [`	none
`scale_top_x`	number	Scale factor for the two sides parallel to the x-axis of the top face of a cube.	`1` `[0, ∞ [`	none
`scale_top_y`	number	Scale factor for the two sides parallel to the y-axis of the top face of a cube.	`1` `[0, ∞ [`	none
`type`	character	The type of shape to be defined.	none `[circle, cube, cylinder, rectangle, sphere]`	none

Examples

This command defines a 2D circle with center {0.0 0.0}, radius 1.0 µm, and name circle_1:

```
define_shape type=circle name=circle_1 center={0.0 0.0} radius=1.0
```

This command defines a cube with bounding points {0.25 0.25 0.25},{0.75 0.75 1.0} and with the name cube_1. The cube can be used in a subsequent etching or deposition step:

```
define_shape type=cube name=cube_1 point_max={0.75 0.75 1.0} \
   point_min={0.25 0.25 0.25}
```

This command defines a cylinder with an axis of {0.0 0.0 0.0},{0.0 0.0 1.0} parallel to the z-axis, with a radius of 0.25 µm and the name cylinder_1. The cylinder can be used in a subsequent etching or deposition step:

```
define_shape type=cylinder name=cylinder_1 \
   center1={0.0 0.0 0.0} center2={0.0 0.0 1.0} radius=0.25
```

This command defines a 2D rectangle with corners {0.0 0.0} {1.0 1.0} and the name r_1:

```
define_shape type=rectangle name=r_1 point_max={1.0 1.0} point_min={0.0 0.0}
```

This command defines a sphere with a center point of {0.0 0.0 0.0} and a radius of 0.25 µm, with the name sphere_1. The sphere can be used in a subsequent etching or deposition step:

```
define_shape type=sphere name=sphere_1 center={0.0 0.0 0.0} radius=0.25
```

This command defines a pyramid with vertices at the points of coordinates {0.0 0.0 0.0}, {0.0 0.0 1.0}, {0.0 1.0 0.0}, {0.0 1.0 1.0}, and {0.5 0.5 1.0}. The pyramid can be used in a subsequent etching or deposition step:

```
define_shape type=cube name=pyramid point_min={0.0 0.0 0.0} \
   point_max={1.0 1.0 1.0} scale_top=0
```

This command defines a square truncated pyramid with vertices at the points of coordinates {0.0 0.0 0.0}, {0.0 0.0 1.0}, {0.0 1.0 0.0}, {0.0 1.0 1.0}, {0.25 0.25 1.0}, {0.75 0.25 1.0}, {0.25 0.75 1.0}, and {0.75 0.75 1.0}. The square truncated pyramid can be used in a subsequent etching or deposition step:

```
define_shape type=cube name=square_truncated_pyramid point_min={0.0 0.0 0.0} \
   point_max={1.0 1.0 1.0} scale_top=0.5
```

This command defines a nonsquare truncated pyramid with vertices at the points of coordinates {0.0 0.0 0.0}, {0.0 0.0 1.0}, {0.0 1.0 0.0}, {0.0 1.0 1.0}, {0.25 0.375 1.0}, {0.75 0.375 1.0}, {0.25 0.625 1.0}, and {0.75 0.625 1.0}. The nonsquare truncated pyramid can be used in a subsequent etching or deposition step:

```
define_shape type=cube name=truncated_pyramid point_min={0.0 0.0 0.0} \
   point_max={1.0 1.0 1.0} scale_top_x=0.5 scale_top_y=0.25
```

define_species_distribution

The `define_species_distribution` command defines a new energy and angular distribution function that can be used in a reaction model. As a special case, it allows you to define angular distributions that are energy independent.

A distribution function whose angular-dependent part is described by Eq. 5, p. 32 can be defined with the following command:

```
define_species_distribution exponent=<n> flux=<n> name=<c> species=<c> \
   [energy_max=<n> energy_min=<n>]
```

Such a distribution function is energy independent only if parameters `energy_min` and `energy_max` are omitted. When parameters `energy_min` and `energy_max` are specified, the energy distribution is assumed to be uniform over the energy window specified by the values of these two parameters.

A distribution function that specifies particles traveling along the vertical direction can be defined with the following command:

```
define_species_distribution flux=<n> name=<c> species=<c> unidirectional=<c> \
   [energy_max=<n> energy_min=<n>]
```

Such a distribution function is energy independent only if parameters `energy_min` and `energy_max` are omitted. When parameters `energy_min` and `energy_max` are specified, the energy distribution is assumed to be uniform over the energy window specified by the values of these two parameters.

A distribution function whose angular-dependent part is described by tabular data can be defined with the following command:

```
define_species_distribution flux=<n> name=<c> species=<c> table=<v> \
   [angle_unit=<c>] [energy_max=<n> energy_min=<n>]
```

Such a distribution function is energy independent only if parameters `energy_min` and `energy_max` are omitted. When parameters `energy_min` and `energy_max` are specified, the energy distribution is assumed to be uniform over the energy window specified by the values of these two parameters.

An energy and angular distribution computed by the Plasma Chemistry Monte Carlo (PCMC) module of the HPEM plasma simulator[1] [5] can be imported with the following command:

```
define_species_distribution hpem_file=<c> name=<c> species=<c> \
   [energy_max=<n> energy_min=<n> flux=<n> hpem_material_index=<n> \
    hpem_species=<c>]
```

An energy-independent angular distribution can be defined by integrating an energy and angular distribution computed by HPEM with the following command:

```
define_species_distribution hpem_file=<c> integration_energy_max=<n> \
   integration_energy_min=<n> name=<c> species=<c> \
   [flux=<n> hpem_material_index=<n> hpem_species=<c>]
```

Table 33 Parameters of define_species_distribution command

Parameter	Type	Description	Default value [Possible values]	Unit
angle_unit	character	The unit of the angles listed in the table parameter.	deg [deg, rad]	none
energy_max	number	When parameter hpem_file is specified, energy_max sets the maximum energy to take into account when reading the specified HPEM output file. Otherwise, it sets the maximum energy of the specified uniform energy distribution.	∞ [0, ∞ [	eV
energy_min	number	When parameter hpem_file is specified, energy_min sets the minimum energy to take into account when reading the specified HPEM output file. Otherwise, it sets the minimum energy of the specified uniform energy distribution.	0 [0, ∞ [	eV
exponent	number	The exponent of the angular distribution $\cos^m\theta$ of the species.	none [1, 2147483647]	none
flux	number	The number of molecules or atoms of the specified species entering the top face of the simulation domain per unit time and area. If provided when reading data from an HPEM file, it will overwrite the flux specified by the HPEM file.	none [0, ∞ [	mol/s/m^2

1. HPEM is a plasma simulation tool developed by Prof. Kushner and distributed by Quantemol Ltd (http://www.quantemol.com). For further information, contact info@quantemol.com.

Table 33 Parameters of define_species_distribution command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
hpem_file	character	Sets the name of the file containing the output of the PCMC module of the HPEM simulator, that is, the energy and angular distribution functions.	none	none
hpem_material_index	number	Sets the index of the HPEM material where the HPEM index species was measured.	none	none
hpem_species	character	Sets the name of the HPEM species that the angular distribution functions must read. If omitted, the HPEM species with the name provided by the species parameter will be used.	none	none
integration_energy_max	number	Sets the maximum energy to take into account when integrating an energy and angular distribution function.	none [0, ∞ [	eV
integration_energy_min	number	Sets the minimum energy to take into account when integrating an energy and angular distribution function.	none [0, ∞ [	eV
name	character	The name of the angular distribution function.	none	none
species	character	The name of the species for which the distribution is defined.	none	none
table	vector	Tabular format of the distribution. The same format is used as for the table parameter of the define_iad command (see define_iad on page 117).	none	none
unidirectional	character	Specifies whether all the molecules or atoms are distributed along the vertical direction.	false [false, true]	none

NOTE The format of a table specifying a distribution must be the same as that required by the define_iad command (see define_iad on page 117).

To ease the reuse of measured or simulated data, the definition of species distributions is not part of any model. Species distributions are bound to reaction models when a machine using a reaction model is defined (see define_etch_machine on page 109).

For example, the following command specifies that atoms of species Ar travel in the reactor along the vertical direction and are energy independent. The flux of Ar atoms entering the reactor is set to 10^{-4} mol/s/m^2:

```
define_species_distribution flux=1e-4 name=my_dist species=Ar \
   unidirectional=true
```

An energy-independent angular distribution according to Eq. 5, p. 32 with $m = 1000$ can be set to species `Ar` with the following command:

```
define_species_distribution exponent=1000 flux=1e-4 name=my_dist species=Ar
```

An energy and angular distribution describing molecules of species `F2` moving isotropically in the reactor and with energy uniformly distributed between 10 eV and 20 eV can be specified with the following command:

```
define_species_distribution exponent=1 flux=1e-4 name=my_dist species=F2 \
   energy_min=10 energy_max=20
```

An energy and angular distribution computed by the PCMC module of the HPEM simulator for species `AR^` and stored in a file named `pcmc.prof` can be imported with the following command:

```
define_species_distribution name=my_dist species=AR^ hpem_file=pcmc.prof
```

Since the `flux` parameter is not specified, the flux of species `AR^` used in the simulation of a reaction model will be determined from the specified HPEM output.

When the `flux` parameter is specified, its value is used as the flux of the species distribution read from the HPEM output. For example, the following command defines an energy and angular distribution for species `AR^` as computed by the HPEM simulator, but with the flux equal to 10^{-3} mol/s/m^2:

```
define_species_distribution name=my_dist species=AR^ flux=1e-3 \
   hpem_file=pcmc.prof
```

When parameters `energy_min` and `energy_max` are not specified, all the energy levels available in the HPEM output are imported. It is possible to limit the imported energy levels to those included in the range specified by parameters `energy_min` and `energy_max`. For example, the following command imports only the energy levels between 100 eV and 120 eV:

```
define_species_distribution name=my_dist species=AR^ hpem_file=pcmc.prof \
   energy_min=100 energy_max=120
```

The data computed by the HPEM simulator for one species can be used to define the distributions of multiple reaction model species. For example, the following commands:

```
define_species_distribution name=my_dist species=AR_high_energy \
   hpem_species=AR^ hpem_file=pcmc.prof energy_min=150 energy_max=200

define_species_distribution name=my_dist species=AR_low_energy \
   hpem_species=AR^ hpem_file=pcmc.prof energy_min=50 energy_max=100
```

define the energy and angular distributions of the reaction model species `AR_high_energy` and `AR_low_energy` using the data computed by the HPEM simulator for the HPEM species named `AR^`.

It is also possible to define an energy-independent angular distribution from an energy- and angular-dependent distribution computed by the HPEM simulator. For example, the following command:

```
define_species_distribution name=my_dist species=AR^ hpem_file=pcmc.prof \
   integration_energy_min=150 integration_energy_max=200
```

defines an energy-independent angular distribution for species `AR^` obtained as the integral over the energy of the data specified for this species in the `pcmc.prof` file. The integration range is specified by the `integration_energy_min` and `integration_energy_max` parameters.

define_species_properties

The `define_species_properties` command defines the properties of a species used in a reaction model.

```
define_species_properties default_event=<c> name=<c> species=<c>
```

Table 34 Parameters of define_species_properties command

Parameter	Type	Description	Default value [Possible values]	Unit
`default_event`	character	Specifies what happens to a molecule or an atom of the specified species when it interacts with the structure and no reaction occurs.	`reemit` `[discard, reemit]`	none
`name`	character	The name used to refer to the species properties in a `define_etch_machine` command.	none	none
`species`	character	The name of the species for which the properties are defined.	none	none

It is not mandatory to issue the `define_species_properties` command for all the species involved in the simulation. The species whose properties are not set with the `define_species_properties` command will be reemitted when they interact with the structure and no reaction occurs.

define_structure

The `define_structure` command defines a new structure. The structure can be loaded from a TDR boundary file, or it can be a cuboid (3D) or a rectangle (2D). For a PMC simulation, the structure can also be loaded from a PMC file or from a TDR file containing a 3D GC structure.

NOTE Multiple structures can be defined in the same input file, even of different dimensions. This allows you to mix 2D and 3D simulations in the same input file (see Simulating Process Steps on 2D and 3D Structures on page 14).

To load a structure from a TDR file containing a boundary:

```
define_structure file=<c> [conformalize=<c>] [name=<c>] [slice_angle=<n>] \
   [tdr_geometry=<c>]
```

To load a structure from a TDR file containing a 3D GC structure:

```
define_structure file=<c> [name=<c>] [slice_angle=<n>] [tdr_geometry=<c>]
```

To create a 3D cuboid or 2D rectangular structure from the beginning:

```
define_structure material=<c> point_max=<v> point_min=<v> \
   [flat_orientation=<v>] [name=<c>] [region=<c>] [slice_angle=<n>] \
   [vertical_orientation=<v>]
```

To load a PMC structure from a PMC file:

```
define_structure pmc_file=<c> [initial_structure_name=<c>] [name=<c>] \
   [slice_angle=<n>]
```

Table 35 Parameters of define_structure command

Parameter	Type	Description	Default value [Possible values]	Unit
`conformalize`	character	If `true`, an operation is performed to make input boundaries conformal.	`false` `[false, true]`	none
`file`	character	Path of the TDR boundary file containing the structure.	none	none
`flat_orientation`	vector	Specifies the Miller index of the direction perpendicular to the wafer flat (y-axis of the wafer coordinate system) for the newly created region.	`{1 1 0}`	none

Table 35 Parameters of define_structure command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
initial_ structure_name	character	Specifies a new name for the initial structure loaded from a PMC file and that can be used as the body of Boolean operations.	none	none
material	character	Sets the material of the initial cube.	none	none
name	character	Sets the name of the structure.	none	none
pmc_file	character	File name or path to a PMC file from which to load the initial data structure.	none	none
point_max	vector	Maximum corner point of the initial cube.	none	µm
point_min	vector	Minimum corner point of the initial cube.	none	µm
region	character	Name of the created region if the structure is not loaded from a file.	none	none
slice_angle	number	Specifies the rotation of the x-axis of the simulation coordinate system with respect to the y-axis of the wafer coordinate system. **NOTE** This parameter does not support the radian measurement unit.	-90 [-180, 180]	degree
tdr_geometry	character	Specifies the name of a TDR geometry in the TDR file. Only this specified geometry is loaded. The name must exist in the file specified by file.	none	none
vertical_ orientation	vector	Specifies the Miller index of the direction perpendicular to the wafer surface (z-axis of the wafer coordinate system) for the newly created region.	{0 0 1}	none

NOTE To create a rectangle, the vector values of point_min and point_max must have two components. To create a cuboid, three components must be specified.

NOTE The Miller indices for a plane are usually written as (hkl), and the set of equivalent planes is written as {hkl}. A crystal direction is usually written as [hkl], and the set of equivalent directions is written as <hkl>. The value assigned to the parameters flat_orientation and vertical_orientation is a Tcl list of indices representing a crystal direction. The braces in the specified value are required to form a Tcl list and do not indicate a set of equivalent planes.

Examples

Example of conformal input:

```
define_structure file=input.tdr
```

This command imports a structure from the file `input.tdr`. The input is assumed to have conformal boundaries.

NOTE If multiple geometries are present and the `tdr_geometry` parameter is not specified, the last valid TDR boundary in the TDR file is selected automatically.

Example of loading a specific TDR geometry from a file:

```
define_structure file=input.tdr tdr_geometry="geometry_0"
```

This command imports the TDR geometry `geometry_0` from the file `input.tdr`, and it ignores all other geometries in the file. The input is assumed to have conformal boundaries.

Example of nonconformal input:

```
define_structure file=input.tdr conformalize=true name=conformalized_input
```

This command imports a structure from the file `input.tdr`. An operation on the structure is performed to create conformal boundaries between regions. The name of the structure is `conformalized_input`.

Example of defining a 3D cuboid structure from the beginning:

```
define_structure material=Silicon point_max={1 1 1} point_min={0 0 0} \
   region="substrate"
```

This command creates an initial cuboid structure, with silicon as the material and `substrate` as the name of the region.

Example of defining a 2D rectangular structure from the beginning:

```
define_structure material=Silicon point_max={1 1} point_min={0 0}
```

This command creates an initial 2D rectangular structure, with silicon as the material.

Loading a PMC Structure

A PMC structure is loaded from a PMC file with the `define_structure` command when the parameter `pmc_file` is used.

When the PMC file contains an initial structure, it will be loaded. The parameter `initial_structure_name` can be used to rename the initial structure to the specified name. See Saving PMC Structures on page 202.

NOTE A PMC file is not a TDR file and cannot be visualized with Sentaurus Visual. It can only be loaded in to Sentaurus Topography 3D.

define_yield

The `define_yield` command specifies the properties of a new yield function that depends on the species of the incoming particles, its energy, and the target surface material. When defining a yield function for a machine using an RFM model, the value of the `species` parameter must be the name of an ion flux of that model.

This yield function can be used in RFM or reaction models that take sputtering into account.

```
define_yield energy=<n> material=<c> name=<c> normalized=<c> species=<c> \
   table=<v> [angle_unit=<c>]

define_yield energy=<n> material=<c> name=<c> s1=<n> s2=<n> species=<c> \
   [yield_at_zero=<n>]

define_yield energy=<n> material=<c> name=<c> species=<c> sputtering=false

define_yield energy=<n> material=<c> name=<c> species=<c> theta_max=<n> \
   yield_max=<n> [yield_at_zero=<n>]

define_yield expression=<c> material=<c> name=<c> [energy=<n>]
```

Table 36 Parameters of define_yield command

Parameter	Type	Description	Default value [Possible values]	Unit
`angle_unit`	character	The unit of the angles listed in the `table` parameter.	`deg` `[deg, rad]`	none
`energy`	number	The energy for which the yield function is defined.	none `[0, ∞ [`	eV
`expression`	character	The formula used to calculate the yield function.	none	none
`material`	character	The name of the surface material for which the yield function is defined.	none	none
`name`	character	The name used to reference the yield function in a `define_deposit_machine` or `define_etch_machine` command.	none	none
`normalized`	character	Specifies whether the value of the yield function at 0° must be 1 or whether it can be any nonnegative value.	none `[false, true]`	none
`s1`	number	The first sputter coefficient.	none	none
`s2`	number	The second sputter coefficient.	none	none
`species`	character	The name of the flux species for which the yield function is defined.	none	none

Table 36 Parameters of define_yield command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
`sputtering`	character	Completely disables sputtering for the given species and material pair.	`[false]`	none
`table`	vector	Tabular format of the yield function.	none	none
`theta_max`	number	The angle at which the yield function has its maximum.	none `[0, 90]`	degree

`yield_at_zero`	number	The value of the yield function at normal incidence.	`1`
			`[0, ∞ [`
			(when `s1` and `s2` are used)
			`(0, ∞ [`
			(when `theta_max` and `yield_max` are used)
				none

`yield_max`	number	The maximum value of the yield function.	none `[1, ∞ [`	none

For energy-dependent yield functions, for each combination of species and material, the properties of the yield function for at least two different energy levels must be specified.

The value of the yield function for energies below the specified minimum energy is zero. Therefore, the minimum-specified energy is the threshold energy for the yield function.

For other energy values that have not been specified, the value of the yield function is calculated by linearly interpolating the parameter values for analytic yield functions or the specified values for tabular yield functions.

Energy-independent yield functions can be specified by setting the energy to zero. In this case, only one specification for each combination of species and material is allowed.

When the parameters `s1` and `s2` are specified, the yield function is defined as:

$$\gamma(\theta) = \gamma_0(s_1\cos\theta + s_2\cos^2\theta + (1 - s_1 - s_2)\cos^4\theta) \quad (45)$$

where γ_0 denotes the value of the `yield_at_zero` parameter.

When the parameters `theta_max` and `yield_max` are specified, the yield function is defined as [6]:

$$\gamma(\theta) = \gamma_0\cos^{-f}\theta \exp(-s(1/\cos\theta - 1)) \quad (46)$$

with:

$$f = -\ln(\gamma_{max}/\gamma_0)/(\ln(\cos(\theta_{max})) + 1 - \cos\theta_{max})$$
$$s = f\cos\theta_{max} \quad (47)$$

The values of the yield function also can be specified in tabular form similar to the ion-angular distribution (IAD) where the first column represents angles and the second column is the value of the yield function.

The restrictions on the format of the table are the same as for the `define_iad` command (see define_iad on page 117). In addition, if `normalized=true`, the value of the yield function for $0°$ must be exactly one.

When the parameter `expression` is specified, the yield function is defined from the user-specified expression and the `energy` parameter is optional.

There are three cases:

- The parameter `energy` is not specified. The specified expression defines an energy-dependent yield function that holds true for all the energy levels.
- The parameter `energy` is set to `0`. The specified expression defines an energy-independent yield function.
- The parameter `energy` is set to a positive value. The specified expression defines the angle-dependent part of the yield function for the given energy level.

The syntax for expressions consists of the following components:

- Numeric constants containing the decimal point.
- Arithmetic operators: +, -, *, /
- Mathematical constants:
  - M_E (denotes the value of the Euler number)
  - M_PI (denotes the value of π)
- Mathematical functions: fabs(x), sin(x), cos(x), tan(x), asin(x), acos(x), atan(x), atan2(x, y), sinh(x), cosh(x), tanh(x), exp(x), log(x), log10(x), pow(x, y), sqrt(x), ceil(x), floor(x)

  The argument of the trigonometric functions is supposed to be given in radians.
- The conditional expression `"condition ? x : y"`, returning `x` if `condition` is true, and `y` otherwise.
- Parentheses.

- The string `theta`. This string denotes the angle in radians between the normal to the surface at the collision point and the direction of the incoming particle.
- When the `energy` parameter is omitted, string `E`. This string denotes the energy in electron volts of the incoming particle.

The arguments `x` and `y` to the above functions are themselves expressions.

A condition is an expression that, in addition, can contain the relational operators (`"<"`, `"<="`, `">"`, `">="`, `"=="`, `"!="`).

For example, the following command defines an energy-independent yield function for `Silicon` sputtering from species `Ar` that is proportional to the cosine of the angle between the direction of the impinging particle and the surface normal at the collision point:

```
define_yield name=my_yield species=Ar material=Silicon energy=0 \
   expression="0.3*cos(theta)"
```

The following command defines an energy-dependent yield function for `Silicon` sputtering from species `Ar` that is equal to zero when the energy of the impinging particle is smaller than 85 eV and that is proportional to the cosine of the angle between the direction of the impinging particle and the surface normal at the collision point, otherwise:

```
define_yield name=my_yield species=Ar material=Silicon \
   expression="E >= 85.0 ? 0.3*cos(theta) : 0."
```

NOTE The `expression` parameter is supported only if the GNU compiler gcc is installed on the system where Sentaurus Topography 3D is running and it is globally available there.

When `sputtering=false` for a certain species and material pair, sputtering by particles of that species from that material is disabled.

As for ion distributions, Sentaurus Topography 3D supports collections of yield functions. For example, the following commands define a collection of energy-independent yield functions called `my_yield` for flux `I` and several target materials:

```
define_yield name=my_yield species=I energy=0 material=Silicon \
   theta_max=55 yield_max=1.5
define_yield name=my_yield species=I energy=0 material=Oxide s1=6.0 s2=-5.5
define_yield name=my_yield species=I energy=0 material=Nitride theta_max=60 \
   yield_max=1.75
define_yield name=my_yield species=I energy=0 material=Photoresist \
   table=$yield_table
```

NOTE Currently, energy-dependent yield functions can be used only with reaction models, but not with level set–based models.

deposit

The `deposit` command starts a deposition simulation either by using a machine defined using the `define_deposit_machine` command or by geometrically depositing a shape defined by the `define_shape` command.

Deposition simulation on the surface of a structure using a machine:

```
deposit spacing=<v> time=<n> [cfl=<n>] [comment=<c>] \
   [diffusion_flux_error_control_strategy=<c>] \
   [diffusion_flux_integration_error=<n>] [engine=<c>] \
   [extraction=<c> extraction_interval=<n>] \
   [file=<c>] [flat_orientation=<v>] \
   [integration_error=<n> | integration_samples=<n>] \
   [machine=<c>] [max_ion_reflections=<n>] [merge=<c>] \
   [min_ion_reflection_probability=<n>] [plot_interval=<n> [plot_type=<l>]] \
   [region=<c>] [remove_spurious_parts=<c>] \
   [stop_plane=<n> | stop_point=<v>] [structure=<c>] \
   [update_scheme=<c>] [variance_reduction=<c>] [vertical_orientation=<v>]
```

Geometric deposition using a shape:

```
deposit shape=<c> material=<c> [comment=<c>] [flat_orientation=<v>] \
   [merge=<c>] [region=<c>] [structure=<c>] [vertical_orientation=<v>]
```

Table 37 Parameters of deposit command

Parameter	Type	Description	Default value [Possible values]	Unit
`cfl`	number	Courant–Friedrichs–Lewy (CFL) number [7] for the integration of the surface evolution equation.	`0.95 (when using a machine having model=simple), 0.5 (when using a machine not having model=simple) ]0, 1]`	none
`comment`	character	Sets a comment for this process step that will be displayed or written to the log file.	empty string	none
`diffusion_flux_error_control_strategy`	character	Specifies whether the parameter `diffusion_flux_integration_error` calculates an absolute or a relative error.	`relative [absolute, relative]`	none
`diffusion_flux_integration_error`	number	Sets the requested flux error of the diffusing species.	`0.05 [0, 1]`	none

Table 37 Parameters of deposit command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
engine	character	Sets the engine to use for flux computation. This choice is available only for flux models.	radiosity [monte_carlo, radiosity]	none
extraction	character	The name of the group of extractions to run during this process step.	none	none
extraction_ interval	number	Sets the process time interval after which intermediate extractions must be executed.	none]0, ∞[	minute
file	character	Sets the file name into which intermediate surfaces and surface data should be saved. Used in conjunction with plot_interval only.	basename.tdr	none
flat_orientation	vector	Specifies the Miller index of the direction perpendicular to the wafer flat (y-axis of the wafer coordinate system) for the newly created region.	{1 1 0}	none
integration_error	number	Specifies the maximum relative discretization error in the direct flux calculation.	0.05]0, 1]	none
integration_ samples	number	Sets the number of integration points in the numeric integration. The meaning of this parameter depends on the value of the engine parameter: • engine=radiosity: integration_samples describes the number of discretization points, per surface element, in the hemisphere where visibility is determined. • engine=monte_carlo: integration_samples describes the number of particle paths, per surface element, used in the flux integration. **NOTE** This parameter is deprecated for engine=radiosity. Instead, use the integration_error parameter.	1000 (engine=radiosity), 700 (engine=monte_carlo and variance_reduction= version_M), 350 (engine=monte_carlo and variance_reduction= version_N) [1, ∞[	none
machine	character	Sets the name of the machine that is used for the deposition process.	default_machine	none
material	character	Sets the material to be deposited.	none	none

Table 37 Parameters of deposit command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
max_ion_ reflections	number	The maximum number of ion reflections to compute during flux integration. The specified value must be an integer. Only applicable when using RFM models.	1 [1, ∞ [	none
merge	character	If true, the deposited region merges with the already existing regions of the same material in the structure.	false [false, true]	none
min_ion_ reflection_ probability	number	The threshold for the reflection probability below which multiple reflection is stopped.	0.001 [0, 1]	none
plot_interval	number	Sets the process time interval after which intermediate exposed surfaces and surface data should be plotted.	none [0, ∞ [	minute
plot_type	list	Specifies the type of data to save at the time intervals specified by plot_interval. The available types are exposed surface with datasets or multiregion boundary.	surface [structure, surface]	none
region	character	Sets the name of the region that is deposited.	none	none
remove_spurious_ parts	character	Specifies whether to remove spurious parts after running the process step.	false [false, true]	none
shape	character	Sets the name of the shape that is used for the deposition process.	none	none
spacing	vector	Defines the cell size of the grid used to compute the surface evolution for each dimension. A vector with only one component can be used to set the grid cell size to the same value for all dimensions. For a machine using any model other than spin_on, for 2D and 3D structures, a vector with two and three components, respectively, can be used to specify the grid cell size separately for each dimension. For a machine using the spin_on model, for 3D structures, a vector with two components can be used to specify the grid cell size separately for each dimension. See Discretization Size and Accuracy on page 8.	none	µm

Table 37 Parameters of deposit command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
stop_plane	number	The vertical coordinate of a horizontal plane above the exposed surface. When the exposed surface reaches or crosses this plane, time-stepping stops.	none	µm
stop_point	vector	The coordinates of a point above the exposed surface. When the exposed surface reaches or crosses this point, time-stepping stops.	none	µm
structure	character	Sets the name of the structure on which deposition will occur.	default_structure	none
time	number	Simulation time.	none	minute
update_scheme	character	Specifies the update scheme for the time integration of the level-set function.	upwind [lax_friedrichs, upwind]	none
variance_ reduction	character	Specifies the algorithm to use for variance reduction by the Monte Carlo flux integrator.	version_N [version_M, version_N]	none
vertical_ orientation	vector	Specifies the Miller index of the direction perpendicular to the wafer surface (z-axis of the wafer coordinate system) for the newly created region.	{0 0 1}	none

NOTE The parameter `structure` cannot specify a PMC structure. To process such a structure, an explicit conversion of the structure is needed before calling the `deposit` command (see Integration With Other Sentaurus Topography 3D Functionality on page 231).

NOTE The parameters `cfl`, `file`, `merge`, `plot_interval`, `plot_type`, `stop_plane`, `stop_point`, and `update_scheme` are not supported when using a machine having `model=crystal`.

NOTE The parameters `stop_plane`, `stop_point`, and `update_scheme` are not supported when using a machine having `model=spin_on`.

NOTE For machines having `model=spin_on`, structures whose exposed surface cannot be described as a single-valued function of the x- and y-coordinates in three dimensions, or of the x-coordinate in two dimensions, are not supported. However, topologically connected exposed surfaces containing elements parallel to the vertical direction are supported.

NOTE The Miller indices for a plane are usually written as (hkl), and the set of equivalent planes is written as {hkl}. A crystal direction is usually written as [hkl], and the set of equivalent directions is written as <hkl>. The value assigned to the parameters `flat_orientation` and `vertical_orientation` is a Tcl list of indices representing a crystal direction. The braces in the specified value are required to form a Tcl list and do not indicate a set of equivalent planes.

Examples

Example of a deposition process using a machine:

```
deposit machine=depomachine spacing={0.05 0.3 0.05} time=0.5
```

This example simulates a deposition process on the structure, using the machine `depomachine` with a grid spacing of dx=0.05 µm, dy=0.3 µm, dz=0.05 µm, and a simulation time of 0.5 minutes.

NOTE The resolution is adjusted internally to match the actual structure.

Example of a deposition process using a shape:

```
deposit shape=shape_1 material=Oxide comment="deposit shape"
```

This command deposits the shape `shape_1` onto the structure. The created region will consist of the material `Oxide`. The name of the process step is “deposit shape” and is displayed in the log file.

NOTE Parts of `shape_1` that overlap the existing structure are not added.

Example of a deposition process with intermediate surface plotting:

```
deposit machine=depomachine spacing={0.1} time=0.5 \
   file=depo_surfaces.tdr plot_interval=0.1
```

This command deposits using the machine `depomachine`, with a grid spacing of dx=dy=dz=0.1 µm. Intermediate surfaces along with data such as surface velocities and fluxes are plotted into the file `depo_surfaces.tdr`. The surface of the first and last time steps of the simulation will always be plotted. During the simulation, whenever the simulation time surpasses an integral multiple of `plot_interval` (for example, 0.1, 0.2, 0.3...), a surface will be plotted to the file. This feature can check process parameters and model behavior.

The stop_plane and stop_point Parameters

The parameters `stop_plane` and `stop_point` of the `deposit` command allow you to stop time-stepping when a specified plane or point has been reached before the final process time.

The parameters `stop_plane` and `stop_point` can be used when processing both 2D and 3D structures.

The dimension of the point specified with `stop_point` must match the dimension of the structure for which it is used.

The values specified with `stop_plane` and `stop_point` must fulfill certain constraints with respect to the computational domain and the initial exposed surface.

The computational domain and the initial exposed surface are defined in Boundary Types on page 13:

- When using `stop_plane` with the `deposit` command, the specified plane must intersect the computational domain, and the specified plane must be above the initial exposed surface.
- When using `stop_point` with the `deposit` command, the specified point must be inside the computational domain, and the specified point must lie above the initial exposed surface.

Due to the discrete time steps, the exposed surface can cross the specified plane or point in the last time step. The maximum distance of the final exposed surface beyond the specified plane or point is limited by the value of the parameter `spacing` that specifies the discretization of the level-set grid.

When using `stop_plane` or `stop_point`, the output written to the log file indicates why the time-stepping of the corresponding process step stopped.

If the specified plane or point was reached or crossed, the output indicates this. The output includes the specified plane or point and the process time at which this occurred, for example:

```
Exiting loop early: stop_plane= <> [um] has been reached at t= <>.
Exiting loop early: stop_point= {} [um] has been reached at t= <>.
```

If the specified plane or point is not reached or crossed, this is indicated in the output together with the specified plane or point and the specified process time, for example:

```
stop_plane= <> [um] has not been reached at any time step within
   time= <> [min].

stop_point= {} [um] has not been reached at any time step within
   time= <> [min].
```

Examples:

This command specifies a stop plane for deposition on a 2D or 3D structure:

```
deposit spacing=0.1 time=1.0 stop_plane=1.0
```

This command specifies a stop point for deposition on a 3D structure:

```
deposit spacing=0.1 time=1.0 stop_point={0.0 0.5 0.3}
```

This command specifies a stop point for deposition on a 2D structure:

```
deposit spacing=0.1 time=1.0 stop_point={0.0 0.5}
```

TDR Dataset Names for Fluxes

For models that calculate ion or neutral fluxes, the values of the fluxes are written to TDR datasets when using the parameter `plot_interval`.

The name of a flux dataset starts with the name of the flux species. If the flux takes a yield function into account, it is followed by `yield`.

The type of flux is indicated by one of the following suffixes:

`direct`	The direct flux that arrives from the plasma at a surface element.
`reactive`	The incoming flux at a surface element that is available for reemission.
`reflected`	The direct flux that has been reflected away from a surface element.
`reflection`	The incoming flux at a surface element that has been reflected from other surface elements.
`total`	Sum of the reactive flux and the incoming indirect flux caused by the reactive flux.

etch

The `etch` command starts an etching simulation or a simultaneous etching and deposition simulation, as well as a deposition simulation based on a reaction model, and can be used in one of the following ways:

- Using a machine defined by `define_etch_machine`.
- Geometrically etching a shape defined with the `define_shape` command.
- Geometrically etching a mask defined with the `define_mask` command.

Etching simulation on the surface of the structure using a machine with `method=levelset`:

```
etch method=levelset spacing=<v> time=<n> \
   [cfl=<n>] [comment=<c>] \
   [diffusion_flux_error_control_strategy=<c>] \
   [diffusion_flux_integration_error=<n>] \
   [engine=<c>] [extraction=<c> extraction_interval=<n>] \
   [file=<c>] [fill_material=<c>] [fill_region=<c>] [flat_orientation=<v>] \
   [integration_error=<n> | integration_samples=<n>] [machine=<c>] \
   [max_ion_reflections=<n>] \
   ([abs_min_deposition_thickness=<n>] | min_deposition_thickness=<n>) \
   [merge=<c>] [min_ion_reflection_probability=<n>] \
   [pad_pressure_acceleration=<c>] [pad_pressure_accuracy=<n>] \
   [pad_pressure_max_iterations=<n>] [plot_interval=<n> [plot_type=<l>]] \
   [region=<c>] [region_query_accuracy=<c>] [remove_spurious_parts=<c>] \
   [selective_deposition=<c> [selective_deposition_threshold=<n>]] \
   [stop_plane=<n> | stop_point=<v>] [structure=<c>] \
   [update_scheme=<c>] [variance_reduction=<c>] [vertical_orientation=<v>]
```

NOTE The `fill_material` and `fill_region` parameters are not available for simultaneous etching and deposition models.

NOTE Etching using a machine with `method=levelset` is not supported when the structure specified with the parameter `structure` is a PMC structure. To process such a structure, an explicit conversion of the structure is needed before calling the `etch` command (see Integration With Other Sentaurus Topography 3D Functionality on page 231).

Etching simulation on the surface of the structure using a machine with `method=pmc`:

```
etch method=pmc spacing=<v> time=<n> [averaging_interval=<n>] \
   [averaging_runs=<n>] [comment=<c>] [compute_process_graph=<c>] \
   [deposition_method=<c>] [extraction=<c> extraction_interval=<n>] \
   [file=<c>] [machine=<c>] [max_number_edges_process_graph=<n>] \
   [max_number_reflections=<n>]
   [plot_interval=<n> [plot_type=<l> [gc_samples_per_cell=<n>] \
```

```
   [num_flux_orders=<n>] [plot_quality=<n>]]] \
[selective_deposition=<c>] \
[stop_material=<c> | stop_plane=<n> | stop_point=<v>] [structure=<c>] \
[top_gas_thickness=<n>]
```

Geometric etching using a shape:

```
etch shape=<c> [comment=<c>] [fill_material=<c>] [fill_region=<c>] \
   [material=<c>] [structure=<c>]
```

Geometric etching using a mask:

```
etch mask=<c> thickness=<n> [comment=<c>] [fill_material=<c>] \
   [fill_region=<c>] [structure=<c>]
```

NOTE Geometric etching using a shape or using a mask is not supported when the structure specified with the parameter `structure` is a PMC structure. To process such a structure, an explicit conversion of the structure is needed before calling the `etch` command (see Integration With Other Sentaurus Topography 3D Functionality on page 231).

NOTE Geometric etching using a mask is not supported for 2D structures.

Table 38 Parameters of etch command

Parameter	Type	Description	Default value [Possible values]	Unit
`abs_min_deposition_thickness`	number	Specifies the absolute minimum thickness a deposited layer must grow during a time step to be created or added.	`0` `[0, ∞[`	µm
`averaging_interval`	number	The time interval after which the different PMC runs are averaged. If the parameter is omitted, the value of the `time` parameter is used.	none `]0, ∞[`	minute
`averaging_runs`	number	The number of PMC runs to be averaged.	`1` `[1, ∞[`	none

`cfl`	number	Courant–Friedrichs–Lewy (CFL) number [7] for the integration of the surface evolution equation.	`0.95 (when using a machine having model=simple),`
			`0.5 (when using a machine not having model=simple)`
			`]0, 1]`
				none

`comment`	character	Sets a comment for this process step that will be displayed or written to the log file.	empty string	none

Table 38 Parameters of etch command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
compute_process_graph	character	Specifies whether to compute the process graph during a PMC simulation.	false [false, true]	none
deposition_method	character	Selects the PMC deposition method. The methods differ with respect to the surface diffusion model, which affects the smoothness and the shape evolution of the deposited film.	minimum [generic, minimum, patch]	none
diffusion_flux_error_control_strategy	character	Specifies whether the value of the parameter diffusion_flux_integration_error is an absolute or a relative error.	relative [absolute, relative]	none
diffusion_flux_integration_error	number	The requested flux error of the diffusing species.	0.05 [0, 1]	none
engine	character	Sets the engine to use for flux computation. This choice is available only for flux models.	radiosity [monte_carlo, radiosity]	none
extraction	character	The name of the group of extractions to run during this process step.	none	none
extraction_interval	number	Sets the process time interval after which intermediate extractions must be executed.	none]0, ∞[	minute
file	character	Sets the file into which intermediate surfaces and surface data should be saved. Used in conjunction with plot_interval only.	basename.tdr	none
fill_material	character	Sets the material to fill the etched volume. If this parameter is omitted, the etched volume is not filled.	none	none
fill_region	character	Sets the name of the region filling the etched volume. The structure specified with the parameter structure must not contain regions with this name. If fill_region is omitted, the name of the region filling the etched volume is generated automatically.	none	none
flat_orientation	vector	Specifies the Miller index of the direction perpendicular to the wafer flat (y-axis of the wafer coordinate system) for the newly created region for models that perform simultaneous etching and deposition.	{1 1 0}	none

Table 38 Parameters of etch command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
gc_samples_per_cell	number	Specifies the number of samples to take along the x- and y-directions per grid cell.	1 [1, 2147483647]	none
integration_error	number	Specifies the maximum relative discretization error in the direct flux calculation.	0.05]0, 1]	none
integration_samples	number	Sets the number of integration points in the numeric integration. The meaning of this parameter depends on the value of the engine parameter: • engine=radiosity: integration_samples describes the number of discretization points, per surface element, in the hemisphere where visibility is determined. • engine=monte_carlo: integration_samples describes the number of particle paths, per surface element, used in the flux integration. **NOTE** This parameter is deprecated for engine=radiosity. Instead, use the integration_error parameter.	1000 (engine=radiosity), 700 (engine=monte_carlo and variance_reduction=version_M), 350 (engine=monte_carlo and variance_reduction=version_N) [1, ∞ [	none
machine	character	Sets the name of the machine that is used for the etch process.	default_machine	none
mask	character	Sets the name of the mask to be used for the etch process.	none	none
material	character	Sets the material that can be etched. If omitted, all materials can be etched.	none	none
max_ion_reflections	number	The maximum number of ion reflections to compute during flux integration. The specified value must be an integer. Only applicable when using RFM models.	1 [1, ∞ [	none
max_number_edges_process_graph	number	The maximum number of edges of the process graph that can be visualized.	10000 [1, 2147483647]	none
max_number_reflections	number	The number of reflections a species can undergo before being discarded.	2000000 [0, 2147483647]	none
merge	character	If true, the deposited region merges with the already existing regions of the same material in the structure.	false [false, true]	none

Table 38 Parameters of etch command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
method	character	The simulation method to use.	levelset [levelset, pmc]	none
min_deposition_ thickness	number	Specifies the minimum thickness a deposited layer must grow during a time step to be created or added as a fraction of the mean spacing.	0.01 [0, 1]	none
min_ion_ reflection_ probability	number	The threshold for the reflection probability below which multiple reflection is stopped.	0.001 [0, 1]	none
num_flux_orders	number	The number of flux orders to produce for each PMC intermediate surface plot. It can be specified only when plot_type contains the value surface.	1 [1, 2147483647]	none
pad_pressure_ acceleration	character	Enables an accelerated solver to evaluate the RFM function pad_pressure(). **NOTE** You can set this parameter to false only when processing 2D structures.	true [false, true]	none
pad_pressure_ accuracy	number	The relative residual below which the solver stops when evaluating the RFM function pad_pressure().	1e-3 [0, ∞ [	none
pad_pressure_max_ iterations	number	The maximum number of iterations that are allowed to evaluate the RFM function pad_pressure().	100 [1, 2147483647]	none
plot_interval	number	Sets the process time interval after which intermediate exposed surfaces and surface data should be plotted.	none [0, ∞ [	minute
plot_quality	number	Specifies the quality of a PMC intermediate surface plot. The larger its value, the better the plot quality. It can be specified only when plot_type contains the value surface.	1]0, ∞ [	none
plot_type	list	Specifies the type of data to save at time intervals specified by plot_interval.	surface [structure, surface] (when method=levelset), [gc, surface, vbe] (when method=pmc)	none
region	character	Sets the name of the region that can be redeposited in advanced etching models. Not valid for etching models that do not create redeposition phenomena.	none	none

Table 38 Parameters of etch command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
region_query_accuracy	character	Sets the accuracy used to determine the regions and materials associated with each element of the exposed surface. See Discretization Size and Accuracy on page 8.	resolution [resolution, subresolution]	none
remove_spurious_parts	character	Specifies whether to remove spurious parts after running the process step.	true [false, true]	none
selective_deposition	character	If activated, it suppresses the surface spreading of deposited material across material interfaces, thereby avoiding lateral creeping of the deposited film.	false (for method=levelset) true (for method=pmc) [false, true]	none
selective_deposition_threshold	number	Specifies the threshold for the thickness of a layer below which the deposited material is considered to be an artifact. The specified value is relative to the mean discretization size.	1.0 [0.0, ∞ [	none
shape	character	Sets the name of the shape that is used for the etch process.	none	none
spacing	vector	Defines the size of a grid cell for each direction. A vector with one component can be used to set the grid cell size to the same value for all dimensions. When method=pmc, this is the only supported option. Whereas, when method=levelset, nonuniform grid cell sizes are supported. When method=levelset, for 2D and 3D structures, a vector with two and three components, respectively, can be used to specify the grid cell size separately for each dimension. See Discretization Size and Accuracy on page 8.	none	µm
stop_material	character	Etching stops when the surface reaches the specified bulk material.	none	none
stop_plane	number	The vertical coordinate of a horizontal plane below the exposed surface and not intersecting any void. When the exposed surface reaches or crosses this plane, time-stepping stops.	none	µm

Table 38 Parameters of etch command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
stop_point	vector	The coordinates of a point below the exposed surface and not inside any void. When the exposed surface reaches or crosses this point, time-stepping stops.	none	µm
structure	character	Sets the name of the structure to be etched.	default_structure	none
thickness	number	Specifies the etch depth when etching with a mask, measured relative to the exposed surface of the structure.	none [0, ∞ [	µm
time	number	Simulation time.	none	minute
top_gas_thickness	number	The minimum thickness of the gas region added on top of the structure when using a boundary as input.	0 [0, ∞ [	µm
update_scheme	character	Specifies the update scheme for the time integration of the level-set function.	upwind [lax_friedrichs, upwind]	none
variance_ reduction	character	Specifies the algorithm to use for variance reduction by the Monte Carlo flux integrator.	version_N [version_M, version_N]	none
vertical_ orientation	vector	Specifies the Miller index of the direction perpendicular to the wafer surface (z-axis of the wafer coordinate system) for the newly created region for models that perform simultaneous etching and deposition.	{0 0 1}	none

NOTE Currently, when `method=pmc`, the grid spacing specified with the parameter `spacing` is ignored if the structure specified with the parameter `structure` stores the result of a previous simulation using the PMC method.

NOTE Currently, when running a simulation with a machine using a reaction model that allows deposition, the vertical extent of the computational domain (see Boundary Types on page 13) must be large enough to contain the deposited material. The vertical extent of the computational domain can be increased by adding `Gas` to the initial structure using the `fill` command (see fill on page 185).

NOTE For machines using RFM models with a rate formula involving the function `pad_pressure()`, 2D structures whose exposed surface cannot be described as a single-valued function of the x-coordinate are not supported.

NOTE The parameters `abs_min_deposition_thickness`, `cfl`, `file`, `min_deposition_thickness`, `plot_interval`, `plot_type`, `region_query_accuracy`, `stop_plane`, `stop_point`, and `update_scheme` are not supported when using a machine having `model=crystal`.

NOTE The Miller indices for a plane are usually written as (hkl), and the set of equivalent planes is written as {hkl}. A crystal direction is usually written as [hkl], and the set of equivalent directions is written as <hkl>. The value assigned to the parameters `flat_orientation` and `vertical_orientation` is a Tcl list of indices representing a crystal direction. The braces in the specified value are required to form a Tcl list and do not indicate a set of equivalent planes.

NOTE When using a combination of the parameters `averaging_interval`, `extraction_interval`, and `plot_interval` at the same time in a PMC simulation, the specified values must be the same.

Examples

Example of an etch process using an etch machine:

```
etch machine=etchmachine spacing={0.05 0.3 0.05} time=0.5
```

This command simulates an etching process on the structure, using the machine `etchmachine` with a grid spacing of dx=0.05 µm, dy=0.3 µm, dz=0.05 µm, and a simulation time of 0.5 minutes.

NOTE The resolution is adjusted internally to match the actual structure.

Example of an etch process using a shape:

```
etch shape=shape_1 comment="etch shape"
```

This command etches the shape `shape_1` from the structure. The comment given to the process step is “etch shape” and is displayed in the log file.

Example of an etch process using a mask:

```
etch mask=mask_1 thickness=0.1 comment="etch mask"
```

This command uses the mask `mask_1` to etch 0.1 µm, measured relative to the exposed surface of the structure. The comment for the process step is `"etch mask"` and is displayed in the log file.

Example of an etching process with intermediate surface plotting:

```
etch machine=etchmachine spacing={0.1} time=0.5 file=etch_surfaces.tdr \
   plot_interval=0.1
```

This command etches using the machine `etchmachine`, with a grid spacing of dx=dy=dz=0.1 µm. Intermediate surfaces along with data such as surface velocities and fluxes are plotted into the file `etch_surfaces.tdr`. The surface of the first and last time steps of the simulation will always be plotted. During the simulation, whenever the simulation time surpasses an integral multiple of `plot_interval` (for example, 0.1, 0.2, 0.3...), a surface will be plotted to file. This feature is useful to check process parameters and model behavior.

The stop_material, stop_plane, and stop_point Parameters

The `stop_plane` and `stop_point` parameters can be used with the `etch` command in the same way as for the `deposit` command. For details, see The stop_plane and stop_point Parameters on page 152.

The only differences are the constraints on the values specified with `stop_plane` and `stop_point`:

- When using `stop_plane` with the `etch` command, the specified plane must intersect the computational domain. In addition, the specified plane must be below the initial exposed surface and must not intersect any voids.
- When using `stop_point` with the `etch` command, the specified plane must intersect the computational domain. In addition, the specified point must be below the exposed surface and must not lie inside any void.

When `method=pmc`, the parameter `stop_material` can be used to stop etching when the specified material is used for the first time as a bulk material in a reaction.

TDR Dataset Names for Fluxes

For models that calculate ion or neutral fluxes, the values of the fluxes are written to TDR datasets when using the parameter `plot_interval`.

The name of a flux dataset starts with the name of the flux species. If the flux takes a yield function into account, it is followed by `yield`.

The type of flux is indicated by one of the following suffixes:

`direct`	The direct flux that arrives from the plasma at a surface element.
`reactive`	The incoming flux at a surface element that is available for reemission.
`reflected`	The direct flux that has been reflected away from a surface element.
`reflection`	The incoming flux at a surface element that has been reflected from other surface elements.
`total`	Sum of the reactive flux and the incoming indirect flux caused by the reactive flux.

Variance Reduction in PMC Simulations

PMC simulations use pseudo-random numbers to compute the evolution of the structure under the specified process conditions. When the sequence of pseudo-random numbers changes (for example, due to a different scheduling of the threads by the operating system), the results might also change, to an extent that depends on the model used and on the simulation parameters. This variability can be reduced by running the same simulation multiple times independently and by merging the different results.

You can control the number of PMC simulations to be run and merged together using the `averaging_runs` parameter. In addition, you can average the results at the end of the simulation or after a certain amount of time, which can be specified by the `averaging_interval` parameter.

For example, the following command runs a PMC simulation of a 1 minute etching process 8 times and takes the average of the eight runs as the final result:

```
etch time=1 spacing=0.01 method=pmc averaging_runs=8
```

The following command splits the simulation time into five intervals, each of which is 0.2 minute long:

```
etch time=1 spacing=0.01 method=pmc averaging_runs=8 averaging_interval=0.2
```

The evolution of the structure is simulated 8 times over each time interval, and the average of the obtained results is used as the initial structure for the simulation over the next interval.

extend_structure

The `extend_structure` command extends an existing structure by mirroring it, or by copying and shifting it.

```
extend_structure plane=<c> type=mirror [structure=<c>]

extend_structure plane=<c> type=shift [structure=<c>] [times=<n>]
```

Table 39 Parameters of extend_structure command

Parameter	Type	Description	Default value [Possible values]	Unit
`plane`	character	Specifies at which side the initial structure will be extended (see Figure 18). **NOTE** The values `back` and `front` can be used only for 3D structures.	none `[back, front, left, right]`	none
`structure`	character	The name of the structure that is extended.	`default_structure`	none
`times`	number	Specifies the number of times the initial structure is copied and shifted.	`1` `[1, ∞ [`	none
`type`	character	Specifies which operation is used to extend the initial structure.	none `[mirror, shift]`	none

NOTE The parameter `structure` cannot specify a PMC structure. To process such a structure, an explicit conversion of the structure is needed before calling the `extend_structure` command (see Integration With Other Sentaurus Topography 3D Functionality on page 231).

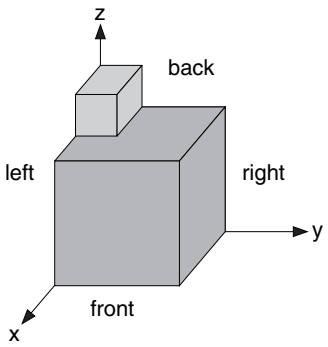


Figure 18 Initial structure and different values of the plane parameter

The initial structure and the extension created by mirroring, or copying and shifting, are merged automatically into one structure.

NOTE When a structure is extended by copying and shifting (`type=shift`), the resulting structure might contain a path that connects the top gas with the bottom plane of the simulation domain, without passing through any other material. These types of structure are invalid for machine-based etching and deposition processes. There is no such problem with structures extended by mirroring.

Examples

This command extends the default structure by mirroring at the xz plane at the maximum y-coordinate of the simulation domain:

```
extend_structure type=mirror plane=right
```

Figure 19 shows the result of applying this command to the structure in Figure 18 on page 164.

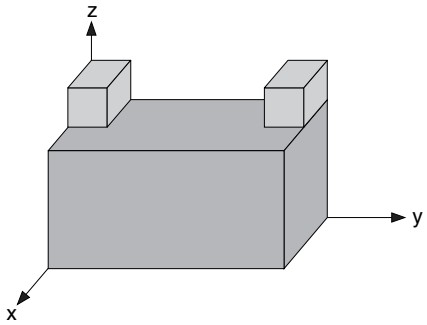


Figure 19 Extending structure by mirroring

This command extends the default structure by copying the default structure and shifting the copy to the xz plane at the maximum y-coordinate of the simulation domain. Then, a second copy is shifted to the maximum y-coordinate of the intermediate structure created by adding the first copy (see Figure 20):

```
extend_structure type=shift plane=right times=2
```

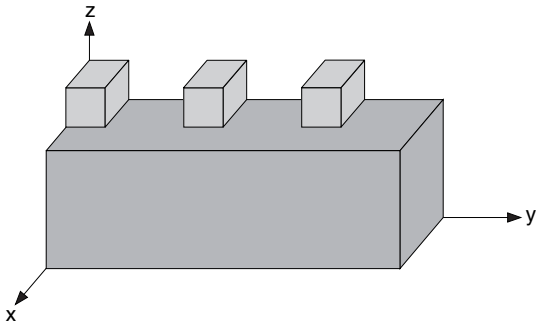


Figure 20 Extending structure by copying and shifting twice

extract

The `extract` command extracts different properties of a structure. The main categories are:

- One-dimensional or 2D cuts of the processed structure.
- Properties of the exposed surface.
- The area of the interface between two regions.
- The surface area and the volume of regions and parts.
- The materials of a structure, the materials of regions, and the names of regions.
- The names of regions and parts.
- The bounding box of materials and regions.
- The vertical coordinates of the top and the bottom of the bounding box of materials and regions.
- The position of an interface between two different materials, two different regions, or two different parts along a line in a structure.
- The shortest distance between two different materials, two different regions, or two different parts in a structure.
- The intersection of a structure with a line.
- The intersection of a 3D structure with a plane.

NOTE The `extract` command can extract properties from a structure only before or after a processing step. To extract properties during processing, see define_extraction on page 114.

Specification of Extraction Lines, Planes, and Pairs

For some extractions, an extraction line, an extraction plane, or extraction pairs of materials or regions must be specified.

An extraction line can be specified in these different ways:

- Axis-aligned direction and one point:

```
axis=<c> point=<v>
```

- Arbitrary direction and one point:

```
direction=<c> point=<v>
```

- Two points:

```
point1=<v> point2=<v>
```

An extraction plane can be specified in these different ways:

- Axis-orthogonal plane at axis position:

  ```
  axis=<c> position=<n>
  ```

- Arbitrary plane-normal direction and one point:

  ```
  normal=<v> point=<v>
  ```

- Generic plane defined by three points:

  ```
  point1=<v> point=2<v> point3=<v>
  ```

An extraction pair can be specified using pairs of identifiers:

- For two different materials:

  ```
  material1=<c> material2=<c>
  ```

- For two different regions:

  ```
  region1=<c> region2=<c>
  ```

- For two different parts:

  ```
  region_part1=<l> region_part2=<l>
  ```

The names of regions can be extracted with `type=region_names` as described in Region Names and Materials on page 169. Lists of region and part names can be extracted with `type=region_parts` (see Region and Part Names on page 170).

In the following syntax descriptions, the specifications for an extraction line, an extraction plane, and pairs of material or region are represented by `<extraction_line>`, `<extraction_plane>`, and `<extraction_pair>`, respectively.

Extraction Type

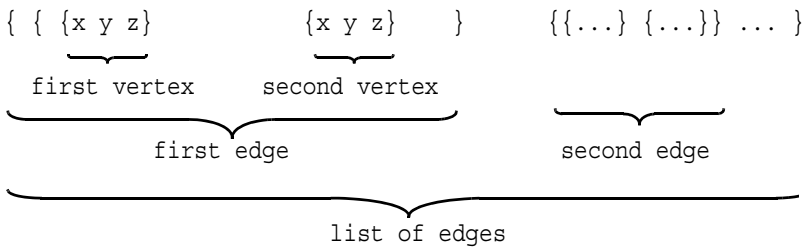
The extraction type is set with the parameter `type`.

1D or 2D Cuts

A `1d_cut` intersects the closed surface (defined in Boundary Types on page 13) with a line:

```
extract type=1d_cut <extraction_line> [file=<c>] [name=<c>] [silent=<c>] \
   [structure=<c>]
```

The data returned by `type=1d_cut` is a list of edges represented in Tcl with the following format:



where:

- A vertex is a list of two or three floating-point numbers, depending on the dimension of the structure.
- An edge is a list of exactly two vertices.
- A 1D cut is a list of edges.

A `2d_cut` intersects the closed surface with a plane:

```
extract type=2d_cut <extraction_plane> [file=<c>] [name=<c>] [silent=<c>] \
   [structure=<c>]
```

Exposed Surface

To extract the properties of the exposed surface:

- Extraction of the bounding box of the exposed surface:

  ```
  extract type=bounding_box_exposed [name=<c>] [silent=<c>] [structure=<c>]
  ```

  Returns a list containing the minimum and maximum vertices of the bounding box of the exposed surface.

- Extraction of the minimum vertical coordinate of the exposed surface:

  ```
  extract type=bottom_exposed [name=<c>] [silent=<c>] [structure=<c>]
  ```

  Returns the minimum vertical coordinate of the exposed surface.

- Extraction of the maximum vertical coordinate of the exposed surface:

  ```
  extract type=top_exposed [name=<c>] [silent=<c>] [structure=<c>]
  ```

  Returns the maximum vertical coordinate of the exposed surface.

Interface Area

To extract the interface area:

- Extraction of the area of the interface between two regions:

```
extract region_1=<c> region_2=<c> type=interface_area [name=<c>] \
   [silent=<c>] [structure=<c>]
```

  Returns the total area of the interface between the two specified regions.

Areas and Volumes of Regions and Parts

To extract geometric properties:

- Extraction of the surface area of a region:

```
extract region=<c> type=area [name=<c>] [silent=<c>] [structure=<c>]
```

  Returns the total surface area of the specified region.

- Extraction of the surface areas of the parts of a region:

```
extract region=<c> type=part_area [name=<c>] [silent=<c>] [structure=<c>]
```

  Returns the surface areas of all disconnected parts of the specified region as a Tcl list.

- Extraction of the volumes of the parts of a region:

```
extract region=<c> type=part_volume [name=<c>] [silent=<c>] \
   [structure=<c>]
```

  Returns the volumes of all disconnected parts of the specified region as a Tcl list.

- Extraction of the volume of a region:

```
extract region=<c> type=volume [name=<c>] [silent=<c>] [structure=<c>]
```

  Returns the total volume of the specified region.

Region Names and Materials

To extract region names and materials:

- Extraction of material names:

```
extract type=material_names [exposed_only=<c>] [name=<c>] [silent=<c>] \
   [structure=<c>]
```

  Returns the names of the materials of a structure as a Tcl list.

- Extraction of region materials:

```
extract region=<c> type=material_name [exposed_only=<c>] [name=<c>] \
   [silent=<c>] [structure=<c>]
```

  Returns the name of the material of the specified region.

- Extraction of region names:

```
extract type=region_names [exposed_only=<c>] [name=<c>] [silent=<c>] \
   [structure=<c>]
```

  Returns the names of the regions of a structure as a Tcl list.

Region and Part Names

To extract the names of regions and the names of their parts:

- Extraction of all region and part names of a structure:

```
extract type=region_parts [exposed_only=<c>] [name=<c>] [silent=<c>] \
   [structure=<c>]
```

  Returns a list of pairs. The first element of each pair is the name of a region of the structure and the second element is a list of the names of the parts contained in that region.

- Extraction of part names of a region of a structure:

```
extract region=<c> type=region_parts [exposed_only=<c>] [name=<c>] \
   [silent=<c>] [structure=<c>]
```

  Returns a list containing the names of the parts in the given region.

Interface Position

Specifying `type=interface` locates an interface between two different materials, or two different regions, or two different parts along a line in a structure, as well as obtains a vertex for the interface that the line passes through or a segment of the interface that the line is tangent to:

```
extract type=interface <extraction_line> <extraction_pair> [name=<c>] \
   [output=<c>] [silent=<c>] [structure=<c>]
```

If the `material1` and `material2` parameters are used, the special name `Gas` can be used to represent not only explicitly available gaseous regions, but also the empty space at the exposed surface.

If you are only interested in finding interfaces between two specific but different regions, use the `region1` and `region2` parameters.

To find the interfaces between two specific but different parts, use the `region_part1` and `region_part2` parameters.

The `output` parameter allows you to filter the computed interface positions, and it can have the following values:

- `output=all`: Along the input line, returns all interfaces in the output format described below. This is the default.
- `output=first`: Returns only the first interface.
- `output=inside`: Returns only the vertices or segments that are fully contained within or exactly on the `point1` to `point2` input segment. Only available if the `extract` command is specified with `point1` and `point2`.
- `output=last`: Returns only the last interface.
- `output=outside`: Returns only the vertices or segments that do not overlap the `point1` to `point2` input segment. Only available if the `extract` command is specified with `point1` and `point2`.

The data returned by `type=interface` is structured as follows:

```
{{x y z} {x y z} {x y z x y z} ...}
```

Each element of this list represents either a vertex or a segment.

The exposed surface is treated specially with `type=interface`: An interface is returned for the exposed surface even if the input structure does not contain a gas region that touches the exposed surface. The material name `Gas` can always be used for one of the two parameters `material1` or `material2`, irrespective of whether the structure actually contains a gas region touching the exposed surface. On the other hand, specifying an interface using either `region1` and `region2`, or `region_part1` and `region_part2`, with an assumed gas region is not allowed.

Intersections With a Line

Specifying `type=probe` intersects a structure with a line and returns a segment for each part of a region that the line passes through as well as a property associated with that segment, namely, the material name, or the region and part name:

```
extract type=probe <extraction_line> \
   (materials=<l> | [probe_empty_space=<c>]) \
   [name=<c>] [output=<c>] [property=<c>] [silent=<c>] [structure=<c>]
```

The property parameter specifies the data to be associated with each extracted segment in the returned value. This data refers to the part of a region where a segment comes from:

- property=length: The total length of the segments lying on the extraction line that go through the specified materials or through empty space.

  If probe_empty_space=false, the total length of the segments lying on the extraction line that go through any material except Gas or empty space is returned.

  If probe_empty_space=true, the total length of the segments lying on the extraction line that go through the material Gas or empty space is returned.

  If the parameter materials is specified, the total length of the segments lying on the extraction line that go through the specified materials is returned.
- property=material: The material name of the underlying part of a region. This is the default.
- property=region: The region name of the underlying part.
- property=region_part: A list of two elements. The first element is the region name of the underlying part, and the second element is the name of the underlying part itself. The name of the underlying part is always a number.

The output parameter can have the following values:

- output=all: Along the input line, returns all segments within regions in the output format described below. This is the default.
- output=first: Returns only the first entry.
- output=inside: Returns only the segments that are fully contained within or exactly on the point1 to point2 input segment. Only available if the extract command is specified with point1 and point2.
- output=last: Returns only the last entry.
- output=outside: Returns only the segments that do not overlap the point1 to point2 input segment. Only available if the extract command is specified with point1 and point2.

For property=length, the data returned by type=probe is a number that represents the sum of the length of the segments that lie in the specified materials or in empty space.

For property=material, property=region, and property=region_part, the data returned by type=probe is structured as follows:

```
{{{x y z} {x y z} property} {{x y z} {x y z} property} ...}
```

The returned property is enclosed in double quotation marks if it contains spaces.

Intersections With a Plane

Specifying `type=slice` intersects a 3D structure with a plane. The obtained 2D cut represents a 2D boundary that contains information about materials. The 2D boundary is saved into a TDR file:

```
extract type=slice <extraction_plane> [file=<c>] [name=<c>] [silent=<c>] \
   [structure=<c>]
```

The obtained 2D cut is saved into a TDR file specified using the `file` parameter.

The saved structure can be used as input for a 2D simulation if the exposed surface of the structure (excluding voids) does not touch the bottom of the simulation domain (see Boundary Types on page 13). In other words, if you imagine pouring a liquid onto the structure, and the liquid does not reach the bottom of the rectangular domain, the structure is valid input for a simulation.

Shortest Distance

```
extract type=shortest_distance <extraction_pair> [name=<c>] [silent=<c>] \
   [structure=<c>]
```

Specifying `type=shortest_distance` finds the shortest distance between two different materials, or two different regions, or two different parts in a structure.

The `material1` and `material2` parameters specify two different materials between which the shortest distance is sought.

The `region1` and `region2` parameters specify two different regions between which the shortest distance is sought. These two regions do not have to contain two different materials as in the case of the `material1` and `material2` parameters. All available parts of the two regions are considered in the search.

You can use `type=region_names` to extract a list of region names in a structure.

To find the shortest distance between two specific but different parts, use the `region_part1` and `region_part2` parameters.

You can use `type=region_parts` to extract a list of region and part names of a structure.

The data returned by `type=shortest_distance` is a list with the following two elements (an empty list is returned if one of the specified materials, regions, or parts does not exist in the structure, and a warning is issued):

```
{shortest_distance, segments}
```

The first element, `shortest_distance`, is the shortest distance measured in µm. The second element, `segments`, indicates where the shortest distance occurs. Since the shortest distance can occur at more than one location, `segments` is a list in the following format:

```
{segment1, ...} = { {vertex1, vertex2}, ...} = { { {x, y, z}, {x, y, z} }, ...}
```

The list contains only one segment if the shortest distance occurs uniquely at one specific location in the structure. When the shortest distance occurs at several locations, the number of segments depends on the discretization of the structure.

Cylindrical Hole Profile

```
extract type=cylinder_hole_profile center=<v> point_max=<v> point_min=<v> \
   [file=<c>] [structure=<c>]
```

For PMC structures, when specifying `type=cylinder_hole_profile`, the vertical profile of a cylindrical hole is extracted. The values specified with the parameters `point_max` and `point_min` define a cuboid within which, for each vertical coordinate, the average, the maximum, and the minimum radius and the standard deviation of the radius are extracted.

The extracted values are written to a file in comma-separated value (CSV) format. By default, the file name is the base name of the command file and it uses the suffix `_cyl.csv`. Alternatively, a file name can be specified with the parameter `file`. The resulting file can be visualized in Sentaurus Visual, Inspect, or other visualization tools.

NOTE The extraction of vertical profiles of cylindrical holes is only available for PMC structures.

Bounding Box of Materials and Regions

To extract the bounding box of regions having any of the specified materials, use the command:

```
extract type=bounding_box [materials=<l>] [name=<c>] [silent=<c>] \
   [structure=<c>]
```

This command returns the bounding box of regions of the given structure that have any of the specified materials. If no materials are specified, the bounding box of the entire structure is returned. If the list of materials is not empty but contains only materials that are not present in the specified structure, an error is issued.

NOTE When the structure specified with the `structure` parameter is the result of a PMC simulation, the specified list of materials must be empty.

To extract the bounding box of regions having any of the specified names, use the command:

```
extract type=bounding_box [name=<c>] [regions=<l>] [silent=<c>] \
   [structure=<c>]
```

This command returns the bounding box of regions of the given structure that have any of the specified region names. If no regions are specified, the bounding box of the entire structure is returned. If the list of regions is not empty but contains only regions that are not present in the specified structure, an error is issued.

NOTE When the structure specified with the `structure` parameter is the result of a PMC simulation, the specified list of region names must be empty.

Vertical Coordinates of the Top and Bottom of the Bounding Box of Materials and Regions

To extract the bottom coordinate of the bounding box of regions having any of the specified materials, use the command:

```
extract type=bounding_box_bottom [materials=<l>] [name=<c>] [silent=<c>] \
   [structure=<c>]
```

This command returns the vertical coordinate of the bottom of the bounding box of regions of the given structure that have any of the specified materials. The specified structure must be a boundary. If no materials are specified, the vertical coordinate of the bottom of the bounding box of the entire structure is returned. If the list of materials is not empty but contains only materials that are not present in the specified structure, an error is issued.

To extract the top coordinate of the bounding box of regions having any of the specified materials, use the command:

```
extract type=bounding_box_top [materials=<l>] [name=<c>] [silent=<c>] \
   [structure=<c>]
```

This command returns the vertical coordinate of the top of the bounding box of regions of the given structure that have any of the specified materials. The specified structure must be a boundary. If no materials are specified, the vertical coordinate of the top of the bounding box of the entire structure is returned. If the list of materials is not empty but contains only materials that are not present in the specified structure, an error is issued.

To extract the bottom coordinate of the bounding box of regions having any of the specified names, use the command:

```
extract type=bounding_box_bottom [name=<c>] [regions=<l>] [silent=<c>] \
   [structure=<c>]
```

This command returns the vertical coordinate of the bottom of the bounding box of regions of the given structure that have any of the specified region names. The specified structure must be a boundary. If no region names are specified, the vertical coordinate of the bottom of the bounding box of the entire structure is returned. If the list of region names is not empty but contains only regions that are not present in the specified structure, an error is issued.

To extract the top coordinate of the bounding box of regions having any of the specified names, use the command:

```
extract type=bounding_box_top [name=<c>] [regions=<l>] [silent=<c>] \
   [structure=<c>]
```

This command returns the vertical coordinate of the top of the bounding box of regions of the given structure that have any of the specified region names. The specified structure must be a boundary. If no region names are specified, the vertical coordinate of the top of the bounding box of the entire structure is returned. If the list of region names is not empty but contains only regions that are not present in the specified structure, an error is issued.

Other Properties

To extract other properties:

- Extraction of the dimension of a structure:

  ```
  extract type=dimension [name=<c>] [silent=<c>] [structure=<c>]
  ```

  Returns the dimension of a structure.

Table 40 Parameters of extract command

Parameter	Type	Description	Default value [Possible values]	Unit
axis	character	The direction of one of the coordinate axes.	none [x, y, z]	none
center	vector	Two-dimensional position of the vertical extraction axis.	none	μm
direction	vector	The direction of the cutting line.	none	none
exposed_only	character	Specifies whether only the exposed materials, regions, or region parts must be extracted.	false [false, true]	none
file	character	The name of the file in which the extracted information is saved.	basename_extract.tdr	none
material1	character	The name of a material.	none	none
material2	character	The name of a material.	none	none

Table 40 Parameters of extract command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
materials	list	The list of materials to consider for the extraction when any of the following are specified: type=bounding_box, type=bounding_box_bottom, type=bounding_box_top, type=probe. When using type=probe, any edge found along the specified line that lies in a region having a material not included in the list will be ignored and will not be included in the result. If materials is omitted, any material will be included in the output. **NOTE** When type=probe, you can specify materials only if property=length.	none	none
name	character	The name of the current extraction.	The extraction number	none
normal	vector	The normal of the specified plane.	none	none
output	character	A filtering parameter. The values inside and outside only apply if a cutting line is specified with the point1 and point2 parameters.	all [all, first, inside, last, outside]	none
point	vector	A point on the cutting line or the cutting plane.	none	µm
point_max	vector	The maximum corner point of the cuboid.	none	µm
point_min	vector	The minimum corner point of the cuboid.	none	µm
point1	vector	The first point on the cutting line or the cutting plane.	none	µm
point2	vector	A second point on the cutting line or the cutting plane.	none	µm
point3	vector	A third point on the cutting plane.	none	µm
position	number	The cutting plane distance from the origin measured in the direction defined by axis.	none	µm

Table 40 Parameters of extract command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
probe_empty_space	character	Specifies whether to probe empty space. **NOTE** You can specify this parameter only if property=length.	false [false, true]	none
property	character	A property to be associated with segments returned by type=probe.	material [length, material, region, region_part]	none
region	character	The name of the region for which properties are extracted.	none	none
region_1	character	The name of the region on the first side of the interface for which the area is extracted.	none	none
region_2	character	The name of the region on the second side of the interface for which the area is extracted.	none	none
region_part1	list	The name of a region and the name of a part of that region, given in the format: {region_name part_name}	none	none
region_part2	list	The name of a region and the name of a part of that region, given in the format: {region_name part_name}	none	none
region1	character	The name of a region.	none	none
region2	character	The name of a region.	none	none
regions	list	The list of regions to consider when any of the following are specified: type=bounding_box, type=bounding_box_bottom, type=bounding_box_top.	none	none
silent	character	When set to true, the extracted values are returned but are not written to the log file.	false [false, true]	none
structure	character	Sets the name of the structure whose properties must be extracted.	default_structure	none

Table 40 Parameters of extract command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
type	character	The extraction type. **NOTE** You can use type=cylinder_hole_profile only for structures resulting from simulations that used the PMC method.	none [1d_cut, 2d_cut, area, bottom_exposed, bounding_box, bounding_box_bottom, bounding_box_exposed, bounding_box_top, cylinder_hole_profile, dimension, interface, interface_area, material_name, material_names, part_area, part_volume, probe, region_names, region_parts, shortest_distance, slice, top_exposed, volume]	none

NOTE The parameter structure can specify a PMC structure only when type=bounding_box, or type=dimension, or type=interface is used to extract the interface between two materials (that is, when also parameters material1 and material2 are specified), or when type=probe and property=material, or when type=cylinder_hole_profile, or when type=probe and property=length. In other cases, to process a PMC structure, an explicit conversion of the structure is needed before calling the extract command (see Integration With Other Sentaurus Topography 3D Functionality on page 231).

The command extract can be used more than once in the same command file. The first time that the command extract is used with type=1d_cut, type=2d_cut, type=interface, type=probe, or type=slice, an extract file is created with the name specified with the parameter file. In subsequent uses of the command with the same value of the parameter file, the results are added to the previously computed and saved extractions. Each intersection has a name that is provided by the parameter name.

The parameter silent can be used with all extraction types to control whether the extracted values are written to the log file. When silent=true, the extracted values are not written to the log file, but they can be assigned to a Tcl variable.

The dimension of the point specified with point, point1, or point2 must match the dimension of the structure for which it is used.

Example of 1d_cut Extraction Type

```
extract axis=x point={-0.5 0.5 1.3} type=1d_cut
```

A line passing through the point (–0.5 0.5 1.3) and with a direction parallel to the x-axis is created and intersected with the closed surface. All the intersection points are written into a TDR file.

Example of the extraction of the third coordinate of the second vertex of the second edge of a 1D cut:

```
set 1dcut = [extract axis=z point={0.5 0.5 0} type=1d_cut]
set edge1 [lindex $1dcut 1]
set vertex1 [lindex $edge1 1]
set coord3 [lindex $vertex1 2]
```

Example of 2d_cut Extraction Type

```
extract axis=z position=0.5 type=2d_cut
```

A plane with normal parallel to the z-axis and intersecting it at z = 0.5 is created and intersected with the closed surface. The intersection points are saved in a TDR boundary file. Every intersection between the plane and the closed surface has a name that is provided by the parameter `name`.

NOTE If not specified explicitly by setting the parameter `file`, 1D and 2D extractions are saved in the same TDR file.

Example of area Extraction Type

```
extract type=area region=insulation
```

The surface area of the region named `insulation` is extracted, and the following output is created:

```
extract
 type = area
 Surface_area(insulation) = 2.24
extract done.
```

Example of bottom_exposed Extraction Type

```
extract type=bottom_exposed
```

The minimum vertical coordinate of the bounding box of the exposed surface is extracted and output:

```
extract
 type = bottom_exposed
```

```
 Bottom_exposed = -5e-07
extract done.
```

Example of bounding_box_exposed Extraction Type

```
extract type=bounding_box_exposed
```

A list containing the minimum and maximum vertices of the bounding box of the exposed surface is extracted and output:

```
extract
 type = bounding_box_exposed
 Bounding_box_exposed = {-1 -1 -0.5} {1 1 1}
extract done.
```

Example of dimension Extraction Type

```
extract type=dimension
```

The dimension of the processed structure is extracted and output. Here, the processed structure is three dimensional:

```
extract
 type = dimension
 Dimension = 3
extract done.
```

Example of interface Extraction Type

```
extract type=interface material1=Silicon material2=Oxide point={0.5 0.5 0} \
   direction={0 0 1}
```

The interfaces between silicon and oxide in the processed structure along the specified line are extracted and output. Here, there is only one interface between silicon and oxide along the specified line at the point of the coordinates `{0.5 0.5 0}`:

```
extract
 type = interface
 line type = 'direction - point'
 point = {0.5 0.5 0} [um].
 direction = {0 0 1}.
 material1 = 'Silicon'.
 material2 = 'Oxide'.
extract done.
```

Example of interface_area Extraction Type

```
extract type=interface_area region_1=substrate region_2=source_contact
```

The area formed by the interface between the regions named `substrate` and `source_contact` is extracted, and the following output is created:

```
extract
 type = interface_area
 Interface_area(source_contact/substrate) = 0.4
extract done.
```

The following commands demonstrate how the interface area can be extracted and assigned to a Tcl variable. Here, the output usually generated by the `extract` command is suppressed by setting the parameter `silent` to `true`:

```
set contact_area [extract type=interface_area region_1=substrate \
   region_2=source_contact silent=true]

puts "The source contact area is $contact_area"
```

Example of material_name Extraction Type

```
extract type=material_name region=insulation
```

The name of the material of the region named `insulation` is extracted and output. Here, the material of the region named `insulation` is oxide:

```
extract
 type = material_name
 Material(insulation) = Oxide
extract done.
```

Example of part_area Extraction Type

```
extract type=part_area region=insulation
```

The surface area of the individual parts of the region named `insulation` is extracted and output separately. Here, the region named `insulation` consists of two parts:

```
extract
 type = part_area
 Part_surface_area(insulation) = 1.192 1.048
extract done.
```

Example of part_volume Extraction Type

```
extract type=part_volume region=insulation
```

The volume of the individual parts of the region named `insulation` is extracted and output separately. Here, the region named `insulation` consists of two parts:

```
extract
 type = part_volume
 Part_volume(insulation) = 2.2 1.7
extract done.
```

Example of probe Extraction Type

```
extract type=probe point1={0 0 0} point2={0 0 1} property=material
```

The processed structure is intersected with a line passing through the points of the coordinates `{0 0 0}` and `{0 0 1}`.

Example of region_names Extraction Type

```
extract type=region_names
```

The name of each region of the processed structure is extracted and output. Here, the structure consists of two regions named `bulk` and `mask`:

```
extract
 type = region_names
 Region_names = bulk mask
extract done.
```

Example of region_parts Extraction Type

```
extract type=region_parts
```

The name of each part of each region of the processed structure is extracted and output. Here, the structure consists of two regions named `bulk` and `mask`. The region named `bulk` consists of a part named `0`, and the region named `mask` consists of two parts named `0` and `1`:

```
extract
 type = region_parts
 Region_parts = {bulk 0} {mask {0 1}}
extract done.
```

The parts of a specific region can be extracted by specifying the `region` parameter:

```
extract type=region_parts region=mask
```

The name of each part of the region named `mask` is extracted and output separately:

```
extract
 type = region_parts
 Region_parts(mask) = 0 1
extract done.
```

Example of shortest_distance Extraction Type

```
extract type=shortest_distance region_part1={mask 0} region_part2={mask 1}
```

The shortest distance between the parts named `0` and `1` of the region named `mask` is computed and returned along with the segments where the shortest distance occurs. The command returns a list whose first element is the requested shortest distance.

Example of the extraction of the shortest distance from the returned value:

```
set res [extract type=shortest_distance region_part1={mask 0} \
   region_part2={mask 1}]

set shortest_distance [lindex $res 0]
```

Example of slice Extraction Type

```
extract type=slice point={0 0 0} normal={0 1 0} file=slice.tdr
```

The processed structure is intersected with a plane having its normal direction parallel to the y-axis and passing through the point of the coordinates `{0 0 0}`. The 2D extracted boundary is saved in the TDR file named `slice.tdr`.

Example of top_exposed Extraction Type

```
extract type=top_exposed
```

The maximum vertical coordinate of the bounding box of the exposed surface is extracted and output:

```
extract
 type = top_exposed
 Top_exposed = 1e-06
extract done.
```

Example of volume Extraction Type

```
extract type=volume region=bulk
```

The volume of the region named `bulk` of the processed structure is extracted and output:

```
extract
 type = volume
 Volume(mask) = 2
extract done.
```

fill

The `fill` command is used to fill up a structure with a material up to a certain height, measured from the top of the existing structure. The result is similar to a deposition followed by a planarization step.

```
fill material=<c> thickness=<n> [comment=<c>] [merge=<c>] [region=<c>] \
   [structure=<c>]
```

Table 41 Parameters of fill command

Parameter	Type	Description	Default value [Possible values]	Unit
`comment`	character	Sets a comment for this process step that will be displayed or written to the log file.	empty string	none
`material`	character	Sets the material.	none	none
`merge`	character	If `true`, the deposited region merges with the already existing regions of the same material in the structure.	`false` `[false, true]`	none
`region`	character	Sets the name of the region that is deposited.	none	none
`structure`	character	Sets the name of the structure that must be filled.	`default_structure`	none
`thickness`	number	Sets the thickness, measured from the top of the structure.	none `[0, ∞ [`	µm

NOTE The parameter `structure` cannot specify a PMC structure. To process such a structure, an explicit conversion of the structure is needed before calling the `fill` command (see Integration With Other Sentaurus Topography 3D Functionality on page 231).

Examples

This command deposits a region consisting of material `Oxide`, with a thickness of 0.5 µm. The thickness is measured from the top of the existing structure. The comment given to the process step is “oxide fill” and is displayed in the log file. The region name is set to `filled_oxide`:

```
fill material=Oxide thickness=0.5 comment="oxide fill" region="filled_oxide"
```

filter_structure

The `filter_structure` command is used to perform different operations.

Execute Boolean operations:

```
filter_structure type=boolean [body=<c>] [deposit_material=<c>] [name=<c>] \
   [structure=<c>] [surface_accuracy=<n>] [surface_ridge_angle=<n>] \
   [surface_simplify=<c>]
```

Create a copy of an existing structure:

```
filter_structure type=copy [name=<c>] [structure=<c>]
```

Decimate the number of surface elements:

```
filter_structure type=decimate tolerance=<n> [mean_spacing=<n>] [name=<c>] \
   [structure=<c>]
```

Merge several regions that have the same material into one region:

```
filter_structure type=merge_regions new_region_name=<c> merged_regions=<l> \
   [name=<c>] [structure=<c>]
```

Create a structure with a different surface discretization:

```
filter_structure type=rediscretize_boundary [decimate=<c> [accuracy=<n>]] \
   [material_priority=<l>] [material_selection=<c>] [mean_spacing=<n>] \
   [structure=<c>]
```

Remove those parts from a structure that are completely surrounded by gas and, therefore, are disconnected topologically from the bulk (see Figure 21 on page 191):

```
filter_structure type=remove_disconnected_top_parts [name=<c>] [structure=<c>]
```

Remove those parts from a boundary structure that fit the specified criteria of material, maximum area or maximum volume, and bounding box:

```
filter_structure type=remove_parts [exposed_only=<c>] [materials=<l>] \
   [maximum_area=<n>] [maximum_volume=<n>] [name=<c>] \
   [point_max=<v>] [point_min=<v>] [silent=<c>] [structure=<c>]
```

Remove those parts from a PMC structure that fit the specified criterion of maximum volume:

```
filter_structure type=remove_parts maximum_volume=<n> [name=<c>] \
   [silent=<c>] [structure=<c>]
```

Remove a region from a boundary structure:

```
filter_structure type=remove_region region=<c> [name=<c>] [structure=<c>]
```

Rename a region:

```
filter_structure type=rename_region new_region_name=<c> region=<c> \
   [name=<c>] [structure=<c>]
```

Replace the material of a region in a boundary structure:

```
filter_structure type=replace_material region=<c> new_material=<c> \
   [name=<c>] [structure=<c>]
```

Replace a material in all matching regions of a boundary structure:

```
filter_structure type=replace_material material=<c> new_material=<c> \
   [name=<c>] [structure=<c>]
```

Smooth a structure:

```
filter_structure type=smooth [factor=<n>] [mean_spacing=<n>] [name=<c>] \
   [structure=<c>]
```

NOTE The `filter_structure` command with `type=smooth` is not available for 2D structures.

Create a boundary structure from a PMC structure (see Saving Boundaries for PMC Structures Using the VBE Method on page 204):

```
filter_structure type=vbe [decimate=<c> [accuracy=<n>]] \
   [material_priority=<l>] [material_selection=<c>] [structure=<c>]
```

Table 42 Parameters of filter_structure command

Parameter	Type	Description	Default value [Possible values]	Unit
`accuracy`	number	Specifies the maximum deformation during decimation of the boundary extracted with the volume boundary extraction (VBE) method relative to the mean PMC spacing.	`1e-3` `[0, ∞ [`	none
`body`	character	Sets the name of the structure to use as the body of the Boolean operation. If omitted, the structure specified with the parameter `structure` will be used as the body.	none	none
`decimate`	character	Specifies decimation of the boundary extracted with the VBE method.	`true` `[false, true]`	none

Table 42 Parameters of filter_structure command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
deposit_material	character	The material to assign to all the deposited regions of a structure obtained from a PMC simulation when executing Boolean operations.	When processing the result of a PMC simulation where a single material was deposited and deposit_material was not specified, that material is used as the deposited material for Boolean operations. If multiple materials were deposited in a PMC simulation, then Anymaterial is used as the deposited material for Boolean operations if deposit_material was not specified. If deposit_material is specified, that material is used as the deposited material for Boolean operations.	none
exposed_only	character	Specifies whether only exposed parts should be removed.	true [false, true]	none
factor	number	Specifies how strong the smoothing is relative to the mean spacing of the last preceding etching or deposition step. Higher values cause stronger smoothing.	1 [0.2, 5]	none
material	character	Specifies the material to replace. At least one region with the specified material must exist.	none	none
material_priority	list	Specifies a material priority list for materials at interfaces when using the VBE method.	none	none
material_selection	character	Specifies the algorithm used to select the materials of a PMC cell when using the VBE method (see Saving Boundaries for PMC Structures Using the VBE Method on page 204).	mixed [maximum, mixed, proportional]	none
materials	list	Specifies the materials of the parts to remove. If omitted, a part will be removed if it matches the other specified criteria, independently of its material.	none	none
maximum_area	number	Specifies the maximum area of the parts to remove. If omitted, a part will be removed if it matches the other specified criteria, independently of its area.	[0, ∞ [	μm^2

Table 42 Parameters of filter_structure command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
maximum_volume	number	Specifies the maximum volume of the parts to remove. If omitted, a part will be removed if it matches the other specified criteria, independently of its volume. This parameter is mandatory when processing PMC structures.	[0, ∞ [	μm^3
mean_spacing	number	Specifies the mean spacing relative to which the decimation or smoothing is specified.	Mean spacing of the last preceding etching or deposition step. [0, ∞ [	µm
merged_regions	list	Specifies the names of regions to be merged.	none	none
name	character	Sets the name of the result structure. If omitted, the input structure will be replaced.	none	none
new_material	character	Specifies the name of the replacement material.	none	none
new_region_name	character	When renaming regions, this parameter specifies the new name for a region. No region with this name must exist before the renaming. When merging regions, this parameter specifies the name of the region created by merging.	none	none
point_max	vector	Specifies the maximum bounding vertex of the cuboid to which the parts to remove belong. If omitted, a part will be removed if it matches the other specified criteria, independently of its position.	none	µm
point_min	vector	Specifies the minimum bounding vertex of the cuboid to which the parts to remove belong. If omitted, a part will be removed if it matches the other specified criteria, independently of its position.	none	µm
region	character	Specifies the name of a region to remove or for which region the material will be replaced, or the name of a region to be renamed. The specified region must exist and, when removing a region, this region must not be the only region of the structure.	none	none
silent	character	Specifies whether to write information about the parts being removed to the log file.	false [false, true]	none

Table 42 Parameters of filter_structure command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
structure	character	Sets the name of the structure that must be used for the Boolean operations (when type=boolean), or copied (when type=copy), or decimated (when type=decimate), or smoothed (when type=smooth), or from which the disconnected parts must be removed (when type=remove_disconnected_top_parts).	default_structure	none
surface_accuracy	number	The accuracy to use to simplify the exposed surface of a structure obtained from a PMC simulation before executing Boolean operations. When set to -1, the actual accuracy will be determined based on the simulation grid size.	-1 {-1} ∪ [0, ∞[	µm
surface_ridge_angle	number	The ridge angle to use to simplify the exposed surface of a structure obtained from a PMC simulation before executing Boolean operations.	176 [0, 180]	degree
surface_simplify	character	Specifies whether the exposed surface of a structure obtained from a PMC simulation should be simplified before executing Boolean operations.	false [false, true]	none
tolerance	number	Specifies the tolerance relative to the mean spacing for decimating the number of surface elements. Higher values cause stronger decimation.	0 [0, 1]	none
type	character	Specifies the type of operation.	none [boolean, copy, decimate, merge_regions, rediscretize_boundary, remove_disconnected_top_parts, remove_parts, remove_region, rename_region, replace_material, smooth, vbe]	none

NOTE When the parameter type is set to decimate, merge_regions, rediscretize_boundary, remove_disconnected_top_parts, remove_region, rename_region, replace_material, or smooth, the parameter structure cannot specify a PMC structure. To process such a structure, an explicit conversion of the structure is needed before calling the filter_structure command (see Integration With Other Sentaurus Topography 3D Functionality on page 231).

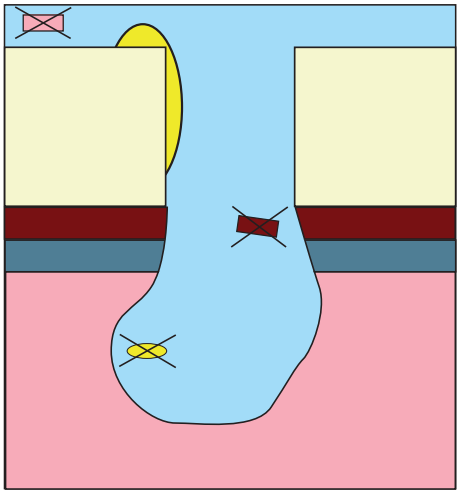


Figure 21 Using the filter_structure type=remove_disconnected_top_parts command removes the three cross-out parts

The default value of the mean spacing is calculated from the last preceding `etch` or `deposit` command. If there is no such command, the mean spacing must be specified with the parameter `mean_spacing`.

Creating a boundary structure from a PMC structure is very similar to saving a boundary structure from a PMC structure except that a new structure is created instead of saving it to a TDR file. For more information about the VBE method, see Saving Boundaries for PMC Structures Using the VBE Method on page 204.

finalize_model

The `finalize_model` command is used to indicate that the definition of the specified model is completed.

```
finalize_model model=<c>
```

Table 43 Parameters of finalize_model command

Parameter	Type	Description	Default value [Possible values]	Unit
`model`	character	The name of the model that is finalized.	none	none

Following this command, the specified model cannot be modified using the command `add_float_parameter`, `add_formula`, `add_int_parameter`, `add_reaction`, or `add_source_species`. Before using the command `add_flux_properties`, `add_reaction_properties`, `define_deposit_machine`, or `define_etch_machine` for the specified model, you must specify the `finalize_model` command.

let

The `let` command is used to set the global properties of the simulator, such as logging verbosity and parallelization options. The command `let` is followed by one parameter:

```
let parameter=<type>
```

where `parameter` can be any of the parameters listed in Table 44.

Table 44 Parameters of let command

Parameter	Type	Description	Default value [Possible values]	Unit
`base_name`	character	Modifies the basename for the output files (see Output Files on page 2).	none	none
`keep_parallel_licenses`	character	Specifies whether parallel licenses must remain checked out after a command using them has been executed.	`false` `[false, true]`	none
`log_file`	character	Sets the name of the log file.	`base_name.log`	none
`log_level`	character	Sets the verbosity level.	`info` `[all, error, info]`	none
`log_target`	character	Sets the log output.	`both` `[both, console, file]`	none
`num_threads`	number	Sets the maximum number of threads that should be used. Only valid if: `let parallel=true`. If no value is specified, an implementation-defined optimum number of threads will be used (see Advanced Parallelization Options on page 24).	`Automatic` `[1...256]`	none
`parallel`	character	Switches parallel computation on or off for models that support parallel computation.	`false` `[false, true]`	none
`parallel_license`	character	Defines how the simulator behaves if the number of requested threads cannot be used due to a lack of the respective number of parallel licenses.	`serial` `[abort, available, serial, wait]`	none
`snmesh_repair`	character	Enables or disables the repair of the structure after Boolean operations. Enabling repair reduces very thin regions and facilitates meshing of the structure. **NOTE** The repair functionality sometimes uses excessive CPU time and memory.	`false` `[false, true]`	none

Examples

```
let log_level=info

let log_target=both

let log_file=out.log
```

The verbosity level is set to `info`; the output is directed to both the screen and a file; the log file name is set to `out.log`.

The parameters `parallel`, `parallel_license`, and `num_threads` refer to parallelization and are explained in more detail in Parallelization on page 24.

litho

The `litho` command performs a lithography simulation and adds a resist region.

NOTE The `litho` command is not available for 2D structures.

```
litho inputfile=<c> [comment=<c>] [machine=<c>] [mask=<c>] [material=<c>] \
   [merge<c>] [options=<c>] [outputfile=<c>] [region=<c>] [structure=<c>]
```

Table 45 Parameters of litho command

Parameter	Type	Description	Default value [Possible values]	Unit
`comment`	character	Sets a comment for this process step that will be displayed or written to the log file.	empty string	none
`inputfile`	character	The name of the input SLO file.	none	none
`machine`	character	The name of the lithography machine to be used. This must be predefined with the `define_litho_machine` command.	`default_machine`	none
`mask`	character	The name of the mask file to be used. **NOTE** If this parameter is not specified, the mask specified in the SLO file is used for the lithography simulation.	none	none
`material`	character	The name of the material of the created region.	`Photoresist`	none
`merge`	character	If `true`, the deposited region merges with the already existing regions of the same material in the structure.	`false` `[false, true]`	none
`options`	character	Specify additional command-line options for Sentaurus Lithography. **NOTE** If you specify more than one option, they must be enclosed in double quotation marks.	none	none
`outputfile`	character	The name of the output SLO file.	none	none
`region`	character	The name of the resist region that is created.	none	none
`structure`	character	Sets the name of the structure on which a lithography simulation must be run.	`default_structure`	none

NOTE The parameter `structure` cannot specify a PMC structure. To process such a structure, an explicit conversion of the structure is needed before calling the `litho` command (see Integration With Other Sentaurus Topography 3D Functionality on page 231).

pattern

The `pattern` command uses a mask defined in the `define_mask` command to create patterned profiles on top of an existing structure.

NOTE The `pattern` command is not available for 2D structures.

```
pattern material=<c> thickness=<n> [comment=<c>] [mask=<c>] [merge=<c>] \
   [region=<c>] [structure=<c>]
```

Table 46 Parameters of pattern command

Parameter	Type	Description	Default value [Possible values]	Unit
`comment`	character	Sets a comment for this process step that will be displayed or written to the log file.	empty string	none
`mask`	character	The name of the mask defined in a `define_mask` command.	`default_mask`	none
`material`	character	The material of the pattern that is deposited on the structure.	none	none
`merge`	character	If `true`, the deposited region merges with the already existing regions of the same material in the structure.	`false` `[false, true]`	none
`region`	character	The region name of the pattern that is created.	none	none
`structure`	character	Sets the name of the structure that must be patterned.	`default_structure`	none
`thickness`	number	The thickness of the pattern, measured from the top of the structure.	none `[0, ∞ [`	µm

NOTE The parameter `structure` cannot specify a PMC structure. To process such a structure, an explicit conversion of the structure is needed before calling the `pattern` command (see Integration With Other Sentaurus Topography 3D Functionality on page 231).

Examples

This command uses `default_mask` to pattern a film consisting of silicon material, with a thickness of 0.1 µm , measured from the top of the current structure:

```
pattern material=Silicon thickness=0.1
```

Example using a mask and a region name:

```
pattern material=Silicon thickness=0.1 mask=mask_1 comment="pattern mask 1" \
   region=mask_1_pattern
```

This command uses the mask `mask_1` to pattern a film consisting of silicon material, with a thickness of 0.1 µm , measured from the top of the device. The patterned region is given the name `mask_1_pattern`. The comment for the process step is “pattern mask 1” and is displayed in the log file.

remove_material

The `remove_material` command removes material from a structure.

```
remove_material material=<c> [comment=<c>] [exposed_only=<c>] [structure=<c>]
```

Table 47 Parameters of remove_material command

Parameter	Type	Description	Default value [Possible values]	Unit
`comment`	character	Sets a comment for this process step that will be displayed or written to the log file.	empty string	none
`exposed_only`	character	Specifies whether only the exposed material should be removed or all materials.	`true` `[false, true]`	none
`material`	character	The material to be removed from the structure.	none	none
`structure`	character	Sets the name of the structure from which a material must be removed.	`default_structure`	none

Examples

This command removes all of the exposed silicon material from the structure. Silicon that is completely covered by other material will not be removed. The comment for this process step is “strip silicon” and is displayed in the log file:

```
remove_material material=Silicon comment="strip silicon"
```

save

The `save` command saves a boundary structure to a TDR file:

```
save [type=final_structure] [add_interfaces=<c>] [file=<c>] [structure=<c>] \
   [tdr_geometry=<c>]
```

The `save` command extracts the exposed surface of a boundary structure and saves it to a TDR file:

```
save type=exposed_surface [file=<c>] [structure=<c>] [tdr_geometry=<c>]
```

The `save` command extracts the closed exposed surface of a boundary structure and saves it to a TDR file:

```
save type=closed_surface [file=<c>] [structure=<c>] [tdr_geometry=<c>]
```

The `save` command saves a 3D GC structure to a TDR file:

```
save type=gc [gc_samples_per_cell=<n>] [structure=<c>] [tdr_geometry=<c>]
```

The `save` command extracts the boundary of a PMC structure and saves it to a TDR file:

```
save type=vbe [add_interfaces=<c>] [decimate=<c> [accuracy=<n>]] [file=<c>] \
   [material_priority=<l>] [material_selection=<c>] [structure=<c>] \
   [tdr_geometry=<c>]
```

The `save` command saves a PMC structure to a PMC file:

```
save type=pmc [file=<c>] [initial_structure=<c>] [structure=<c>]
```

The `save` command saves IADs and probability, reflection, and yield functions to a TDR file:

```
save iad=<c> [file=<c>] [tdr_geometry=<c>]

save probability=<c> [angle_samples=<n>] [energy_max=<n>] [energy_min=<n>] \
   [energy_samples=<n>] [file=<c>]

save reflection=<c> [angle_samples=<n>] [energy_max=<n>] [energy_min=<n>] \
   [energy_samples=<n>] [file=<c>]

save yield=<c> [angle_samples=<n>] [energy_max=<n>] [energy_min=<n>] \
   [energy_samples=<n>] [file=<c>]
```

Table 48 Parameters of save command

Parameter	Type	Description	Default value [Possible values]	Unit
accuracy	number	Specifies the maximum deformation during decimation of the boundary extracted with the VBE method relative to the mean PMC spacing.	1e-3 [0, ∞ [	none
add_interfaces	character	When activated, interface regions for the interface of bulk regions that touch each other are added.	false [false, true]	none
angle_samples	number	The number of angular samples to save.	90 [2, 2147483647]	none
decimate	character	Specifies decimation of the boundary extracted with the VBE method.	true [false, true]	none
energy_max	number	The upper end of the energy window to save.	0 [0, ∞ [	eV
energy_min	number	The lower end of the energy window to save.	0 [0, ∞ [	eV
energy_samples	number	The number of energy samples to save.	100 [2, 2147483647]	none
file	character	Path of the TDR file or PMC file where the structure is saved. In the file name, <base_name> denotes the value of the base_name parameter of the let command.	<base_name>.tdr <base_name>.pmc (only when type=pmc)	none
gc_samples_per_cell	number	Specifies the number of samples to take along the x- and y-directions per grid cell.	1 [1, 2147483647]	none
iad	character	The name of the IADF to be saved.	none	none
initial_structure	character	The name of the initial structure that is saved in the PMC file and can be used as the body for Boolean operations.	none	none
material_priority	list	Specifies a material priority list for materials at interfaces when using the VBE method.	none	none
material_selection	character	Specifies the algorithm used to select the materials of a PMC cell when using the VBE method (see Saving Boundaries for PMC Structures Using the VBE Method on page 204).	mixed [maximum, mixed, proportional]	none
probability	character	Specifies the name of the probability function to save.	none	none

Table 48 Parameters of save command (Continued)

Parameter	Type	Description	Default value [Possible values]	Unit
reflection	character	Specifies the name of the reflection function to save.	none	none
structure	character	The name of the structure to save.	default_structure	none
tdr_geometry	character	The name of the TDR geometry to be saved.	none	none
type	character	Specifies the type of data to save.	final_structure [closed_surface, exposed_surface, final_structure, gc, pmc, vbe]	none
yield	character	Specifies the name of the yield function to save.	none	none

NOTE When type=closed_surface, or type=exposed_surface, or type=final_structure, the parameter structure cannot specify a PMC structure. To process such a structure, an explicit conversion of the structure is needed before calling the save command (see Integration With Other Sentaurus Topography 3D Functionality on page 231).

When type=gc, type=pmc, or type=vbe, the parameter structure must specify a PMC structure.

Saving Structures

As described in Boundary Types on page 13, the following boundary types can be saved:

- Closed surface (type=closed_surface)
- Exposed surface (type=exposed_surface)
- Final structure (type=final_structure)

By default, the save command saves the final structure.

NOTE See Integration With Other Sentaurus Topography 3D Functionality on page 231 for the limitations when saving a PMC structure.

The command save can be used more than once in the same command file. The first time that a file name (default name or user-defined name using the parameter file) is used, a new TDR file with that name is created or overwritten in the case where the file already exists from a previous simulation.

In subsequent uses of the `save` command that refer to the same file name, the results are appended to the previously saved structures in that file independently of the saved type or of the simulation step that preceded the `save` command. In other words, a file that is created within a simulation is never overwritten, and data is always appended to it.

You can save the resulting structure into a different file by creating a new file with the parameter `file`.

Examples

```
save
save type=closed_surface
```

The first `save` command saves the final structure into the default TDR file. The second `save` command adds the corresponding closed surface to the same file.

```
save file="mosfet.tdr" tdr_geometry="pmos"
```

This command saves the current structure to the TDR file `mosfet.tdr` and gives the created TDR geometry the name `pmos`. The name can be reused to load this specific geometry in the `define_structure` command, for example.

Saving Ion-Angular Distribution Functions

The `save` command can save ion-angular distribution functions (IADFs) to TDR files.

The following command saves the IADF named `iad_1` to the default TDR file:

```
save iad=iad_1
```

The following command saves the IADF `iad_1` to the file `iad1.tdr`, and it names the IADF in the file `ion_iad`:

```
save iad=iad_1 file=iad1.tdr tdr_geometry=ion_iad
```

Saving PMC Structures

A PMC structure is stored in a PMC file with the `save` command when `type=pmc`.

The advantage of using PMC files to create splits in a process flow, compared to TDR files containing boundaries, is that no conversion to a boundary and back to the PMC representation is required. Therefore, loading and saving is more efficient and does not introduce artifacts.

To perform Boolean operations and to create a boundary from a PMC structure that has been loaded from a PMC file, the initial structure from which the PMC structure has been created is required. The parameter `initial_structure` is used to specify the initial structure that is stored in the PMC file.

NOTE A PMC file is not a TDR file and cannot be visualized with Sentaurus Visual. It can only be loaded in to Sentaurus Topography 3D.

Examples

```
etch time=0.5 spacing=0.05 method=pmc machine=m_100
save file=pmc_only.pmc type=pmc
```

The first command etches a structure using the PMC method. The second command saves the PMC structure to a PMC file without saving the initial structure to the PMC file. Therefore, this PMC file can be used only as a starting point for other PMC simulations. It is not possible to perform Boolean operations or to convert the PMC structure to a boundary.

```
define_structure pmc_file=pmc_only.pmc
etch time=0.5 spacing=0.05 method=pmc machine=m_100
```

Here, the previously saved PMC structure is loaded from a PMC file and used for a PMC simulation.

```
define_structure material=Silicon point_min={0 0 0} point_max={1 1 1}
filter_structure type=copy name=sc
etch time=0.5 spacing=0.05 method=pmc machine=m_100
save file=pmc_and_initial.pmc type=pmc initial_structure=sc
```

In this example, an initial structure is created and copied to a structure named `sc`. After etching the initial structure, it is saved to a PMC file together with the boundary of the copy of the initial structure.

```
define_structure pmc_file=pmc_and_initial.pmc
etch time=0.5 spacing=0.05 method=pmc machine=m_100
filter_structure type=boolean body=sc
save file=f.tdr
```

In the first command, the PMC structure and the initial structure are loaded from a PMC file. After etching the PMC structure, Boolean operations are performed with the initial structure named `sc` loaded from the PMC file. Finally, the structure is saved as a boundary to a TDR file.

If the name of the initial structure stored in the PMC file is unknown, a new name can be specified with the parameter `initial_structure_name` as demonstrated in the following example:

```
define_structure pmc_file=pmc_and_initial.pmc initial_structure_name=xyz
etch time=0.5 spacing=0.05 method=pmc machine=m_100
```

```
filter_structure type=boolean body=xyz
save file=f.tdr
```

Saving Boundaries for PMC Structures Using the VBE Method

To save a boundary structure for a PMC structure that has been created from a boundary structure using a PMC etch or deposition model, you can create a new boundary structure using `filter_structure type=boolean` and the initial boundary structure. The advantage of this method is that artifacts are limited to the parts of the structure that were modified by process steps. However, this method is computationally expensive.

An alternative method for saving a boundary structure is the volume boundary extraction (VBE) method that extracts a boundary structure from a PMC structure without using the initial boundary structure. The advantage of this method is that it is computationally more efficient than the previously described method. However, conversion artifacts might be created throughout the entire structure, not only in the parts modified by process steps.

The VBE method can also be used to save boundary structures for structures that were initially created as PMC structures.

Converting a PMC structure to a boundary structure using the VBE method might introduce artifacts at material interfaces due to the different discretization methods. The parameters `material_priority` and `material_selection` provide some control over the selection of materials at the interface of different materials.

When using the material selection algorithm:

- If `material_selection=maximum`, the entire volume of a PMC cell is considered to be of the material that has the maximum partial volume.
- If `material_selection=proportional`, each material in a PMC cell is initially represented proportionally to its partial volume. Because the final result also depends on the materials in neighboring cells, materials for which the partial volume is less than 50% might be replaced by a different material.
- If `material_selection=mixed`, the effect is similar to `proportional`, but if the partial volume of the material with the highest partial volume in a cell is less than 50%, the partial volume is increased to 50% to ensure that this material is not replaced by a different material in the final result.

When two materials have the same partial volume in a cell, preference is given to the material with the higher priority. The parameter `material_priority` can specify materials according to their priority. The material listed first has the highest priority. Materials that are not in the list have lower priority than any of the listed materials.

set_orientation

The `set_orientation` command is used to set or change the crystal orientation of a region.

```
set_orientation flat_orientation=<v> region=<c> vertical_orientation=<v> \
   [structure=<c>]
```

Table 49 Parameters of set_orientation command

Parameter	Type	Description	Default value [Possible values]	Unit
`flat_orientation`	vector	Specifies the Miller index of the direction perpendicular to the wafer flat (y-axis of the wafer coordinate system) for this region.	none	none
`region`	character	Specifies the name of the region for which the crystal orientation is set.	none	none
`structure`	character	Sets the name of the structure whose crystal orientation must be set.	`default_structure`	none
`vertical_ orientation`	vector	Specifies the Miller index of the direction perpendicular to the wafer surface (z-axis of the wafer coordinate system) for this region.	none	none

truncate

The `truncate` command truncates a structure, cutting away everything that lies outside of a cuboid bounding box.

```
truncate point_min=<v> point_max=<v> [accuracy=<n>] [comment=<c>]
   [decimate=<c>] [structure=<c>]
```

Table 50 Parameters of truncate command

Parameter	Type	Description	Default value [Possible values]	Unit
`accuracy`	number	The maximum-allowed deformation when decimating the boundary. If this value is negative, the value actually used is determined automatically based on the structure size.	`-1` `{-1} ∪ [0, ∞[`	µm
`comment`	character	Sets a comment for this process step that will be displayed or written to the log file.	empty string	none
`decimate`	character	Specifies whether to decimate the truncated boundary structure.	`true` `[false, true]`	none
`point_max`	vector	The maximum corner point of the bounding cube.	none	µm
`point_min`	vector	The minimum corner point of the bounding cube.	none	µm
`structure`	character	Sets the name of the structure that must be truncated.	`default_structure`	none

NOTE The parameter `structure` cannot specify a PMC structure. To process such a structure, an explicit conversion of the structure is needed before calling the `truncate` command (see Integration With Other Sentaurus Topography 3D Functionality on page 231).

Examples

This command truncates the structure, removing all parts of the structure that lie outside the cuboid bounding box with corners `{0 0 0}` `{1 1 1}`. The comment for this process step is “truncation” and is displayed in the log file:

```
truncate point_min={0 0 0} point_max={1 1 1} comment="truncation"
```

References

[1] T. Mizuno *et al*., "Analytical Model for Oblique Ion Reflection at the Si Surface," *IEEE Transactions on Electron Devices*, vol. 35, no. 12, pp. 2323–2327, 1988.

[2] D. Zhang and M. J. Kushner, "Surface kinetics and plasma equipment model for Si etching by fluorocarbon plasmas," *Journal of Applied Physics*, vol. 87, no. 3, pp. 1060–1069, 2000.

[3] R. Akolkar and U. Landau, "Mechanistic Analysis of the "Bottom-Up" Fill in Copper Interconnect Metallization," *Journal of the Electrochemical Society*, vol. 156, no. 9, pp. D351–D359, 2009.

[4] J. A. Sethian and Y. Shan, "Solving partial differential equations on irregular domains with moving interfaces, with applications to superconformal electrodeposition in semiconductor manufacturing," *Journal of Computational Physics*, vol. 227, no. 13, pp. 6411–6447, 2008.

[5] M. J. Kushner, "Hybrid modelling of low temperature plasmas for fundamental investigations and equipment design," *Journal of Physics D: Applied Physics*, vol. 42, no. 19, p. 194013, 2009.

[6] T. K. Chini *et al*., "The angular dependence of sputtering yields of Ge and Ag," *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, vol. 72, no. 3–4, pp. 355–358, 1992.

[7] R. Courant, K. Friedrichs, and H. Lewy, "On the Partial Difference Equations of Mathematical Physics," *IBM Journal*, vol. 11, no. 2, pp. 215–234, 1967.

CHAPTER 5 Integration With Sentaurus Process and Sentaurus Interconnect

This chapter describes the integration of Sentaurus Topography 3D functionality into Sentaurus Process and Sentaurus Interconnect. The integration allows for advanced etching and deposition modeling in these tools in a user-friendly way.

Introduction

The Sentaurus Process and Sentaurus Interconnect executables support a subset of Sentaurus Topography 3D functionality for advanced etching and deposition modeling. This integration enables the users of Sentaurus Process and Sentaurus Interconnect to easily incorporate one or more etching and deposition process steps into their simulation flow without having to create a separate command file and simulation node for Sentaurus Topography 3D.

Not all commands of Sentaurus Topography 3D are supported in Sentaurus Process and Sentaurus Interconnect, however.

NOTE For brevity, in this chapter, whenever *Sentaurus Process* is mentioned, it implicitly means *Sentaurus Process and Sentaurus Interconnect*.

Supported Commands and Syntax

Sentaurus Topography 3D functionality is accessible using the `topo` command in Sentaurus Process. The `topo` command acts as a prefix and branches into a subset of Sentaurus Topography 3D commands. In general, the `topo` command is followed by the Sentaurus Topography 3D subcommand.

For example, this command calls the Sentaurus Topography 3D `etch` command from Sentaurus Process (the parameters are the same as for the original `etch` command):

```
topo etch spacing= {0.1} time=1.0
```

The syntax of the Sentaurus Topography 3D subcommands is the same as in Sentaurus Topography 3D. For the syntax of the commands, see Chapter 4 on page 65.

Table 51 Sentaurus Topography 3D commands supported from within Sentaurus Process

Sentaurus Topography 3D command	Sentaurus Process command
add_float_parameter	topo add_float_parameter
add_flux_properties	topo add_flux_properties
add_formula	topo add_formula
add_int_parameter	topo add_int_parameter
add_ion_flux	topo add_ion_flux
add_material	topo add_material
add_neutral_flux	topo add_neutral_flux
define_deposit_machine	topo define_deposit_machine
define_etch_machine	topo define_etch_machine
define_iad	topo define_iad
define_model	topo define_model
define_reflection	topo define_reflection
define_yield	topo define_yield
deposit	topo deposit
etch	topo etch
finalize_model	topo finalize_model

Additional Syntax

The `topo` command and Sentaurus Topography 3D subcommands in Sentaurus Process support specific parameters that are exclusive to Sentaurus Process.

The info Parameter

The `info` parameter sets the verbosity of the output of the command. See the *Sentaurus™ Process User Guide* for usage details.

For example, this command calls the `etch` command from within Sentaurus Process with a verbosity level of 3:

```
topo etch spacing= {0.1} time=1.0 info=3
```

The parameters Parameter

The `topo` command can be called with the Boolean flag `parameters` to print a list of all the supported subcommands and their syntax.

For example, this command prints the syntax of all the supported Sentaurus Topography 3D subcommands:

```
topo parameters
```

The repair Parameter

The `topo etch` and `topo deposit` subcommands support a Boolean flag in standard Sentaurus Process syntax to switch on boundary repair for etching and deposition steps. By default, boundary repair is switched on. See the *Sentaurus™ Process User Guide* for a description of the `repair` parameter.

For example, this command calls the `topo deposit` command with repair deactivated:

```
topo deposit spacing= {0.1} time=1.0 !repair
```

Different Default Behavior

To ensure consistency in the behavior of the Sentaurus Topography 3D commands when they are called from within Sentaurus Process, some commands can have different default behavior. These commands will behave slightly differently when called from Sentaurus Topography 3D or from within Sentaurus Process.

Repair Behavior

The repair behavior differs as follows:

- Sentaurus Topography 3D default behavior: Boundary repair is deactivated because it can use excessive CPU time and memory. Repair can be activated by using the command:

  ```
  let snmesh_repair=true
  ```

- Sentaurus Process default behavior: Boundary repair for the `topo etch` and `topo deposit` commands is activated (see The repair Parameter).

Merge Behavior

Both when calling the `topo` command from Sentaurus Process and when using Sentaurus Topography 3D directly, regions with the same materials are not merged by default.

This behavior is different from deposition using the MGOALS module in Sentaurus Process, where regions merge by default.

To activate region merging in the `topo` command, use the `merge` parameter in the `deposit` and `etch` commands (see deposit on page 147 and etch on page 154).

Parallelization

Parallelization differs as follows:

- In standalone Sentaurus Topography 3D, the commands `let parallel` and `let num_threads` are used to activate parallelization and to control the number of threads, respectively (see let on page 193).
- In Sentaurus Process, parallelization is controlled by the `math` command. The `topo` command uses the number of threads defined with the `math` command. See the *Sentaurus™ Process User Guide* for details about parallelization.

Limitations and Known Issues

There are limitations when using Sentaurus Topography 3D from Sentaurus Process.

Boundary Conditions

Boundary conditions in Sentaurus Topography 3D are set using the `define_boundary_conditions` command, which is not a supported subcommand of the `topo` command. Therefore, only the default boundary conditions can be used from within Sentaurus Process.

Meshing of Thin Layers

In standalone mode, Sentaurus Topography 3D creates boundary files as a result of process steps. Sentaurus Process, however, needs 3D meshes for most of its functionality. Boundaries are transferred from the `topo` command to Sentaurus Process automatically, and they are meshed when needed using Sentaurus Process commands such as `implant`.

Meshing of very thin material layers can be difficult. Meshing can either take a very long time or abort if the structure cannot be meshed by Sentaurus Mesh.

Sentaurus Topography 3D often generates thin layers, especially in models where redeposition effects are modeled (for example, when using the etchdepo model).

CHAPTER 6 Rate Formula Module

This chapter describes the rate formula module.

Overview

The rate formula module (RFM) enables the definition of deposition and etching models in Sentaurus Topography 3D. In contrast to the physical model interface (PMI), which requires C++ programming, the RFM uses only built-in commands. Therefore, the RFM simplifies the development of new models and provides greater flexibility in modeling.

You can specify the deposition and etching rate by writing a mathematical formula. This formula can involve fluxes, geometric quantities of the surface, user-defined parameters, and mathematical functions. It is used in each time step as described in Sentaurus Topography 3D Computational Model on page 5.

There are three distinct stages when using an RFM model, as illustrated in Figure 22 on page 216 and described in the following sections:

- Model definition
  (see Model Definition on page 216 and Model Definition Details on page 217)
- Machine configuration
  (see Machine Configuration on page 217 and Machine Configuration Details on page 220)
- Machine use (see Machine Use on page 217)

Detailed descriptions of the first two stages are given in the following sections. For examples of how to create models using the RFM, see the Sentaurus Topography 3D module in the TCAD Sentaurus Tutorial available from Sentaurus Workbench (choose **Help > Training**).

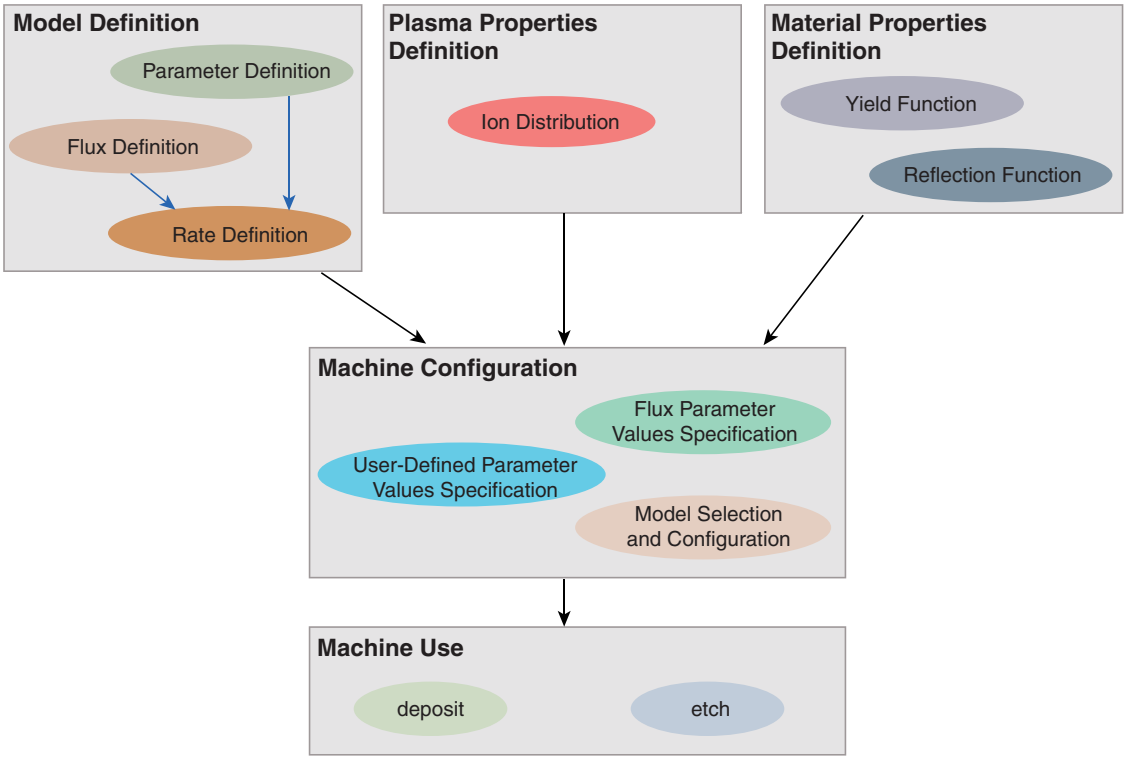


Figure 22 RFM modeling flow

Model Definition

First, the model must be defined by specifying the fluxes used by the RFM model, the global and material-specific parameters, and the formulas for the etching or deposition rate. This stage is analogous to the creation of a PMI model where the new model is defined in the form of C++ code. Since there is only a small number of commands for defining a new model, it is much easier to learn how to create an RFM model than a PMI model.

You can add an arbitrary number of ion and neutral fluxes to an RFM model. The ion fluxes can be configured individually to take the physical effects of reflection, sputtering, and reemission into account. Modeling of Fluxes and Related Physical Effects on page 27 describes these physical effects and how they are calculated.

In general, deposition and etching models have parameters that are used to adjust the deposition and etching rates to specific process conditions. An arbitrary number of material-specific and global parameters can be added to an RFM model. During the machine configuration, the values of these parameters can be set in the command file in the same way as for built-in and PMI models.

At each time step, Sentaurus Topography 3D calculates the rate of each surface element. For an RFM model, the rate is calculated from a user-defined mathematical expression. This expression can contain the values of the previously defined fluxes, the values of user-defined parameters, mathematical functions, and arithmetic operators.

Machine Configuration

After the definition of a new model, a machine that uses the newly defined model can be defined. This is very similar to setting up a machine using a built-in or a PMI model. However, the specification of model parameters related to ion or neutral fluxes must be performed with separate commands because an RFM model can use an arbitrary number of fluxes, and the reflection and yield functions must be configured separately.

Machine Use

Finally, the machine is used with an `etch` or a `deposit` command. This is performed in exactly the same way as with machines that use built-in or PMI models (see deposit on page 147 and etch on page 154).

Model Definition Details

As previously mentioned, the first stage of the modeling flow with the RFM is the definition of a model, which specifies the rate formula. The definition of a model consists of five steps, two of which are optional:

1. The definition of each model starts with the `define_model` command, which provides the name and the type of the model (`deposit`, `etch`, or `etchdepo`), as well as a short description of the model.
2. (Optional) For models that depend on neutral and ion fluxes, these are added with the `add_neutral_flux` and `add_ion_flux` commands, respectively.
3. (Optional) The floating-point and integer parameters used in the rate formula are added with the `add_float_parameter` and `add_int_parameter` commands, respectively.
4. A rate formula must be specified as a mathematical expression with the `add_formula` command.
5. The definition of the model is marked as complete by specifying the `finalize_model` command.

Details of Steps 2–4 are discussed in the following sections.

Flux Definition

This step introduces the ion and neutral fluxes involved in the model and states which physical effects are taken into account for ion fluxes. For neutral fluxes, reemission is always enabled, as discussed in Modeling of Fluxes and Related Physical Effects on page 27.

Neutral and ion fluxes are introduced by the `add_neutral_flux` and `add_ion_flux` commands, respectively, and the definition of each flux requires a unique identifier for the flux using the parameter `name` as well as the model to which the flux refers.

Accordingly, a neutral flux called `N` to be used in the model `my_model` is introduced with the statement:

```
add_neutral_flux model=my_model name=N
```

Typically, you set the value of the sticking coefficient of a neutral flux using the `add_flux_properties` command in the machine configuration. The value also can be specified with the `sticking` parameter in the `add_neutral_flux` command.

It is important to note that, if the sticking coefficient of a neutral flux is set in the `add_neutral_flux` command, it is material independent and cannot be changed later. This is useful, for example, if you want to completely disable reemission for a neutral flux when defining the model, which can be achieved with a command such as:

```
add_neutral_flux model=my_model name=N sticking=1
```

The command `add_ion_flux` allows you to select the physical effects to take into account. For example, the following command defines an energy-independent ion flux called `I` for the model `my_model` with sputtering and reflection enabled, and sputter deposition disabled:

```
add_ion_flux model=my_model name=I energy=independent reflection=true \
   sputter_deposition=false sputtering=true
```

NOTE The definition of an ion flux does not include the specification of its angular distribution, which is provided using the `define_iad` command.

User-Defined Parameter Definition

In this step, user-defined parameters are specified. Here, their attributes (for example, name, valid range, and default value) are set without assigning the actual values of the parameters.

User-defined parameters can be floating-point or integer values, and they are introduced using the `add_float_parameter` and `add_int_parameter` commands, respectively.

In addition, user-defined parameters can be either material dependent or global (material independent). If a parameter is material dependent, it can be assigned a different value for each material. Since deposition uses only one material, material-dependent parameters can be used only in the rate formulas of the `etch` and `etchdepo` models. A default value must be specified for material-dependent parameters. This value is used for materials that are present in the structure but have not been configured in the machine using the `add_material` command.

For example, a material-dependent floating-point parameter named `p`, with default value equal to 5, valid range from 3 to 7, and representing a velocity, can be introduced using the command:

```
add_float_parameter name=p model=my_model min=3 max=7 default=5 \
   quantity=velocity scope=material_dependent
```

As previously mentioned, the actual values of the parameters are not set here, but when you configure a machine using the model to which the parameters belong. The values of the global parameters are set with the `define_etch_machine` or `define_deposit_machine` command; whereas, the values of the material-dependent parameters are set with the `add_material` command.

Rate Formula Definition

The rate formula is defined with the `expression` parameter of the `add_formula` command.

Such an expression might involve any user-defined parameter, and the direct and indirect fluxes of any neutral and ion fluxes, as well as built-in functions to access properties of the exposed surface and mathematical functions.

User-defined parameters can be used in the expression by using their names followed by parentheses ().

The values of the direct and indirect fluxes can be obtained in the rate formula by the RFM built-in functions listed in Data Available for Rate Calculation on page 225.

For example, if a neutral flux `N`, an ion flux `I`, and the parameters `p1`, `p2`, `p3`, and `p4` have been introduced in the model `my_model`, the following statement is a rate formula definition:

```
add_formula model=my_model expression="p1() * total_flux(N) + \
   p2() * cos(theta()) - p3() * direct_flux(I) - p4() * sputtered_flux(I)"
```

NOTE In the rate formula, positive rates correspond to deposition, and negative rates correspond to etching.

Sentaurus Topography 3D also supports material-dependent rate formulas. This means that the mathematical expression used to compute the rate can be different for different materials. A rate formula for a specific material can be specified with the `material` parameter.

For example, the following commands define one rate formula specific for silicon, one specific for photoresist, and one used for all the other materials:

```
add_formula model=my_model material=Silicon expression="-p3() * total_flux(I)"

add_formula model=my_model material=Photoresist \
   expression="-p2() * cos(theta())"

add_formula model=my_model expression="-p1() * total_flux(N)"
```

Machine Configuration Details

The configuration of a machine based on an RFM model consists of three steps:

1. Defining the machine and specifying the user-defined global parameters.
2. Specifying the flux parameter values.
3. Specifying the user-defined material-dependent parameter values.

Defining Machine and Specifying User-Defined Global Parameters

Use the `define_etch_machine` command to configure a machine for models defined with `type=etch` or `type=etchdepo`. Use the `define_deposit_machine` command to configure a machine for models defined with `type=deposit`.

The configuration of a machine with an RFM model requires the following additional information:

- The name of the RFM model must be specified with the `model` parameter.
- If the model is a `deposit` model, the material to deposit must be specified with the `material` parameter.
- If the model is an `etchdepo` model, the material to deposit must be specified with the `deposit_material` parameter.
- If ion fluxes are involved in the model, the name of a collection of ion distributions containing the distributions of all the ion fluxes of the model must be specified with the `iad` parameter.

- If sputtering is enabled for at least one ion flux of the model, the name of a collection of yield functions must be specified with the `yield` parameter. This collection must contain a yield function for all combinations of ion fluxes for which sputtering is enabled and all materials that appear on the surface during at least one of the time steps.
- If reflection is enabled for at least one ion flux of the model, the name of a collection of reflection functions must be specified with the `reflection` parameter. This collection must contain a reflection function for all combinations of ion fluxes for which reflection is enabled and all materials that appear on the surface during at least one of the time steps.
- The values of all the global parameters that are not optional, that is, those defined with `optional=false`.

NOTE The specified collection of reflection or yield functions does not have to contain functions for materials that are present in the structure but do not appear at the surface in any of the time steps.

For example, if `my_model` is an etch RFM model with ion fluxes having sputtering enabled and it involves a mandatory global parameter `p`, a machine using such a model can be defined with the command:

```
define_etch_machine model=my_model iad=my_iad yield=my_yield p=0.5
```

If the parameter `p` is defined with `optional=true`, this parameter can be omitted from the `define_etch_machine` command.

Specifying Flux Parameter Values

The values of the parameters required to compute the indirect fluxes involved in the model are set using the `add_flux_properties` command.

For neutral fluxes, the sticking coefficient can be set with this command. For ion fluxes with sputter deposition enabled, the sticking coefficient, the sputter type, and the sputter exponent can be set with this command.

For `etch` and `etchdepo` models, the flux parameters must be material dependent; whereas, for `deposit` models, they must be material independent.

NOTE The set of flux parameters is not defined by users, but it is determined by the fluxes involved in the model and by the physical effects enabled for them. For example, the following command sets the sticking coefficient for the neutral flux `N` of a `deposit` model to 0.4:

```
add_flux_properties flux=N sticking=0.4
```

Specifying User-Defined Material-Dependent Parameter Values

The values of global parameters are set by the `define_etch_machine` or `define_deposit_machine` command; whereas, the values of material-dependent parameters are set by the `add_material` command. Each `add_material` command sets the values of the material-dependent parameters for a certain material.

Only the values of the material-dependent parameters defined with `optional=false` must be set in the `add_material` command. If a parameter is defined with `optional=true`, the specification of its value is optional in any `add_material` command, and its default value is used for all materials for which it has been omitted.

Therefore, the default value of a material-dependent parameter is used in two different cases:

- When no `add_material` command is issued for a certain material.
- When a material-dependent parameter is defined as optional (`optional=true`) and it is omitted in an `add_material` command.

NOTE It is not necessary to specify material properties with the command `add_material` for all materials, even if none of the material-dependent parameters is optional. All materials for which no properties have been specified explicitly with the command `add_material` are treated equally by using the default values of the material-dependent parameters.

For example, assume that two material-dependent parameters, `p1` and `p2`, are defined as follows:

```
add_float_parameter name=p1 scope=material_dependent default=0.1 \
   optional=true ...

add_float_parameter name=p2 scope=material_dependent default=0.5 \
   optional=false ...
```

In addition, suppose that `p1` and `p2` are the only material-dependent parameters involved in the model for which the machine is being configured. Accordingly, the following statements are valid:

```
add_material material=Silicon p1=0.3 p2=0.9

add_material material=Oxide p2=0.7
```

The first statement sets the values of all the material-dependent parameters. In the second statement, the specification of the value of the optional parameter is omitted and its default

value 0.1 is used. For any other material, the default values 0.1 and 0.5 for the parameters `p1` and `p2`, respectively, will be used.

RFM Commands

The commands for the creation and use of RFM models are provided in Chapter 4 on page 65.

The commands for creating RFM models are listed approximately in the order in which they are used:

- `define_model`
- `add_neutral_flux`
- `add_ion_flux`
- `add_float_parameter`
- `add_int_parameter`
- `add_formula`
- `finalize_model`

The commands for using RFM models are:

- `define_deposit_machine`
- `define_etch_machine`
- `add_flux_properties`

Rate Calculation

The rate formulas defined with the `add_formula` command are used to calculate the deposition or etching rate for each surface element (see add_formula on page 73):

- For deposition, the rate must be greater than or equal to zero.
- For etching, the rate must be less than or equal to zero.
- For simultaneous etching and deposition, there is no restriction on the values of the rate; deposition is represented by positive values and etching by negative values.

Syntax

The syntax for expressions and formulas consists of the following components:

- Numeric constants
- Arithmetic operators: +, -, *, /
- Mathematical constants: e(), pi()
- Mathematical functions: abs(x), sin(x), cos(x), tan(x), asin(x), acos(x), atan(x), atan2(x, y), sinh(x), cosh(x), tanh(x), exp(x), log(x), log10(x), pow(x, y), sqrt(x), ceil(x), floor(x)
- Conditional and relational functions: checked_div(x, y), checked_div(x, y, z), high_pass(x, y, z), higher_pass(x, y, z), low_pass(x, y, z), lower_pass(x, y, z), max(x, y), min(x, y)
- Parentheses
- Data access functions described in Data Available for Rate Calculation on page 225
- Expressions previously defined with the `subexpression` parameter of the `add_formula` command

The arguments x, y, and z to the above functions are themselves expressions.

Conditional and Relational Functions

The following functions are available to implement conditions and control flow:

- `checked_div(<numerator>,<denominator>)`
  Returns `numerator` / `denominator` if `denominator != 0`.
  Otherwise, returns `numerator`, that is, it is equivalent to `checked_div(<numerator>,<denominator>,<numerator>)`.
- `checked_div(<numerator>,<denominator>,<replacement>)`
  Returns `numerator` / `denominator` if `denominator != 0`.
  Otherwise, returns `replacement`.
- `high_pass(<expression>,<threshold>,<attenuation>)`
  Returns `expression` if `expression >= threshold`.
  Otherwise, returns `attenuation`.
- `higher_pass(<expression>,<threshold>,<attenuation>)`
  Returns `expression` if `expression > threshold`. Otherwise, returns `attenuation`.
- `low_pass(<expression>,<threshold>,<attenuation>)`
  Returns `expression` if `expression <= threshold`.
  Otherwise, returns `attenuation`.
- `lower_pass(<expression>,<threshold>,<attenuation>)`
  Returns `expression` if `expression < threshold`. Otherwise, returns `attenuation`.

- `min(<left>,<right>)`
  Returns `left` if `left <= right`. Otherwise, returns `right`.
- `max(<left>,<right>)`
  Returns `left` if `left >= right`. Otherwise, returns `right`.

Data Available for Rate Calculation

The following functions are available to access data related to a surface element, which is available for use in the formula for a deposition or an etching rate:

- `defined_yield(<flux>)`: Returns the value of the yield function for the specified ion flux and the material of the current surface element, and the angle between the vertical and the surface normal of the current element. This function is available for all ion fluxes and the `default_species`.

  NOTE Models using this function in any of their rate formulas are not supported by a machine having `rotation=continuous`. In models to be used with machines having `rotation=continuous`, the sputtered flux can be computed using the function `sputtered_flux(<flux>)`.
- `defined_yield(<flux>, <material>)`: Returns the value of the yield function for the specified ion flux and the specified material, and the angle between the vertical and the surface normal of the current element. This function is available for all ion fluxes and the `default_species`.

  NOTE Models using this function in any of their rate formulas are not supported by a machine having `rotation=continuous`. In models to be used with machines having `rotation=continuous`, the sputtered flux can be computed using the function `sputtered_flux(<flux>)`.
- `direct_flux(<flux>)`: Returns the value of the direct flux arriving at the surface element for the specified flux (all neutral and ion fluxes).
- `directional_value(<value_100>, <value_110>, <value_111>)`: The first argument specifies the value for the <100> direction; the second argument, the <110> direction; and the third argument, the <111> direction. Depending on the direction of the surface normal of an element, an interpolated value is calculated and returned. This function can be used to implement orientation-dependent etching or deposition models.
- `pad_pressure()`: Returns the pressure produced by the pad at the surface element in Pa.

  NOTE Currently:

  - Only models of `type` equal to `etch` or `etchdepo` support this function in their rate formula expressions.

- Models using this function cannot use these RFM functions: `direct_flux(<flux>)`, `sputter_depo_flux(<flux>)`, `sputtered_flux(<flux>)`, `total_flux(<flux>)`.
- Periodic boundary conditions are used to evaluate the RFM function `pad_pressure()` independently of what has been specified with the command `define_boundary_conditions`.

- `<parameter>()`: Returns the value of the specified global or material-dependent user-defined parameter.
- `sputter_depo_flux(<flux>)`: Returns the value of the flux of the redeposited sputtered material associated with the specified ion flux arriving at the surface element (ion flux with sputter deposition enabled).
- `sputtered_flux(<flux>)`: Returns the value of the sputter flux emitted from the surface element for the specified ion flux (ion flux with sputtering enabled).
- `sticking(<flux>)`: Returns the value of the sticking parameter that was specified with the `add_flux_properties` command for the given flux (neutral flux or ion flux with sputter deposition enabled).
- `<subexpression>()`: Returns the value of the specified subexpression. If the current expression is global, only other global subexpressions can be accessed. If the current expression is material dependent, other subexpressions for the same material or global subexpressions can be accessed.
- `theta()`: Returns the value of the angle between the surface normal and the vertical. The value is in the range $[0, \pi]$.

  NOTE Models using this function in any of their rate formulas are not supported by a machine having `rotation=continuous`.

- `total_flux(<flux>)`: Returns the value of the total flux arriving at the surface element for the specified flux (neutral flux or ion flux with reflection enabled).

  The total integrated ion flux is defined as $\Gamma^{ion}_{total} = \Gamma^{ion}_{direct} - \Gamma^{ion}_{direct\ reflected} + \Gamma^{ion}_{reflection}$ where:

  - $\Gamma^{ion}_{direct\ reflected}$ is the portion of the direct flux that has been reflected away from a surface element.
  - $\Gamma^{ion}_{reflection}$ is the sum of all fluxes incoming at a surface element that were reflected from other surface elements.
- `visible()`: Returns if the surface element is visible from vertically above. Returns 1 if the surface element is visible or 0 otherwise.

  Models using the `visible()` function in any of their rate formulas are not supported by a machine having `rotation=continuous`.

  NOTE A syntax error occurs if functions taking a flux name as argument are used with a flux that is of an incompatible type.

NOTE The identifier `default_species` can be used as a placeholder parameter for the `defined_yield` functions in models that do not specify any ion fluxes to access yield functions. The yield functions must be defined with `default_species` as the species.

NOTE The integrated direct reflected flux and the reflection flux are not accessible as RFM functions. However, they are available as datasets created when using the `plot_interval` parameter of the `deposit` and `etch` commands.

Examples

The `direct_flux()` function is available to a model with only a neutral flux `n`. However, the `sputtered_flux()` function is not available:

```
define_model name=N type=deposit description=""

add_neutral_flux model=N name=n

# Next command causes a syntax error because n is not a valid argument
# for sputtered_flux().
add_formula model=N expression={direct_flux(n) + sputtered_flux(n)}

finalize_model model=N
```

Similarly, if ion fluxes are defined for a model but the necessary effects are disabled, a syntax error occurs:

```
define_model name=I type=deposit description=""

add_ion_flux model=I name=i energy=independent reflection=false \
   sputtering=true sputter_deposition=false

# Next command causes a syntax error because i is not a valid argument
# for sputter_depo_flux().
add_formula model=I expression={direct_flux(i) + sputter_depo_flux(i)}

finalize_model model=I
```

Example: Reimplementing and Using ionmill Etching Model

The following example demonstrates how the `ionmill` etching model can be reimplemented and used as an RFM model:

```
define_model name=mck_ionmill type=etch \
   description="user-defined ionmill etching"

add_float_parameter model=mck_ionmill name=rate min=0 default=0 \
   scope=material_dependent quantity=velocity

add_float_parameter model=mck_ionmill name=s1 default=1 \
   scope=material_dependent quantity=dimensionless

add_float_parameter model=mck_ionmill name=s2 default=0 \
   scope=material_dependent quantity=dimensionless

add_formula model=mck_ionmill \
   expression={ -rate() * visible() * (s1() * cos(theta()) \
   + s2() * pow(cos(theta()), 2) + (1 - s1() - s2()) * pow(cos(theta()), 4)) }

finalize_model model=mck_ionmill

define_etch_machine model=mck_ionmill

add_material material=Silicon rate=0.9 s1=5.5 s2=-6

add_material material=Oxide rate=0.7 s1=5 s2=-7

etch ...
```

References

[1] T. K. Chini *et al*., “The angular dependence of sputtering yields of Ge and Ag,” *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, vol. 72, no. 3–4, pp. 355–358, 1992.

CHAPTER 7 Working With Reaction Models

This chapter explains how to use reaction models based on the particle Monte Carlo (PMC) method in Sentaurus Topography 3D. It also discusses how to integrate PMC-based simulations with other Sentaurus Topography 3D functionality.

Setting Up a Simulation Using a Reaction Model

This section provides a simple example to start using reaction models. A model for a simple process will be set up and a simulation using that model will be run.

In the process you want to model, fluorine gas is fed to the reactor, where a silicon wafer is being processed. When fluorine reaches silicon, they react with a given probability, and silicon is etched.

The first step is to define the model. A reaction model consists of:

- The source species, that is, the species coming from the reactor in a gaseous state.
- The reactions relevant for the modeled process.

A reaction model for the above-described process can be set up as follows:

```
# Begin the definition of a model named 'm1'.
define_model name=m1 description="Simple reaction model"

# Add a species named "F" coming from the reactor to model 'm1'.
add_source_species model=m1 name=F

# Add a reaction named 'etch_reaction' to model 'm1'.
# According to this reaction, when species 'F' and 'Silicon' react, a
# silicon atom is removed and a 'SiF' molecule goes into the reactor.
# The net effect is to remove a silicon atom from the structure.
add_reaction model=m1 name=etch_reaction \
   expression="F<g> + Silicon<s> = SiF<g>"

# End the definition of model 'm1'.
finalize_model model=m1
```

As for RFM models, the definition of a new reaction model starts with the `define_model` command (see define_model on page 124) and ends with the `finalize_model` command (see finalize_model on page 192).

Since fluorine gas molecules are electrically neutral, their angular distribution is assumed to be isotropic. An angular distribution described by Eq. 5, p. 32 with m equal to 1 describes an isotropic distribution. Therefore, an isotropic angular distribution can be set up for species `F` with the following command (see define_species_distribution on page 133):

```
define_species_distribution name=my_distribution species=F exponent=1 \
   flux=1e-3
```

After the model and the distribution of its source species have been specified, a machine can be defined using the `define_etch_machine` command (see Machines on page 10 and define_etch_machine on page 109):

```
define_etch_machine model=m1 name=m1_machine \
   species_distribution=my_distribution
```

In Sentaurus Topography 3D, machines are used to select the model to use for the simulation and to assign the parameter values required by the selected model. Reaction models require probabilities to be set for each reaction. The reaction probabilities specify the probability that a reaction will occur when its reactants become available.

The following `add_reaction_properties` command (see add_reaction_properties on page 92) sets the reaction probability to 0.7 for the reaction named `etch_reaction` of the model used by the machine named `m1_machine`:

```
add_reaction_properties machine=m1_machine reaction=etch_reaction p=0.7
```

The machine is now fully set up and can be used to process a structure. As explained in Simulating Process Steps on 2D and 3D Structures on page 14, the initial structure can be either loaded from a TDR boundary file or defined using geometric etching and deposition steps. For simplicity, assume that the initial structure is stored in a TDR file called `init.tdr`. Then, the following two commands define the initial structure and start a simulation using the machine defined above, respectively:

```
define_structure file=init.tdr

etch spacing=0.1 time=0.5 method=pmc machine=m1_machine
```

The results of a simulation that used the PMC method can be saved in TDR file format with the command (see save on page 199):

```
save type=gc
```

Integration With Other Sentaurus Topography 3D Functionality

The `etch` command runs a simulation using the selected machine. The name of the resulting structure is specified by the parameter `structure` of the `etch` command.

The data structure used to store internally the result depends on the simulation method used (see parameter `method` of etch on page 154):

- When `method=levelset`, when etching with a mask or etching a geometric shape, the simulation result is stored internally in a boundary data structure (see Boundary Types on page 13).
- When `method=pmc`, the simulation result is stored internally in a volumetric data structure.

The commands in Table 52 require input structures to be stored in a boundary data structure.

Table 52 Commands requiring input structures to be stored internally in a boundary data structure

Command	Comment
`deposit`	
`etch`	When using a level set–based model or when etching a geometric shape or a mask.
`extend_structure`	
`extract`	Except when using any of the following: `type=bounding_box, type=cylinder_hole_profile, type=dimension, type=interface material1=<c> material2=<c>, type=probe property=length, type=probe property=material`
`fill`	
`filter_structure`	When any of the following are specified: `type=decimate, type=merge_regions, type=rediscretize_boundary, type=remove_disconnected_top_parts, type=remove_region, type=rename_region, type=replace_material, type=smooth`
`litho`	
`pattern`	
`save`	When `type=closed_surface`, or `type=exposed_surface`, or `type=final_structure`. **NOTE** When `type=gc`, `type=pmc`, or `type=vbe` is specified, the `save` command requires the input structure to be stored in a *volumetric* data structure.
`truncate`	

Converting a Result Structure to TDR Boundary File Format

Whenever any of the commands in Table 52 on page 231 must be executed on the result of a simulation that used the PMC method, an explicit conversion is needed.

You can use the `filter_structure` command to convert the result of a PMC-based simulation to a structure stored in TDR boundary file format (see filter_structure on page 186), as demonstrated in the following example. In particular, the `filter_structure` command is used twice:

- Before starting the simulation using the PMC method, it creates a copy of the initial structure (`filter_structure type=copy`).
- When the simulation is completed, it runs Boolean operations (`filter_structure type=boolean`) and produces a structure stored in TDR boundary file format.

Example

```
# The 'define_structure' command creates a structure that is internally
# stored in TDR boundary file format. The newly created structure is named
# 'default_structure'.
define_structure material=Silicon point_min={0 0 0} point_max={1 1 1}

# The 'fill' command operates on a structure stored in TDR boundary file
# format. The result of the command 'fill' overwrites the structure named
# 'default_structure' and is still stored internally in TDR boundary file
# format.
fill material=Oxide thickness=0.5

# A shape named 's' is defined to create an opening in the oxide layer
# added with the command 'fill'.
define_shape type=cube name=s point_min={0.2 0.2 1} point_max={0.8 0.8 2}

# The shape 's' is etched away from the structure named 'default_structure'.
# The result of the command 'etch' overwrites the structure 'default_structure'
# and is still stored internally in TDR boundary file format.
etch shape=s

# A copy of structure 'default_structure' is created. The copy is named
# 'initial_structure' and is stored in TDR boundary file format as well.
filter_structure type=copy name=initial_structure

# A reaction model named 'reaction_model' is defined.
define_model name=reaction_model description=""
...
finalize_model model=reaction_model
...
```

```
# A machine using model 'reaction_model' is defined.
define_etch_machine model=reaction_model ...
...
# The structure named 'default_structure' is etched using the reaction model
# named 'reaction_model'. The result structure overwrites the structure named
# 'default_structure'. Since the PMC method was used, the result of this
# command is stored internally with a volumetric data structure.
etch spacing=0.1 time=1 method=pmc

# The structure 'default_structure' stored in a volumetric data
# structure is saved in the appropriate TDR file format (see Table 52).
save type=gc

# A new structure named 'boundary_structure' is created using a Boolean
# operation. The operands of the Boolean operation are specified with the
# parameters 'body' and 'structure'. It is worth noting that the structure
# given with the parameter 'body' is stored internally in TDR boundary file
# format; whereas, the structure given with parameter 'structure' is stored
# internally in a volumetric data structure in TDR file format.
filter_structure type=boolean body=initial_structure \
   structure=default_structure name=boundary_structure

# Since the structure named 'boundary_structure' is stored in TDR boundary
# file format, the 'extract' command can be used to return information
# on it (see Table 52).
extract type=1d_cut structure=boundary_structure ...
...

# Since the structure named 'boundary_structure' is stored in TDR boundary
# file format, it can be saved with type=final_structure (see Table 52).
save structure=boundary_structure type=final_structure
```

Boolean operations are not run automatically after each PMC-based simulation to convert the resulting structure into a boundary data structure because they are computationally expensive and are not always needed. Boolean operations are necessary only when one of the commands in Table 52 on page 231 must be run, but they are not required to save the resulting structure in a TDR file format and to visualize it. In addition, PMC-based simulations can be run on the result of a previous PMC-based simulation without executing any Boolean operations.

CHAPTER 8 Physical Model Interface for Etching and Deposition

This chapter describes the physical model interface for etching and deposition.

Overview

The physical model interface (PMI) in Sentaurus Topography 3D gives you the possibility to extend the modeling capabilities of Sentaurus Topography 3D by implementing a new model in the form of a C++ class.

Sentaurus Topography 3D uses the level-set method to move the exposed surface during a time step. However, direct use of the level-set data structures requires detailed knowledge of the particular implementation and, therefore, direct use is complicated and error prone. Similar to the PMI in Sentaurus Topography (see the *Sentaurus™ Topography User Guide*), an explicit surface representation is used for the PMI in Sentaurus Topography 3D.

The PMI in Sentaurus Topography 3D supports three different modes: deposition, etching, and simultaneous etching and deposition.

The following sections describe the command file interface for using a PMI-based model, the C++ interface for creating a new model, and the input and output parameters.

Command File Interface

The `add_material`, `define_deposit_machine`, and `define_etch_machine` commands can handle PMI-based models.

The name with which a PMI-based model is specified, with the `model` parameter of the `define_deposit_machine` and `define_etch_machine` commands, is defined in the source code of the model and can have an arbitrary value. The only restriction is that it must not be identical to the name of a built-in model.

NOTE Model names beginning with the prefix `pmi` are guaranteed not to conflict with built-in models.

In the command file, the parameters of a PMI-based model are specified in the same way as for built-in models. Therefore, for deposition models, all model parameters are specified using the `define_deposit_machine` command. For etching models, the material-independent parameters of the model are specified with the `define_etch_machine` and the material-dependent parameters with the `add_material` command. For simultaneous etching and deposition, the deposition-specific parameters are specified with the `define_etch_machine` command.

A PMI-based model can have an arbitrary number of model-specific command file parameters. These parameters can have arbitrary names. The only restriction on the parameter names is that they must not be identical to the names of built-in parameters of a PMI-based model.

Table 53 lists the built-in parameters of the PMI-related commands. The command file parameters have four possible types: Boolean, floating point, integer, and string.

Table 53 Built-in parameters of commands for using PMI-based models

Command	Built-in parameters
`add_material`	`machine, material`
`define_deposit_machine`	`material, model, name, rotation, tilt`
`define_etch_machine`	`deposit_material, model, name, rotation, tilt`

A default value must be specified for each user-defined parameter, and you can specify whether a user-defined parameter is optional or mandatory. When specifying the parameters for a material with the `add_material` command, a parameter can be omitted if it has been declared as optional. In this case, the default value will be used.

Depending on the type of a parameter, the default value can be queried directly with the `bool_parameter_default()`, `float_parameter_default()`, `int_parameter_default()`, or `string_parameter_default()` function.

C++ Interface

At the C++ level, the interface consists of several abstract base classes:

- `Model_info` defines the model name, the model-specific command file parameters, and the plot datasets generated by the model.
- `Model_rate` calculates the surface rate and the plot datasets for each time step.
- `Global_data` provides access to the values of the command file parameters and other data that does not change during the time steps.
- `Time_step_data` provides access to surface-related data and data that changes between time steps.

The developer of a PMI-based model must implement a class that is derived from `Model_rate` and must implement three free functions.

The function `describe_model()`, which must be implemented by users, is called during the initialization of a PMI-based model. It defines the model name, the model parameters, and the plot datasets generated by the model. Figure 23 shows a unified modeling language (UML) diagram of the classes used to initialize a PMI-based model.

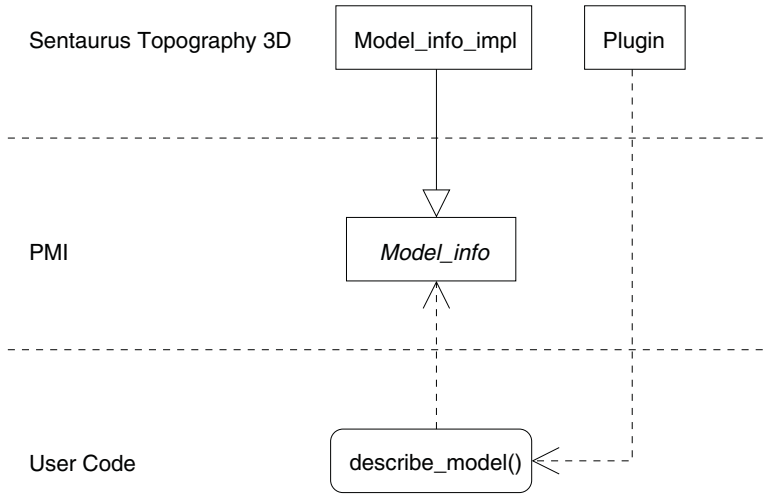


Figure 23 UML diagram for initializing a PMI model

The class derived from `Model_rate` is used in each time step to calculate the surface rate and to initialize the plot datasets. Figure 24 shows a UML diagram of the PMI-related classes used for the surface rate calculation in each time step.

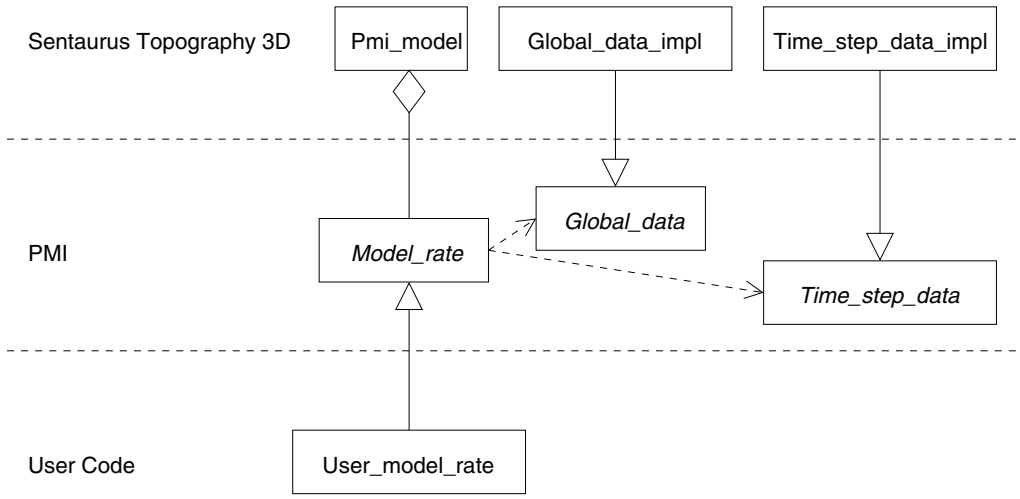


Figure 24 PMI class hierarchy for calculating surface rates

Detailed documentation of the abstract classes is available in the corresponding header files, which are located in the directory `${STROOT}/tcad/${STRELEASE}/lib/sptopo3d/include`.

Implementing a New Model

As previously mentioned, to implement a PMI-based model, it is necessary to implement the function `describe_model()`. The only argument of this function is a reference to an object derived from `Model_info`. This object is used to define the model name and whether the model is used for deposition, or etching, or simultaneous etching and deposition. Besides that, it is used to define all model-specific command file parameters (name, type, and default value if applicable), and the names and units of the datasets that are generated by the model.

A class derived from `Model_rate` is used to implement the surface rate calculation for each time step. An instance of the class derived from `Model_rate` is created when the `deposit` or `etch` command is executed. For each time step, the function `Model_rate::calculate_rate()` is called to calculate the surface rates and the plot datasets. The free functions `model_rate_factory()` and `model_rate_delete()` are used to create and delete instances of the class derived from `Model_rate`, respectively.

Information Available to a Model

All input data available to a PMI-based model is provided by the classes `Global_data` and `Time_step_data`.

The class `Global_data` makes the values of model-specific command file parameters available. In addition, it contains the names of all the materials of the input structure and other data that does not change between time steps. For example, for etching models, a list of all materials for which parameters have been defined with the `add_material` command is available.

The class `Time_step_data` provides information about the current surface and its properties. It also contains other data that changes between time steps, for example, the size of the previous time step.

For each surface vertex, a list of indices in the list of structure materials at the vertex is available. If the vertex is located at a material interface, the list of structure material indices will contain more than one entry, and you are responsible for choosing which material will be used for the rate calculation.

Information about the surface material is available in all three modes. This information allows models to calculate material-specific rates also for the deposition mode.

The arguments of the free function `model_rate_factory()` are a reference to an object derived from `Global_data` and a reference to an object derived from `Time_step_data`. Therefore, all information is available to the constructor of the class derived from `Model_rate`, which is called in `model_rate_factory()`. The reference to the object derived from `Global_data` is guaranteed to be valid for the entire lifetime of the PMI model instance.

In `Model_rate::calculate_rate()`, a reference to an object derived from `Time_step_data` is available. In addition, you obtain a reference to an object derived from `Plot_data`, which provides access to the plot data arrays.

Additional Input Data

Additional input data that is too complicated to be read from the command file might be required by a model, for example, complex chemical reactions. This kind of data can be read and stored as member data by the constructor of the derived class for later use in `Model_rate::calculate_rate()`.

In some models, it might be necessary to store data from previous time steps. This additional data also can be stored as member data of the derived class object.

Data Types Used in Interface

To avoid linking problems, only C-style data types or types for which the corresponding header files are provided are used in the interface. While this makes handling the data more cumbersome, it helps to prevent problems that are difficult to debug.

Error Handling

The only safe way to signal to the calling code that an error has occurred in a constructor is to throw an exception. However, C++ does not allow exceptions to cross interfaces as they are used for the PMI.

An exception that is thrown in a PMI model and that is not caught and handled in the PMI model will abort the simulation immediately. For a PMI model, this means that exceptions that were thrown in the constructor must be caught in the factory functions. If there is an error, the null pointer is returned and the simulation is stopped in a controlled way.

Similarly, exceptions must not leave `Model_rate::calculate_rate()`. Here, an error is signaled to the calling code by returning the value `false`. The simulation is then stopped in a controlled way.

The function `Global_data::message()` can be used to output messages to the standard output and the log file. For error messages, the verbosity level should be set to low; while for other messages, a verbosity level of medium or higher should be used.

Compiling the Source Code

The source code can be compiled and linked using the `cmi` tool, which calls the C++ compiler and linker with the correct command-line options for each supported platform.

The `cmi` tool is contained in each TCAD release and is mainly used for Sentaurus Device CMI-based and PMI-based models:

```
cmi -I${STROOT}/tcad/${STRELEASE}/lib/sptopo3d/include User_model_rate.cc
```

See Compact Models User Guide, Compilation of C++ (.C) Files on page 141.

Using Additional Source Files or Libraries

A PMI-based model is not limited to the functionality declared in the abstract base classes of the PMI or directly derived from it. You can use additional source files or libraries, which provide additional functionality, to build the shared library for a model. For example, a model might need to solve a linear equation system. In this case, it is possible to link with a library that provides this functionality.

Additional source files or libraries can be specified as command-line parameters for the `cmi` tool, which will combine them into one shared library.

Using the C++ Standard Library

When using the features of the C++ Standard Library in the implementation of a PMI-based model, depending on the platform, it might be necessary to link an additional library that provides this functionality. For example, on 64-bit Linux, it is necessary to link `libstdc++` by adding `-lstdc++` to the command-line parameters of the `cmi` tool.

To avoid problems at link-time or runtime, it is necessary to use *exactly* the same version of the C++ compiler for the compilation of the PMI code as used for the compilation of Sentaurus Topography 3D.

Refer to the *Compilers for CMI and PMI* in the TCAD Sentaurus release notes for information about the compiler versions used in Sentaurus Topography 3D Version N-2017.09.

Loading the Shared Library

During the initialization of the simulator, the current directory and the path defined with the environment variable `SPTOPO3D_PMI_PATH` are searched for shared libraries that conform to the CMI-naming or PMI-naming convention (platform-specific suffix, for example `.so.linux64`).

These files are loaded and searched for the three free functions: `describe_model()`, `model_rate_factory()`, and `model_rate_delete()`. If all three functions are found, the model is added to the list of available models and can then be used for an arbitrary number of deposition or etching steps. Otherwise, the shared library is closed again.

NOTE The shared library containing a PMI model is unloaded only when the simulator is stopped. This must be taken into account when using static variables.

Debugging

Debugging the PMI code is more difficult than debugging a standard program for several reasons:

- Only the PMI code created by users is available as source code, and debugging symbols are not available for Sentaurus Topography 3D.
- The library containing the PMI code is loaded dynamically whenever the conditions mentioned in the previous section are fulfilled.
- Typically, Sentaurus Topography 3D is started by a wrapper script that sets some environment variables. When using a debugger, it is necessary to set these environment variables manually and then to start the debugger with the executable of Sentaurus Topography 3D without the wrapper script.

The details of debugging are highly platform dependent and tool dependent, especially when using shared libraries.

Input and Output Parameters

This section describes the input parameters available to a PMI-based model.

The following input data is available from objects derived from `Global_data`, `Time_step_data`, and `Plot_data` for model initialization and rate calculation:

- Model-specific command file parameters
- Default values of model-specific command file parameters
- Model-specific plot datasets
- Surface vertices
- Surface normal at vertices
- Curvature at vertices
- Area associated with surface vertices
- Material at vertices
- Mutual visibility between surface vertices
- Point classification (inside, on surface, outside)
- Size of the previous time step
- Machine materials
- Structure materials
- Conversion between material indices and material names
- Conversion between material names and material indices
- Direct flux at vertices for analytic flux distributions
- Visibility of vertices from source vertically above
- Neighbor elements information

The output parameter for the rate calculation is an array containing the rate at each surface vertex.

Each surface vertex belongs to a discretization element of the surface. For each surface vertex, information about the neighbor elements is available.

The `Plot_data` parameter provides access to the plot datasets and to a query function that determines whether plotting is enabled.

CHAPTER 9 Known Issues and Limitations

This chapter provides information about known issues and limitations in the use of Sentaurus Topography 3D.

Description of Known Issues and Limitations

The following known issues and limitations apply to this version of Sentaurus Topography 3D:

- There is no support for energy-dependent flux integration in level set–based models.
- Reflective sputtering is not supported when processing 2D structures.
- Reflection is not supported for the rate formula module (RFM) when processing 2D structures.
- Boundary conditions specified by the command `define_boundary_conditions` have no effect on the indirect flux computations for 2D structures.
- Flux integration in 2D is performed with the radiosity method exclusively. There is no Monte Carlo engine in 2D.
- The radiosity method does not provide accurate results for machines with a tilt angle greater than approximately 70° for 3D structures. The Monte Carlo method is recommended for such machines.
- Built-in models do not support machines with `rotation=continuous`.
- The RFM functions `defined_yield(<flux>)`, `defined_yield(<flux, material>)`, `theta()`, and `visible()` are not supported by etching or deposition machines with `rotation=continuous`.
- When `rotation=continuous` is used, the actual equivalent error for any species always approaches 0.5 as the tilt angle approaches 90° .
- As described in Discretization Size and Accuracy When Using the Level-Set Method on page 9, the use of the level-set method to discretize and move the surface of a structure introduces artifacts. When using models for simultaneous etching and deposition, these artifacts can cause the creation of thin layers of deposited material. Depending on the values of the model parameters, this might prevent etching and cause even more deposition.
- Only `reflective` boundary conditions are supported when using the built-in `electrodeposition`, `spin_on`, or `wet` deposition model.
- The `define_deposit_machine` and `define_etch_machine` commands do not support the parameters `tilt` and `rotation` when `model=crystal`.

- The parameters `cfl`, `file`, `merge`, `plot_interval`, `plot_type`, `stop_plane`, `stop_point`, and `update_scheme` are not supported by the `deposit` command when using a machine having `model=crystal`.
- The `etch` command does not support the `abs_min_deposition_thickness`, `cfl`, `file`, `min_deposition_thickness`, `plot_interval`, `plot_type`, `region_query_accuracy`, `stop_plane`, `stop_point`, and `update_scheme` parameters when using a machine having `model=crystal`.
- The `define_boundary_conditions` command has no effect on the built-in deposition and etching models named `crystal`. Boundary conditions are always `reflective` in simulations using deposition and etching machines having `model=crystal`.
- The `define_deposit_machine` command does not support the parameters `tilt` and `rotation` when `model=electrodeposition` or `model=spin_on`.
- The parameters `tilt` and `rotation` are not supported by the `define_etch_machine` command when `model=wet`.
- The parameters `tilt` and `rotation` of the `define_deposit_machine` and `define_etch_machine` commands do not support the radian measurement unit.
- The parameters `stop_plane`, `stop_point`, and `update_scheme` are not supported by the `deposit` command for machines having `model=spin_on`.
- For machines having `model=spin_on`, the `deposit` command does not support structures whose exposed surface cannot be described as a single-valued function of the x- and y-coordinates in three dimensions, or of the x-coordinate in two dimensions. However, topologically connected exposed surfaces are allowed to contain elements parallel to the vertical direction.
- The `etch` command does not support the `mask` parameter for 2D structures.
- The `pattern` command is not available for 2D structures.
- Sentaurus Lithography integration only works in three dimensions.
- The `slice_angle` parameter of the `define_structure` command does not support the radian measurement unit.
- The `filter_structure` command with `type=smooth` is not available for 2D structures.
- Boundary conditions set with the `define_boundary_conditions` command are ignored when using RFM models that use the RFM function `pad_pressure()` and periodic boundary conditions are always applied.
- Models using the RFM function `pad_pressure()` cannot use these RFM functions: `direct_flux(<flux>)`, `sputter_depo_flux(<flux>)`, `sputtered_flux(<flux>)`, `total_flux(<flux>)`.

- The parameter `structure` of the following commands cannot specify a structure directly obtained as the result of a PMC-based simulation:
  - `deposit`
  - `etch` (when performing geometric etching using a shape or using a mask, or when using a level set–based model)
  - `extend_structure` and `fill`
  - `filter_structure` (when `type=decimate`, `type=merge_regions`, `type=rediscretize_boundary`, `type=remove_disconnected_top_parts`, `type=remove_region`, `type=rename_region`, `type=replace_material`, or `type=smooth`)
  - `litho` and `pattern`
  - `save` (when `type=closed_surface`, or `type=exposed_surface`, or `type=final_structure`)
  - `truncate`

  Integration With Other Sentaurus Topography 3D Functionality on page 231 explains how to convert the structure in such a way that these commands can be used.
- When `type=gc`, `type=pmc`, or `type=vbe`, the parameter `structure` of the `save` command must specify a structure obtained as the result of a PMC-based simulation.
- The `flat_orientation` and `vertical_orientation` parameters of the `etch` command are not supported when `method=pmc`.
- Boundary conditions of type `none` are not supported when processing a structure with the PMC method.
- Nonuniform spacing of the simulation grid is not supported when using the `etch` command with `method=pmc`.
- When using a built-in model, the `add_interface_layer` command is not supported for the `sticking` and `reflection` parameters.
- When using the `etch method=pmc` command to process a structure obtained as the result of a previous PMC-based simulation, the spacing specified with the `spacing` parameter is ignored.
- When running a simulation with a machine using a reaction model that allows deposition, the vertical extent of the computational domain must be large enough to contain the deposited material.

- The parameter `structure` of the `extract` command can specify a PMC structure only when `type=bounding_box`, `type=dimension`, or `type=interface` is used to extract the interface between two materials (that is, when also parameters `material1` and `material2` are specified), or when `type=probe` and `property=material`, or when `type=probe` and `property=length`.
- The parameter `stop_material` of the `etch` command can be used only when `method=pmc`.

APPENDIX A Examples

This appendix provides examples that demonstrate how to use Sentaurus Topography 3D with different models. Three-dimensional input structures can be created using either Sentaurus Topography 3D itself or Sentaurus Structure Editor. Examples on how to create 3D structures are available in the Sentaurus™ Structure Editor User Guide.

Initial Structure Generation

The following examples show how to use geometric etch and deposition steps to generate simple initial structures in Sentaurus Topography 3D.

Simple Trench

This example shows how to create a simple silicon trench:

```
# Define a silicon unit cuboid as the initial bulk structure.
define_structure material=Silicon point_min={0 0 0} point_max={0.24 0.05 0.33}

# Define a cuboid shape that is used to etch a trench out
# of the initial structure.
define_shape type=cube name=trench point_min={0.06 0.0 0.15} \
   point_max={0.18 0.05 0.33}

# Use the shape to etch a trench.
etch shape=trench

# Save the final structure to file.
save file="Structures/trench.tdr"
```

Fill

This example illustrates the use of the `fill` command when generating an initial structure:

```
# Define a silicon unit cube as the initial bulk structure.
define_structure material=Silicon point_min={0 0 0} point_max={0.35 0.05 0.07}

# Fill the initial structure with aluminum with a thickness
# of 0.08 micrometers.
fill material=Aluminum thickness=0.08

# Fill again with Photoresist with a thickness of 0.18 micrometers.
fill material=Photoresist thickness=0.18

# Define a first cuboid shape that is used to etch a first trench
# out of the previously generated structure.
define_shape type=cube name=cuboid1 point_min={0.07 0.0 0.07} \
   point_max={0.14 0.05 0.33}

# Define a second cuboid shape that is used to etch a second trench
# out of the previously generated structure.
define_shape type=cube name=cuboid2 point_min={0.21 0.0 0.07} \
   point_max={0.28 0.05 0.33}

# Etch the two cuboids.
etch shape=cuboid1
etch shape=cuboid2

# Save the final structure to file.
save file=Structures/etch_input.tdr
```

Mask or Patterning

This example illustrates the use of the `define_mask` and `pattern` commands:

```
# Define an initial substrate.
define_structure material=Silicon point_min={0 0 0} point_max={1 1 1}

# Define a mask
# - the mask is loaded from the file 'mask.gds'.
# - the Sentaurus Topography 3D internal name is 'mask_1'.
# - the layer used from the file 'mask.gds' is called '1:0'.
# - the domain cut from the layer '1:0' is the rectangle {0 0} {1 1}.
define_mask name=mask_1 file=mask.gds layer=1:0 \
   domain_min={0 0} domain_max={1 1}

# Pattern using 'mask_1' and material Photoresist.
pattern mask=mask_1 material=Photoresist thickness=0.1
```

```
# Save the resulting structure to TDR.
save
```

Deposition

The following examples show the use of the deposition models in Sentaurus Topography 3D.

Crystallographic Orientation–Dependent Deposition

This example illustrates the crystallographic orientation–dependent deposition model:

```
# Define a crystal deposition machine, set the deposition rates along the
# <100>, <110>, and <111> directions of the lattice, set the deposited
# material, and set the material where deposition will occur.
define_deposit_machine model=crystal material=Silicon \
   selective_materials=Silicon rate_100=0.3 rate_110=0.1 rate_111=0.01

# Create the initial structure and save it.
define_structure material=Silicon point_min={0 0 0} \
   point_max={0.03 0.06 0.18} slice_angle=-90
define_shape type=cube name=c0 point_min={0.01 0.02 0.14} \
   point_max={0.02 0.04 0.18}
define_shape type=cube name=c1 point_min={0 0 0.05} point_max={0.03 0.02 0.18}
define_shape type=cube name=c2 point_min={0 0.04 0.05} \
   point_max={0.03 0.06 0.18}
define_shape type=cube name=c3 point_min={0 0 0.05} point_max={0.03 0.02 0.15}
define_shape type=cube name=c4 point_min={0 0.04 0.05} \
   point_max={0.03 0.06 0.15}
define_shape type=cube name=c5 point_min={0 0. 0.15} point_max={0.01 0.06 0.2}
define_shape type=cube name=c6 point_min={0.02 0. 0.15} \
   point_max={0.03 0.06 0.2}
etch shape=c0
etch shape=c1
etch shape=c2
deposit shape=c3 material=Oxide merge=true
deposit shape=c4 material=Oxide merge=true
deposit shape=c5 material=Nitride
deposit shape=c6 material=Nitride
save

# Start the silicon deposition process and set the crystallographic orientation
# of the deposited region.
deposit spacing={0.002 0.002 0.002} time=0.25 flat_orientation={1 1 0} \
   vertical_orientation={0 0 1}
```

```
# Save the final structure to a file.
save
```

Electrodeposition

This example illustrates the electrodeposition model:

```
# Define an electrodeposition machine and set its physical parameters.
define_deposit_machine model=electrodeposition material=Copper \
   overpotential=-0.21 bulk_distance=0.05 temperature=300 \
   depo_bulk_concentration=2.4e-4 exchange_current_density=1 \
   exchange_current_density_inh=0.039 \
   exchange_current_density_acc=2.5 inh_bulk_concentration=5e-8 \
   inh_diffusivity=5e-7 inh_adsorption_rate=6e4 \
   inh_displacement_rate=0 acc_bulk_concentration=0.64e-7 \
   acc_adsorption_rate=6e2 depo_diffusivity=4e-6 acc_diffusivity=1e-5

# Create an initial structure with a hole by removing a cuboid
# from a block of silicon.
define_structure material=Silicon point_min={-0.2 0} point_max={0.2 2}

define_shape name=hole type=rectangle point_min={-0.1 1.5} point_max={0.1 2}

etch shape=hole

# Start the copper deposition process and create a plot every 0.1 minutes.
deposit time=1.5 spacing=0.005 plot_interval=0.1

# Save the final structure to a file.
save
```

Electroplating Deposition

This example illustrates the electroplating deposition model:

```
# Define an electroplating deposition machine and set the deposited material,
# the rate, and the initial vertical change of the accelerator concentration.
define_deposit_machine model=electroplating material=Copper rate=0.1 delta=5

# Create an initial structure with a hole by removing a cuboid
# from a block of silicon.
define_structure material=Silicon point_min={0 0 0} point_max={1 1 1} \
   region=bulk

define_shape type=cube name=hole point_min={0.3 0.3 0.5} \
   point_max={0.7 0.7 1.0}

etch shape=hole
```

```
# Start the copper deposition process and create a plot every 0.1 min.
deposit time=1 spacing={0.03 0.03 0.03} plot_interval=0.1

# Save the final structure to a file.
save
```

High-Density Plasma Deposition

This example illustrates the high-density plasma deposition model:

```
# Define a deposition machine that uses the hdp model
# and set the anisotropy, the angular distribution exponent,
# the deposited material, the rate, the redeposition
# coefficient, the sputtering coefficients,
# the sputter rate, and the sticking coefficient.
define_deposit_machine model=hdp material=Oxide rate=2.0 anisotropy=0.7 \
   exponent=100 redeposition=0.1 sputter_rate=0.95 s1=5.5 s2=-6 sticking=0.1

# Load a structure from a file.
define_structure file=Structures/trench.tdr

# Start the oxide deposition process.
deposit spacing={0.015 0.015 0.015} time=0.1

# Save the final structure to a file.
save
```

High-Density Plasma 2 Deposition

This example illustrates the high-density plasma 2 deposition model:

```
# Define a deposition machine that uses the hdp2 model
# and set the anisotropy, the angular distribution exponent,
# the deposited material, the rate, the redeposition coefficient,
# the sputtering coefficients, the sputter rate,
# the sticking coefficient, the sputtering distribution type,
# the sputtering distribution exponent, and the reflection coefficient.
define_deposit_machine model=hdp2 material=Oxide rate=2.0 anisotropy=0.7 \
   exponent=100 redeposition=0.1 sputter_rate=0.95 s1=5.5 s2=-6 sticking=0.1 \
   sputter_type="reflective" sputter_exponent=100 reflection=0.05

# Load a structure from a file.
define_structure file=Structures/trench.tdr

# Start the oxide deposition process.
deposit spacing={0.015 0.015 0.015} time=0.1
```

```
# Save the final structure to a file.
save
```

Low-Pressure Chemical Vapor Deposition

This example illustrates the low-pressure chemical vapor deposition model:

```
# Define a deposition machine that uses the lpcvd model
# and set the deposited material, the rate,
# and the sticking coefficient.
define_deposit_machine model=lpcvd material=Oxide rate=1.0 sticking=0.35

# Load a structure from a file.
define_structure file=Structures/trench.tdr

# Start the oxide deposition process.
deposit spacing={0.015 0.015 0.015} time=0.2

# Save the final structure to a file.
save
```

Physical Vapor Deposition

This example illustrates the physical vapor deposition model:

```
# Define a deposition machine that uses the pvd model
# and set the deposited material, the rate, and the exponent.
define_deposit_machine material=Oxide model=pvd rate=1.0 exponent=1

# Load a structure from a file.
define_structure file=Structures/trench.tdr

# Start the oxide deposition process.
deposit spacing={0.015 0.015 0.015} time=0.15

# Save the final structure to a file.
save
```

Plasma-Enhanced Chemical Vapor Deposition

This example illustrates the plasma-enhanced chemical vapor deposition model:

```
# Define a deposition machine that uses the pecvd model
# and set the anisotropy, the angular distribution exponent,
# the deposited material, the rate, and the sticking coefficient.
```

```
define_deposit_machine model=pecvd material=Oxide rate=1.0 anisotropy=0.6 \
   exponent=30 sticking=0.05

# Load a structure from a file.
define_structure file=Structures/trench.tdr

# Start the oxide deposition process.
deposit spacing={0.015 0.015 0.015} time=0.2

# Save the final structure to a file.
save
```

Simple Deposition

This example illustrates the use of the simple deposition model:

```
# Define a deposition machine that uses the simple deposition model
# and set the deposited material, the curvature, and the rate.
define_deposit_machine model=simple material=Oxide rate=1.0 anisotropy=0.3 \
   curvature=0.1

# Load a structure from a file.
define_structure file=Structures/trench.tdr

# Start the deposition process.
deposit spacing={0.015 0.015 0.015} time=0.1

# Save the final structure to a file.
save

# Extract the intersection points of the final structure
# with a line passing through the point (0.05 0.05 0.0)
# and parallel to the z-axis.
extract type=1d_cut axis=z point={0.05 0.05 0.0}
```

Spin-on-Glass Deposition

This example illustrates the spin-on-glass deposition model:

```
# Define a deposition machine that uses the spin_on model and set the deposited
# material, the viscosity, the density, the initial_thickness,
# the radial_distance, the surface_tension, the angular_velocity,
# the evaporation_rate, and the angular_position.
define_deposit_machine model=spin_on material=Oxide viscosity=0.015 \
   density=1 initial_thickness=2 radial_distance=2e4 surface_tension=27 \
   angular_velocity=4000 evaporation_rate=0 angular_position=0
```

```
# Create the initial structure and save it.
define_structure material=Silicon point_min={0 0 0} point_max={100 100 1}
define_shape type=cube name=l1 point_min={40 0 1} point_max={60 100 4}
define_shape type=cube name=l2 point_min={0 60 1} point_max={60 80 2.5}
define_shape type=cube name=l3 point_min={60 30 1} point_max={100 50 1.5}
deposit shape=l1 material=Aluminum
deposit shape=l2 material=Aluminum
deposit shape=l3 material=Aluminum
save

# Start the oxide deposition process.
deposit spacing={10 10} time=0.01 cfl=0.025

# Save the final structure to a file.
save
```

Etching

The following examples show the use of the etching models in Sentaurus Topography 3D.

Crystallographic Orientation–Dependent Etching

This example illustrates the crystallographic orientation–dependent etching model:

```
# Define a crystal etch machine, set the etching rates along the <100>,
# <110>, and <111> directions of the lattice, and set the material to be
# etched.
define_etch_machine model=crystal rate_100=1 rate_110=1.2 rate_111=0.1 \
   etchable_material=Silicon

# Create the initial structure and save it.
define_structure material=Silicon point_min={0 0 0} point_max={0.4 0.2 0.2} \
   region=Silicon_region
fill thickness=0.05 material=Photoresist
define_shape type=cube point_min={0.1 0.05 0.2} point_max={0.3 0.2 0.3} \
   name=cuboid
etch shape=cuboid
save

# Set the crystallographic orientation of the region with the material to be
# etched.
set_orientation region=Silicon_region flat_orientation={1 1 0} \
   vertical_orientation={0 0 1}

# Start the etch simulation process.
etch spacing={0.004 0.004 0.004} time=0.2
```

```
# Save the final structure to a file.
save
```

Dry Etching

This example illustrates the dry etching model:

```
# Define a dry-etch machine.
# Define the material that is redeposited and set
# the sticking coefficient and the redeposition rate.
define_etch_machine model=dry deposit_material=Anyinsulator \
   sticking=0.035 rate=0.2

# Multimaterial etch.
# Define materials and set the machine-dependent properties:
# rates and sputtering coefficients.
add_material material=Oxide rate=2.0 s1=5.5 s2=-6
add_material material=Resist rate=0.1 s1=5.5 s2=-6
add_material material=Anyinsulator rate=0.1 s1=5.5 s2=-6

# Load a structure from a file.
define_structure file=Structures/dry_etch_input.tdr

# Start etch simulation process.
etch spacing={0.015 0.015 0.015} time=0.1

# Save the final structure to a file.
save
```

Wet Etching

This example illustrates the wet etching model:

```
# Define a wet etch machine and set the etchant diffusivity and the
# distance of the source plane from the topmost point of the structure
# under process.
define_etch_machine model=wet diffusivity=1e-2 source_distance=1

# Define materials to be etched and their rates.
add_material material=Silicon rate=1

# Create an initial structure with a hole by removing a cuboid
# from a block of silicon.
define_structure material=Silicon point_min={0 0 0} point_max={2 3 2}
fill material=Photoresist thickness=0.5
define_shape type=cube name=c point_min={1 1 2} point_max={2 2 2.5}
etch shape=c
```

```
# Start etch simulation process.
etch time=0.5 spacing=0.1

# Save the final structure to a file.
save
```

Simultaneous Etching and Deposition

This example illustrates the simultaneous etching and deposition (`etchdepo`) model:

```
# Define an etchdepo machine and set the ion angular distribution exponent.
# Define the material that is redeposited and set the sticking coefficient
# and the redeposition rate.
define_etch_machine model=etchdepo exponent=100 \
   deposit_material=Anyinsulator sticking=0.035 rate=0.2

# Multimaterial etch.
# Define materials and set the machine-dependent properties:
# rates and sputtering coefficients.
add_material material=Oxide rate=2.0 s1=5.5 s2=-6
add_material material=Anyinsulator rate=0.1 s1=5.5 s2=-6

# Load a structure from a file.
define_structure file=Structures/dry_etch_input.tdr

# Start etch simulation process.
etch spacing={0.01 0.01 0.01} time=0.1

# Save the final structure to a file.
save
```

High-Density Plasma Etching

This example illustrates the high-density plasma etch model:

```
# Define a high-density plasma etch machine and set
# its ion angular distribution exponent.
define_etch_machine model=hdp exponent=15

# Multimaterial etch.
# Define materials and set the machine-dependent properties:
# rates, anisotropy, and sputtering coefficients.
add_material material=Aluminum rate=1.13 anisotropy=0.04 s1=5.5 s2=-6
add_material material=Photoresist rate=0.663 anisotropy=0.95 s1=5.5 s2=-6

# Load a structure from a file.
define_structure file=Structures/etch_input.tdr
```

```
# Start etch simulation process.
etch spacing={0.015 0.015 0.015} time=0.1

# Save the final structure to a file.
save
```

High-Density Plasma 2 Etching

This example illustrates the high-density plasma 2 etch model:

```
# Define a high-density plasma 2 etch machine and
# set its ion angular distribution exponent.
define_etch_machine model=hdp2 exponent=100

# Multimaterial etch.
# Define materials and set for each of them the machine-dependent
# properties: rates, anisotropy, sputtering coefficients,
# sticking coefficient, and sputter rate.
add_material material=Aluminum rate=1.13 anisotropy=0.04 s1=5.5 s2=-6 \
   sticking=0.5 sputter_rate=1.0
add_material material=Photoresist rate=0.663 anisotropy=0.95 s1=5.5 s2=-6 \
   sticking=0.0 sputter_rate=0.0

# Load a structure from a file.
define_structure file=Structures/etch_input.tdr

# Start etch simulation process.
etch spacing={0.015 0.015 0.015} time=0.1

# Save the final structure to a file.
save
```

Ion-Enhanced Etching

This example illustrates the ion-enhanced etching model:

```
# Define an ion-enhanced etch machine and set
# its ion angular distribution exponent.
define_etch_machine model=ion_enhanced exponent=10

# Multimaterial etch.
# Define materials and set the machine-dependent properties:
# rates, anisotropy, sputtering coefficients, and sticking coefficient.
add_material material=Aluminum rate=0.7 anisotropy=0.5 sticking=0.5 \
   s1=5.5 s2=-6
add_material material=Photoresist rate=0.5 anisotropy=0.5 sticking=0.0 \
   s1=5.5 s2=-6
```

```
# Load a structure from a file.
define_structure file=Structures/etch_input.tdr

# Start etch simulation process.
etch spacing={0.015 0.015 0.015} time=0.1

# Save the final structure to a file.
save
```

Ion-Milling

This example illustrates the ion-milling model:

```
# Define an ion-mill machine.
define_etch_machine model=ionmill

# Multimaterial etch.
# Define materials and set the machine-dependent properties:
# rates, anisotropy, and sputtering coefficients.
add_material material=Aluminum rate=0.505 anisotropy=1.0 s1=5.5 s2=-6
add_material material=Photoresist rate=0.18 anisotropy=1.0 s1=5.5 s2=-6
add_material material=Tantalum rate=0.18 anisotropy=1.0 s1=5.5 s2=-6

# Load a structure from a file.
define_structure file=Structures/etch_input.tdr

# Start etch simulation process.
etch spacing={0.015 0.015 0.015} time=0.1

# Save the final structure to a file.
save
```

PMI Simple Etching

This example illustrates the PMI simple etching model:

```
# Define a PMI simple etching machine.
define_etch_machine model=pmi_simple_etch

# Multimaterial etch.
# Define materials and set the machine properties.
add_material material=Silicon rate=0.3 anisotropy=0.3
add_material material=Oxide rate=0.1 anisotropy=0.0

# Load a structure from a file.
define_structure file=Structures/etch_input.tdr

# Start etch simulation process.
etch spacing={0.005 0.005 0.005} time=0.1 plot_interval=0.03
```

```
# Save the final structure to a file.
save
```

The source code for the PMI model is available in the file:

```
${STROOT}/tcad/${STRELEASE}/lib/sptopo3d/examples/pmi_simple_etch/
pmi_simple_etch.cc
```

Reactive Ion Etching

This example illustrates the reactive ion etch model:

```
# Define a reactive ion etch machine and set
# its ion angular distribution exponent.
define_etch_machine model=rie exponent=100

# Multimaterial etch.
# Define materials and set the machine-dependent properties:
# rates and anisotropy.
add_material material=Aluminum rate=0.7 anisotropy=1.0
add_material material=Photoresist rate=0.5 anisotropy=1.0

# Load a structure from a file.
define_structure file=Structures/etch_input.tdr

# Start etch simulation process.
etch spacing={0.015 0.015 0.015} time=0.1

# Save the final structure to a file.
save
```

Reactive Ion Etching 2

This example illustrates the reactive ion etch 2 model:

```
# Define a reactive ion etch 2 machine and set
# its ion angular distribution exponent.
define_etch_machine model=rie2 exponent=100

# Multimaterial etch.
# Define materials and set the machine-dependent properties:
# rates, anisotropy, and the sticking coefficient.
add_material material=Aluminum rate=0.7 anisotropy=1.0 sticking=0.5
add_material material=Photoresist rate=0.5 anisotropy=1.0 sticking=0.0

# Load a structure from a file.
define_structure file=Structures/etch_input.tdr
```

```
# Start etch simulation process.
etch spacing={0.015 0.015 0.015} time=0.1

# Save the final structure to a file.
save
```

This example creates a structure from the beginning and shows how to set the parameter `reflection`. Microtrenching at the bottom of the trench is formed:

```
# Define the initial bulk structure.
define_structure material=Silicon point_min= {0 0 0} \
   point_max= {0.1 0.02 0.01}

# Create an oxide and a photoresist layer.
fill material=Oxide thickness=0.15
fill material=Photoresist thickness=0.02

# Remove part of the photoresist.
define_shape type=cube point_min= {0.025 0.0 0.16} \
   point_max= {0.075 0.02 0.19} name=subtract
etch shape=subtract

# Save the initial structure.
save

# Define the rie2 machine.
define_etch_machine model=rie2 exponent=100

# Set and define the material-dependent parameters.
# Ion reflection is only considered to happen in the resist.
# No neutrals are considered.
add_material material=Photoresist rate=0.0 anisotropy=0.0 sticking=0.0 \
   reflection=0.05
add_material material=Oxide rate=1.0 anisotropy=1.0 sticking=0.0
etch spacing= {0.005 0.005 0.005} time=0.05

# Save the simulation results, appending them
# in the previously generated TDR file.
save
```

Simple Etching

This example illustrates the simple etching model:

```
# Define an etch machine using a simple model.
define_etch_machine model=simple

# Multimaterial etch.
# Define materials and set the machine-dependent properties:
# rate, anisotropy, and curvature factor.
```

```
add_material material=Silicon rate=0.5 anisotropy=0.3 curvature=0.0
add_material material=Oxide rate=0.1 anisotropy=0.0 curvature=0.0

# Load a structure from a file.
define_structure file=Structures/etch_input.tdr

# Start etch simulation process.
etch spacing={0.015 0.015 0.015} time=0.1

# Save the final structure to a file.
save
```

Tilt and Units

This example shows how to tilt the wafer using the parameters `tilt` and `rotation`. At the same time, nondefault units are set:

```
# Define a deposition machine that uses a simple deposition model and
# set the deposited material, the anisotropy, the curvature, and the rate.
# Tilt and rotation are also set for the machine.
define_deposit_machine model=simple material=Oxide anisotropy=1.0 \
   curvature=0.0 rate=0.05<um/min> rotation=90 tilt=45

# Load a structure from a file.
define_structure file=Structures/trench.tdr

# Start the oxide deposition process.
deposit spacing={10<nm> 10<nm> 10<nm>} time=60<s>

# Save the final structure to a file.
save
```

Simulations With 2D Structures

This example shows how to use Sentaurus Topography 3D to run simulations on two-dimensional structures:

```
# Define a 2D initial structure.
define_structure point_min={0 0} point_max={1 1} material=Silicon

# Fill works exactly as for a 3D structure.
fill material=Photoresist thickness=0.1

# Define a rectangle.
define_shape type=rectangle name=rect point_min={0.25 0.5} \
   point_max={0.75 1.5}

# Etch a trench geometrically, exactly as for a 3D structure.
```

```
etch shape=rect

# Define a simple deposition machine, exactly as for a 3D structure.
define_deposit_machine model=simple material=Oxide rate=1.0 anisotropy=0.7 \
   curvature=0.0

# Deposit. Note that the size of the 'spacing' parameter value must match
# the dimension of the structure.
deposit spacing={0.025 0.025} time=0.2

# Save the 2D structure to TDR.
save
```

Integration With Sentaurus Process

This example shows how to use the functionality of Sentaurus Topography 3D from within Sentaurus Process. This is a Sentaurus Process command file. Note the use of the `init` and `struct` commands instead of `define_structure` and `save`, respectively. The `topo` command of Sentaurus Process calls all the functionality related to Sentaurus Topography 3D:

```
# Initialize Sentaurus Process with a trench.
init tdr=Structures/trench.tdr

# Define a deposition machine that uses the simple model.
# Note the 'topo' prefix.
topo define_deposit_machine model=simple material=Oxide rate=2.0 \
   anisotropy=0.7 curvature=0.0

# Start the Oxide deposition process.
# Note the 'topo' prefix.
topo deposit spacing= {0.015 0.015} time=0.1

# Save the structure.
struct tdr=simple_deposition
```

Glossary

This glossary contains the most frequently used terms in the Sentaurus™ Topography 3D User Guide.

A

anisotropy
The ratio of the directional deposition or etch rate to the total deposition or etch rate, which is a combination of a directional component and an isotropic component.

C

chemical vapor deposition (CVD)
The growth of the material from a gaseous medium containing a mixture of reactants that chemically interact and convert to the desired material.

curvature-dependent deposition/etch
The deposition rate along a curved surface is a function of the local curvature, which smooths the surface as it evolves.

conformal structure
If all the regions of a structure share the same vertices and faces along region boundaries that touch each other, the structure is *conformal*. If vertices or faces on shared region borders are not part of both regions, the structure is *nonconformal*. Sentaurus Topography 3D needs conformal structures as input. Nonconformal structures can be made conformal when read into Sentaurus Topography 3D by setting the parameter `conformalize=true` in the `define_structure` command.

CVD
Chemical vapor deposition.

D

dry etching
An etching process that uses plasma gas as the reactive source.

E

etching
The process that removes materials from the wafer surface.

H

HDP
High-density plasma.

high-density plasma (HDP) deposition
A deposition process that uses high-density plasma sources to produce deposition precursors. Usually, this process results in simultaneous deposition and sputter etching of thin film on the wafer surface, due to highly energetic ions.

I

IADF
Ion-angular distribution function. It describes the angular flux distribution of ions in flux models, that is, the probability of an ion traveling at a certain angle.

ion-milling
A synonym for ion-mill etch.

ion-mill etch
An etching process purely caused by physical sputtering of the surface by highly energetic ions. The sputter yield (the number of particles knocked off the surface per incoming ion) is a function of the angle between the surface normal and the direction of the incoming ion.

L

low-pressure chemical vapor deposition (LPCVD)
The deposition environment is at such a low pressure that particle–particle collision based on gas-vapor deposition is negligible compared to particle–surface collision. The particles travel between surfaces in a ballistic trajectory.

LPCVD
Low-pressure chemical vapor deposition.

P

PECVD
Plasma-enhanced chemical vapor deposition.

physical vapor deposition (PVD)
The condensation of material from its own vapors.

plasma-enhanced chemical vapor deposition (PECVD)
A chemical vapor deposition technique that uses a plasma chemical vapor discharge to enhance the deposition characteristics.

PMC
Particle Monte Carlo.

PVD
Physical vapor deposition.

R

reactive ion etch (RIE)
An etching process in which chemically reactive ions are the major source of the reaction that removes particles from the surface.

RFM
Rate formula module.

RIE
Reactive ion etch.

S

Sentaurus Lithography
A microlithography process simulator.

shadowed element
When a surface element has no vertical line of sight to the source because another surface element blocks its view, the element is *shadowed*.

SLO
File format used by Sentaurus Lithography.

sticking coefficient
The statistical probability that an incoming precursor will stay on the surface and contribute to the deposition of thin film, contrary to bouncing back to the gas phase. It controls the conformality in a reemission process.

T

TDR
File format used for TCAD Sentaurus tools.