Process Simulator

Informative session

Saurabh Sant, Dr. sc. ETH
saurabh64sant@gmail.com

SemiVi LLC
Zollikon, Switzerland.

May 10, 2025

Overview

Table of Contents

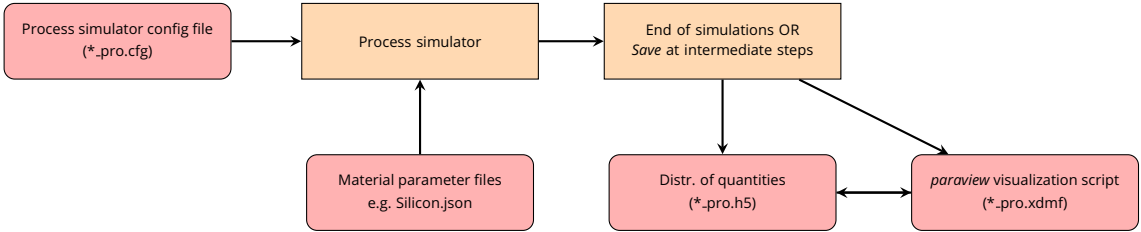
Salient features - process simulator

The process simulator performs step-by-step simulation of semiconductor fabrication process flow to create a final device.
Salient features of the process simulator are –

- The process solver can simulate etching, deposition, implantation, and annealing steps on the 2D and 3D structures.
- Gas flows can be specified in the annealing step to create oxidizing environment. In such conditions, the process solver also simulates the Silicon oxidation step.
- Implant tables are provided to extract analytic implant parameters at the given implant energy and tilt.
- Enhancement of dopant diffusivity due to the defects and the electric field can be included in the dopant diffusion by using a *three-stream model*. Simpler and faster models for the dopant diffusion are also provided.
- Python interface enables convenient scripting of the process flow.
- The process simulator is built on *the structure generator and mesher*. Therefore, python interface of that tool can be readily used in the process simulator, too.
- Pseudo-process mode enables running fast 2D and 3D process simulations by performing pure geometric operations. Thus it assists in *iterative* process-flow development.
- A graphic-user-interface is provided for convenient creation of the structure and the process-flow.

Table of Contents

Process simulator interface and config files



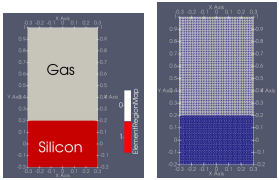
To run a process simulation...

- Create and save *process simulator config file*, *material files* (if modified), and other required files in the same folder.
- Run simulations using the following command –
  ```
  >> process process diode_pro.cfg
  ```
- If you use python scrit (preferred), run simulations using the following command –
  ```
  >> python3 diode_pro.py
  ```
- For GUI-based process-flow development, open GUI using the following command –
  ```
  >> processgen &
  ```

Table of Contents

Setting initial structure in a config file

```
Device: {
  Name = "Diode";
  MeshType = "FEM";
  Simulation = "PRO";
}

Region*RegBack: {
  RefWin = ["RefAll"];
  Material = "Gas";
}

Region*RegSi1: {
  RefWin = ["RefSi1"];
  Material = "Silicon";
}

MeshDef*MDefAll: {
  RefWin = "RefAll";
  X: {cen = 0., minspacing = 0.02,
      maxspacing = 0.02, incr = 1. }
  Y: {cen = 0., minspacing = 0.02,
      maxspacing = 0.02, incr = 1. }
}

MeshDef*MDefSi1: {
  RefWin = "RefSi1";
  X: {cen = 0., minspacing = 0.02,
       maxspacing = 0.02, incr = 1. }
  Y: {cen = 0., minspacing = 0.005,
       maxspacing = 0.005, incr = 1. }
}

Save*sav0: {
  Quantities = ["BActive"];
}
```



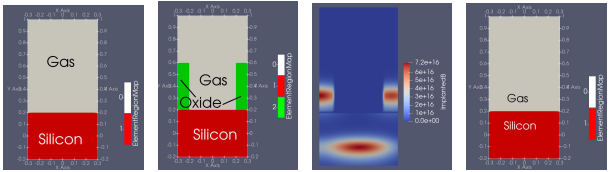
(a) Initial structure (b) Mesh

Figure: (a) Structure and (b) mesh at the beginning

- `Device`: selects the mesh-type and intent of simulation (`PRO`) .
- `RefWin`: Shapes which can be used for a variety of purposes.
- `Region`: Set `RefWin` list and the `Material`.
- `Material`: New material defined by adding a `Material.cfg` file in the working directory.
- `MeshDef`: Meshing definition followed in the specified `RefWin`.
- `DopingDef`: Doping of the vertices in the structure at the beginning.

Setting process flow in the config file

```
Deposit*dep1: {
  Material = "Oxide";
  Type = "Anisotropic";
  Thickness = 0.4;
}

RefWin*polyetMaskWin: {
  Position = ([-0.2001, 0., 0.],
              [0.2001, 0., 0.]);
  Shape = "Segment";
}

Mask*polyetMask: {
  RefWinList = ["polyetMaskWin"];
}

Etch*et1: {
  Material = "Oxide"; Type = "Masked";
  Mask = "polyetMask"; Depth = 0.5;
}

Implant*impl1: {
  Species = "Boron"; Energy = 90;
  Dose = 1E12;
}

Etch*et2: {
  Material = "Oxide"; Type = "Strip";
}
```



(a) Init (b) Masked etch (c) Implantation (d) Strip etch

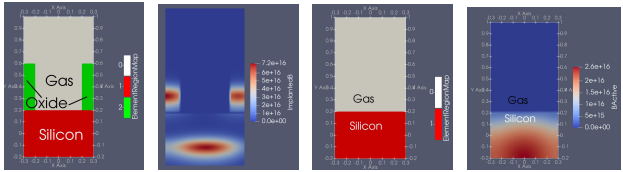
Figure: (a) Init, (b) Masked etch, (c) Implantation, (d) Strip.

- `Deposit`: Deposit a layer of `Material` of specified `Thickness` on the substrate in `Anisotropic` deposition step.
- `Mask`: Create a mask of the shape of a 1D `Segment` defined in the `RefWin`.
- `Etch`: Etch the `Material` up to the `Depth` anisotropically using `Mask` as a stencil.
- `Implant`: Perform *analytic* implantation of the given `Species` at `Energy` and `Dose`. Implant parameters are taken from the table.

Setting annealing parameters in a config file

```
SetParam*s0: {
  Command = "Silicon BActive IntNeutral
             Dstar 0 Arrhenius 0.0121 3.341";
}

SetAnnealParam*s1: {
  initstep = 1E-4; minstep = 1E-7;
  maxstep = 10.0; Pressure = 1.;
  TolRes = 1.0; TimeStepping = "BE";
  MaxIterations = 10; MaxInnerIterations = 10;
  UndampedIterations = 0;
}

Anneal*ann1: {
  Temperature = [800., 1000., 1000., 800.];
  TimeIntervals = [400., 30., 400.];
  N2 = [20., 20., 20.];
}

Save*sav4: {
  Quantities = ["BActive"];
}
```

```
"Silicon": {
  "Dopant": {
    "BActive": {
      "DiffusionModel" : "ChargedPair",
      "Int" : {
      "ChargePair":
           {"0": "Arrhenius 5.17E-26 -1.095",
            "1": "Arrhenius 1.77E-25 -1.165"}
      }
    }
  }
```



(a) Masked etch (b) Implantation (c) Strip etch (d) Annealing

Figure: Structure after (a) etch, (b) implant, (c) strip-etch, and (d) anneal.

- `SetParam`: Set material parameters using text in the `Command`. Here, change the diffusivity `Dstar` of uncharged dopant-defect complex of `BActive` and `IntNeutral` in `Silicon` to `Arrhenius 0.0121 3.341`.
- `SetAnnealParam`: Specify math parameters of diffusion simulation during annealing.
- `Anneal`: Set consecutive `TimeIntervals` and `Temperature` at the beginning of each time interval (and at the end). Set gas flows in each of the `TimeInterval`.

Table of Contents

Python interface for process simulations

```
import process as pr
import numpy as np

dioPro = pr.Process (Name="Diode",
                     ProcessStepsFile="")

xmin = -0.24; xmax = 0.24;
ymin = -0.2; ySiTop = 0.2; ymax = 1.0
posRef = np.array ([[xmin, ymin, 0.],
                    [xmax, ymax, 0.]])
dioPro.setRefWin (Name="RefAll", Shape="Rectangle",
                  Position=posRef)

posRef = np.array ([[xmin, ymin, 0.],
                    [xmax, ySiTop, 0.]])
dioPro.setRefWin (Name="RefSi", Shape="Rectangle",
                  Position=posRef)

dioPro.setRegion (Name="RegBack", RefWin="RefAll",
                  Material="Gas")

dioPro.setRegion (Name="RegSi", RefWin="RefSi",
                  Material="Silicon")

dioPro.setMeshDef (Name="MDefAll", RefWin="RefAll",
                   Xarg=[0., 0.01, 0.01, 1.],
                   Yarg=[0., 0.01, 0.01, 1.])

dioPro.setMeshDef (Name="MDefSi", RefWin="RefSi",
                   Xarg=[0., 0.01, 0.01, 1.],
                   Yarg=[-0.075, 0.0025, 0.01, 1.3])

# create Mesh
dioPro.createDeviceMesh ()
dioPro.saveMeshData ()

# DO NOT START PROCESS BEFORE CREATING DEVICE MESH
```

Argument names in *config* file and *python script* are mostly similar. Please check the manual first!

Compare `MeshDef` and `Region` defined in *config* file vs. in *python* –

```
MeshDef*MDefAll: {
  RefWin = "RefAll";
  X: {cen = 0., minspacing = 0.02,
      maxspacing = 0.02, incr = 1. }
  Y: {cen = 0., minspacing = 0.02,
      maxspacing = 0.02, incr = 1. }
}

Region*RegSil: {
  RefWin = ["RefSil"]; Material = "Silicon";
}
```

- *Equivalency*: Python interface internally runs the software commands.
- *Parametrization*: Script can be reused for a similar structure.
- Advantage of scripting: Use of `for` loops, etc.
- Python libraries (e.g. scikit or optimization libs) can be used in the script.

Python interface for process simulations

```
dioPro.deposit (Name="DepOx", Material="Oxide",
                Type="Anisotropic", Thickness=0.4)

arrTmp=np.array ([[xmin*0.6, 0., 0.], [xmax*0.6, 0., 0.]])
dioPro.setRefWin (Name="OxEtMaskWin", Shape="Segment",
                  Position=arrTmp)

dioPro.mask (Name="OxEtMask", RefWinList=["OxEtMaskWin"])

dioPro.etch (Name="EtOx", Material="Oxide", Type="Masked",
             Depth=0.5, EtchMask="OxEtMask")

dioPro.implant (Name="Impl1", ImplantSpecies="Boron",
                Energy=90, Dose=1E12)

dioPro.etch (Name="StripOx", Material="Oxide", Type="Strip")

dioPro.setParam (Name="s0", Material = "Silicon",
     Dopant = "BActive", Defect = "IntNeutral",
     ParamName = "Dstar", Charge = "0",
     ParamValue = "Arrhenius 0.0121 3.341")

numPar = { "initstep" : 1E-4, "minstep" : 1E-7,
    "maxstep" : 10.0, "TolStep" : 1.0E-1}
strPar = { "TimeStepping" : "BE"}
dioPro.setAnnealPar (Name="sl", NumericParMap=numPar,
                     StringParMap=strPar)

TempProfile = [800., 1100., 1100., 800.]
TimeIntervals = [600., 60., 600.]
Gases = {"N2" : [20., 20., 20.]}
dioPro.anneal (Name="ann1", Temperature=TempProfile,
               Time=TimeIntervals, GasFlows=Gases)

dioPro.save (Name="Sav4", PhysicalQuantities=["BActive"])

dioPro.doProcessing ()
dioPro.saveDevice ()
```

Argument names in *config* file and *python script* are mostly similar. Please check the manual first!

Compare `Deposit` and `Anneal` defined in *config* file vs. in *python* –

```
Deposit*dep1: {
  Material = "Oxide"; Type = "Anisotropic";
  Thickness = 0.4;
}

Anneal*ann1: {
  Temperature = [800., 1000., 1000., 800.];
  TimeIntervals = [400., 30., 400.];
  N2 = [20., 20., 20.];
}
```

Run the *duly completed* python script as follows –

```
>> python3 example_pro.py
```

Table of Contents

Simulating *Deposit* step

`Deposit` definition in *config* file–

```
RefWin*polyetMaskWin: {
  Position = ([-0.2001, 0., 0.],
              [0.2001, 0., 0.]);
  Shape = "Segment";
}

Deposit*dep1: {
  Material = "Oxide";
  OnMaterial = "Silicon";
  Type = "Anisotropic";
  Thickness = 0.4;
  InSituDopantConc = 1E10; // 1/cm3
  InSituDopantSpecies = "Boron";
}
```

`Deposit` definition in *python* script–

```
arrTmp=np.array ([[-0.2001, 0., 0.],
                  [0.2001, 0., 0.]])
dioPro.setRefWin (Name="OxEtMaskWin",
     Shape="Segment", Position=arrTmp)

dioPro.deposit (Name="DepOx", Material="Oxide",
  OnMaterial = "Silicon", Type="Anisotropic",
  Thickness=0.4, InSituDopantConc = 1E10,
  InSituDopantSpecies = "Boron")
```

`Deposit` step defines deposition of the material onto the other material. It takes the following arguments.

- `Material`: Material to be deposited.
- `Thickness`: Thickness of the deposited layer.
- `Type`: Deposition type e.g. `Anisotropic`, `Isotropic`, `Polygonal`.
- `DepWin`: Specifies RefWin. In `Polygonal` deposition, material layer is set in the area confined by this RefWin.

Selective deposition and material growth can be simulated by adding the following arguments.

- `OnMaterial`: If specified, then the `Material` layer is deposited only on the exposed surface of `OnMaterial`.
- `InSituDopantSpecies`: If specified, a uniform doping conc set by `InSituDopantConc` of this species is added to the deposited layer.

Simulating *Etching* step

`Etch` definition in *config* file–

```
RefWin*polyetMaskWin: {
  Position = ([-0.2001, 0., 0.],
              [0.2001, 0., 0.]);
  Shape = "Segment";
}

Mask*polyetMask: {
  RefWinList = ["polyetMaskWin"];
}

Etch*et1: {
  Material = "Oxide"; Type = "Masked";
  Mask = "polyetMask"; Depth = 0.5;
}
```

`Etch` definition in *python* script–

```
arrTmp=np.array ([[xmin*0.6, 0., 0.],
                  [xmax*0.6, 0., 0.]])
dioPro.setRefWin (Name="OxEtMaskWin",
    Shape="Segment", Position=arrTmp)

dioPro.mask (Name="OxEtMask",
    RefWinList=["OxEtMaskWin"])

dioPro.etch (Name="EtOx", Material="Oxide",
    Type="Masked", Depth=0.5,
    EtchMask="OxEtMask")
```

`Etch` step defines etching of the material layer exposed to `Gas`. It takes the following arguments.

- `Material`: Material to be deposited.
- `Depth`: Thickness of the deposited layer.
- `Type`: Etch type e.g. `Anisotropic`, `Isotropic`, `Masked`, or `Strip`.
- `EtchMask`: Specifies a `Mask` object. In `Masked` etch, material layer etched in the region where mask is defined.

Various Etch-types are described below –

- `Anisotropic`: Exposed area is etched vertically, as if the surface is *shifted* downwards by set `Depth`.
- `Isotropic`: Exposed area is etched both vertically and laterally with equal rate.
- `Masked`: Specified mask is used for etching.
- `Strip`: Any exposed regions of the given `Material` are completely removed. Useful for removing mask materials, such as `Oxide` or `Photoresist`.

Simulating *Implantation* step

Defining `Implant` in *config* file–

```
Implant*impl1: {
  Species = "Boron"; Energy = 90;
  Dose = 1E12;
}
```

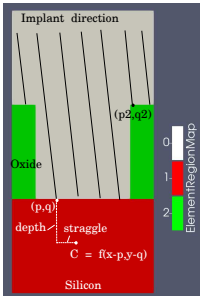
Defining `implant` in *python* script–

```
dioPro.implant (Name="Impl1",
  ImplantSpecies="Boron",
  Energy=90, Dose=1E12)
```

`Implant` step defines ion implantation on the exposed surface of the substrate.

- `Species`: One of the `Boron`, `BF2`, `Phosphorus`, `Arsenic`, or `Antimony` can be specified.
- `Energy`: Implant energy in `keV` is set.
- `Dose`: Implant dose in the units of cm^{-2} is set.
- `Tilt`: Implant tilt in `degree` is set.
- `Table`: If the implanted ion distribution parameters are not specified, then the distribution moments corresponding to the specified implantation energy, tilt, etc. are taken from the predefined implant tables.

Analytic Implantation:



(a) Analytic distribution

Figure: (a) Definition of the points (p, q) and (x, y) for defining analytic dopant distribution.

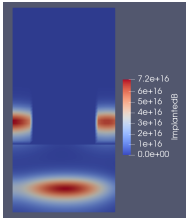


Figure: Analytic implant profile.

The point-response distribution function $F(x, y, p, q)$ is distribution by an ion beam on a single point.

$$F(x, y, p, q) = f_p(y - q) \cdot f_l(x - p) \tag{1}$$

$f_p(y)$ – distribution along beam direction
$f_l(x)$ – distribution perpendicular to the beam.
Adding distribution by each ion beam at each vertex –

$$C(x, y) = N_d \int_{\Omega_{\text{gas}}} F(x, y, p, q) dpdq \tag{2}$$

Analytic implantation profiles and parameters

Analytic profiles and parameters:
Primary distributions ($f_p(y)$).

- Gaussian distribution:

$$f_p(y) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y - R_p)^2}{2\sigma^2}\right) \quad (3)$$

  R_p and σ is set by `mu` and `sigma` parameters.

- Pearson distribution: The profile parameters R_p, σ, γ, and β are set by `mu`, `sigma`, `gamma`. and `beta`.

- Dual-Pearson distribution:

$$f_p(y) = \alpha \cdot f_{\text{head}}(y) + (1 - \alpha) \cdot f_{\text{tail}}(y) \quad (4)$$

  For the head of the distribution $f_{\text{head}}(y)$, parameters are set by `mu`, `sigma`, `gamma`. and `beta`. Parameters for Pearson profile of the tail are set by `mu2`, `sigma2`, `gamma2`. and `beta2`.

Lateral Straggle ($f_l(x)$):

$$f_l(x, y) = \frac{1}{\sqrt{2\pi}\sigma_l(y)} \exp\left(-\frac{x^2}{2\sigma_l^2(y)}\right) \quad (5)$$

$\sigma_l(y)$ is specified by using `lat.sigma` and `lat.sigma2`.

Implant table format:
Specify implant table in *Taurus* format as follows.

1. First line corresponds to the implant species information.
   `<type> <species> <mater>`
   Here, `<type>` can be `Implant` or `Damage`.
2. Next, specify the independent parameters in the implant table, which can be– `energy` (`keV`), `tilt` (`degrees`), `rotation` (`degrees`), or `dose` (cm^{-2}). In the implant table, first *n* columns correspond to the listed independent parameters.
3. Next columns list analytic distribution parameters corresponding to the listed independent parameters. e.g. for Pearson profile, list
   `mu sigma lat.sigma lat.dSigma gamma beta`

Example implant table is as follows –

```
Implant Phosphorus Silicon
energy
10 0.0139 0.0069 0.0080 0 0.641 3.68617127
20 0.0252 0.0119 0.0139 0 0.549 3.50333967
30 0.0368 0.0166 0.0195 0 0.459 3.35183727
\end{verbatim}
```

Analytic implant profile parameters are obtained from *implant tables*, if they are not provided in `Implant` definition in the config/python file.

Simulating *Annealing* step

Defining Anneal step:
Defining `Anneal` step in *config* file–

```
Anneal*ann1: {
  Temperature = [800., 1100., 1100., 800.];
  TimeIntervals = [600., 60., 600.];
  N2 = [20., 20., 20.];
}
```

Defining `Anneal` step in *python* script–

```
TempProfile = [800., 1100., 1100., 800.]
TimeIntervals = [600., 60., 600.]
Gases = {"N2" : [20., 20., 20.]}
dioPro.anneal (Name="anni", Temperature=TempProfile,
               Time=TimeIntervals, GasFlows=Gases)
```

- `TimeIntervals`: A list of time-intervals (in seconds) in a temperature ramp cycle. OR
- `Time`: A list of time values (in seconds) at the end of each time interval.
- `Temperature`: A list of temperature values starting with the initial chamber temperature followed by temperature values at the end of each time interval.
- Gas flows: Gas names can be listed followed by a list of gas flow values at each time interval. Currently following gas names are recognized by the process simulator – `H2`, `O2`, `N2`, `H2O`, `N2O`, `HCl`, and `Cl2`.

Chamber gas reactions:
The following reactions in the chamber are taken into account by the simulator.

$$2H_2 + O_2 \longrightarrow 2H_2O \text{ and } O_2 + 4HCl \longrightarrow 2H_2O + Cl_2 \quad (6)$$

Gas flows of the product gases are determined from the reacting gases depending on which gas is the limiting ingredient. If $f^{\text{ext}}_{O_2} > 0.5 f^{\text{ext}}_{H_2}$,

$$f^{\text{int}}_{H_2} = 0 \text{ and } f^{\text{int}}_{O_2} = f^{\text{ext}}_{O_2} - 0.5 f^{\text{ext}}_{H_2} \text{ and } f^{\text{int}}_{H_2O} = f^{\text{ext}}_{H_2O} + f^{\text{ext}}_{H_2}$$

else,

$$f^{\text{int}}_{O_2} = 0 \text{ and } f^{\text{int}}_{H_2O} = f^{\text{ext}}_{H_2O} + 2 f^{\text{ext}}_{O_2} \text{ and } f^{\text{int}}_{H_2} = f^{\text{ext}}_{H_2} - 2 f^{\text{ext}}_{O_2}$$

Partial pressure of gases in the chamber is calculated as follows:

$$p_{\text{gas}} = p \cdot \frac{f^{\text{int}}_{\text{gas}}}{\sum_{\text{all-gases}} f^{\text{int}}_{\text{gas}}} \quad (7)$$

p : Chamber pressure set by `Pressure` in annealing step.
$f^{\text{int}}_{\text{gas}}$: Gas flows *after* gaseous reaction has taken place.

Dopant/defect diffusion during annealing

Dopant/defects diffuse during high temperature annealing leading to their distribution changing over time. This is simulated on FEM grid in annealing simulation. Following dopant diffusion models can be used.

- `ChargedPair` model: Time evolution of the dopant concentration as well as interstitial and vacancy concentration by calculating the flux of the respective quantities.

$$\frac{\partial C_A}{\partial t} = -\nabla\cdot\vec{J}_A \text{ and } \frac{\partial C_{X^{all}}}{\partial t} = -\nabla\cdot\vec{J}_A - \nabla\cdot\vec{J}_X - R_{IV} \text{ where } J_A = -\left(\sum_{c,X} D_{AXc}\left(\frac{n}{n_i}\right)^{-c}\right)\times\left(\frac{C_{X^0}}{C^*_{X^0}}\nabla C_A^+ + C_A^+\nabla(\frac{C_{X^0}}{C^*_{X^0}}) + C_A^+(\frac{C_{X^0}}{C^*_{X^0}})\nabla \ln \right) \tag{8}$$

Time evolution of defects needs to model their recombination R_{IV} as follows,

$$R_{IV} = K_{IV}(C_I \cdot C_V - C_I^* \cdot C_V^*) \tag{9}$$

Here, C_I^* and C_V^* are equilibrium conc of interstitials and vacancies at annealing temperature.

- `ChargedFermi` model: Time evolution of the dopant species conc, taking into account that charged dopant species diffuse at different rates.

$$\frac{\partial C_A}{\partial t} = -\nabla\cdot\vec{J}_A \text{ where } \vec{J}_A = -\left(\sum_{c,X} D_{AXc}(\frac{n}{n_i})^{-c}\right)\left(\nabla C_A^+ + C_A^+\nabla \ln\left(\frac{n}{n_i}\right)\right) \tag{10}$$

Interstitials and vacancies are assumed to be in equilibrium at annealing temperature.

- `Constant` model: Time evolution of dopant species conc, ignoring different diffusivities of different charged species.

$$\frac{\partial C_A}{\partial t} = -\nabla\cdot\vec{J}_A \text{ where } \vec{J}_A = -D^*\partial C_A^+ \tag{11}$$

Annealing boundary conditions

Following boundary conditions can be activated in the simulation of dopant diffusion process –

- `HomNeumann`: No fluxes or transfers across the interface are assumed. $J_{A/X} \cdot \hat{n} = 0|_{\Omega}$. Here, $\vec{J}_X$ – defect flux, $\hat{n}$ is the unit normal to the interface.
- `Continuous`: Both concentration and fluxes of dopants or defects normal to the interface are set to be equal.
- `Natural`: Flux of defects normal to the interface is given by the following expression.

$$\vec{J}_X \cdot \hat{n} = k_s \left(1 + k_{Rat}(\frac{|V_{ox}|}{V_{Scale}})^{k_{pow}} P_0^{k_{ppow}}\right)(C_{X0} - C^*_{X0}) - G_{ox} \tag{12}$$

Here, k_s is the surface recombination rate, G_{ox} is the generation rate of the defect X due to the oxidation process, P_0 is the oxygen partial pressure, $|V_{ox}|$ is the local oxidation rate, and C^*_{X0} is equilibrium defect conc.

- Default BC at the Oxide-Silicon interface. *This BC is valid only for defects.*
- Can model interstitial injection into Si substrate during oxidation. i.e. oxidation-enhanced diffusion (OED).

- `Segregation`: Dopant flux normal to the interface is given by,

$$\vec{J}_X \cdot \hat{n} = k_{\text{transfer}} \left(C_A^a - \frac{C_A^b}{k_{\text{segregation}}}\right) \tag{13}$$

where, C_A^a is the concentration of the dopant A on one side of the interface and C_A^b is that on the other side of the interface.

- Default BC for dopants at the Oxide-Silicon interface. *This BC is valid only for dopants.*
- Can model segregation of dopants into the oxide during high temperature oxidation.

- `Dirichlet`: This BC allows to specify a fixed concentration of the dopant or the defect species at the interface.
  - To mimic in-diffusion of dopants from the dopant source.
  - To model interstitial injection from Silicon substrate.

Note: This BC can only be activated at the interface between any material and the gas.

Modeling oxidation

Deal-Grove model:

- Oxidation rate at the surface of Silicon wafer is modeled by Deal-Grove model.

$$\frac{dx_{ox}}{dt} = \frac{B}{2x_{ox} + A} \tag{14}$$

Here, x_{ox} is the oxide thickness at any time t during annealing, A and B are oxidation parameters.

- *Quadratic rate constant* $B(T)$ depends on the diffusion of gas in oxide. It is a material property, set by–

```
SetParam*s0: {
  Command = "Oxide <gas> <param> <val>";
}
```

- *Linear rate constant* $\frac{B}{A}(T)$ depends on *reaction rate* of O_2 at oxide-Si interface. It is an interface property set by–

```
SetParam*s0: {
  Command = "Oxide_Silicon <gas> <param> <val>";
}
```

  - `<gas>` is one of the oxidizing gases – `H2O`, `O2`, or `N2O`.
  - `<param>` is the param name – `Bh`, `Bl` or `BAh`, `BAl`.
  - `<val>` is an `Arrhenius` expression.

Incorporating oxide layer:

- Deal-Grove Eq. 14 is implemented as an update equation for the oxidation rate in time-stepping iterations of annealing simulation.

$$\frac{x_{ox}^{i}[q+1] - x_{ox}^{i}[q]}{t[q+1] - t[q]} = \sum_{g} \frac{B_g(T[q], p[q])}{x_{ox}^{i}[q] + A_g(T[q], p[q])} \tag{15}$$

- Time integration in Eq. 15 is performed at each of the vertices at the exposed oxide-Silicon interfaces or gas-Silicon interface.
- Once annealing step is over, the oxide layer is incorporated into the structure by creating a new oxide region.
- Note: *If oxidation is performed on bare Si surface, the above procedure would ignore dopant segregation into the oxide, due to the absence of initial oxide layer.*

Table of Contents

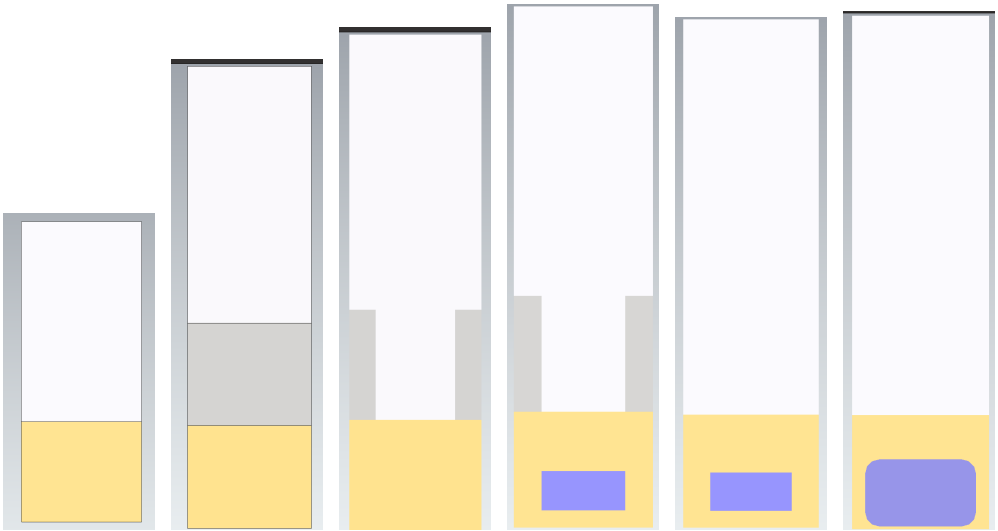
Pseudo-process mode

Pseudo-process mode enables running fast 2D and 3D process simulations by performing pure geometric operations. Thus it assists in *iterative* process-flow development.

- Enable pseudo-process mode in initialization line using `PseudoProcessMode=True`
- Generate initial structure exactly as described before. *Do not perform meshing, since pseudo-process mode does not allow it.* Preprocess device by calling `preprocessDevice()`.
- Add all the process steps exactly as before.
- In `anneal` step, specify estimated `DiffusionLength` by anneal process in μm.
- Save the structure after *every process step* as a STEP file by calling `saveDeviceAsSTEP(.)`. Give step filename as an input.
- Visualize the structure using any STEP viewer, e.g. `cadassistant`.
- Along with the structure, 'doped zones' are also stored in step file allowing quick visualization.

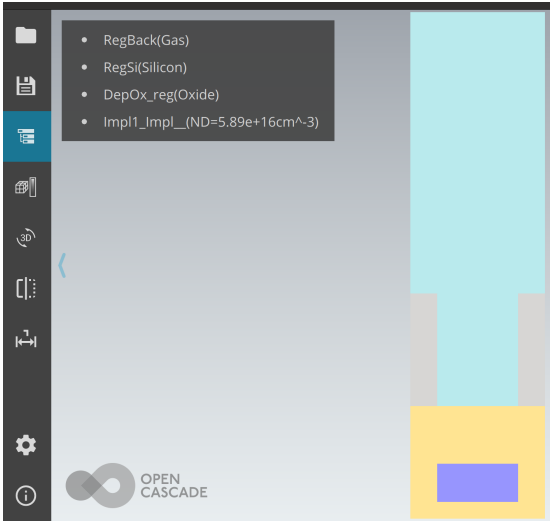
```
dioPro = pr.Process (Name="Diode", ProcessStepsFile="",
          MeshType="FEM", PseudoProcessMode=True)
...
...
dioPro.preprocessDevice ()
dioPro.saveDeviceAsSTEP("InitStructure.stp")

# start process steps
dioPro.deposit (Name="DepOx", Material="Oxide",
                Type="Anisotropic", Thickness=0.4)
dioPro.saveDeviceAsSTEP("AfterDepOx.stp")

dioPro.setRefWin (Name="OxEtMaskWin", Shape="Segment",
                  Position=arrTmp, NumericParams={},
                  NumericListParams={})
dioPro.mask (Name="OxEtMask", RefWinList=["OxEtMaskWin"])

dioPro.etch (Name="EtOx", Material="Oxide",
 Type="Anisotropic", Depth=0.5, EtchMask="OxEtMask")
dioPro.saveDeviceAsSTEP("AfterEtOx.stp")

dioPro.implant (Name="Impl1",Type="Ana",
 ImplantSpecies="Boron", Energy=90, Dose=1E12)
dioPro.saveDeviceAsSTEP("AfterImplB.step")

dioPro.etch (Name="StripOx", Material="Oxide", Type="Strip")
dioPro.saveDeviceAsSTEP("AfterStripOx.step")

TempProfile = [800., 950., 950., 800.]
TimeIntervals = [20., 5., 20.]
Gases = {"N2" : [20., 20., 20.]}
dioPro.anneal (Name="ann1", Temperature=TempProfile,
               Time=TimeIntervals, GasFlows=Gases,
               NumericParMap={"DiffusionLength" : 0.05})
dioPro.saveDeviceAsSTEP("AfterAnneal")
```

Psuedo-process mode - results



(a) Initial Structure (b) After oxide deposition (c) After masked oxide etch (d) After implantation (e) After oxide mask removal (f) After annealing

Psuedo-process mode - results

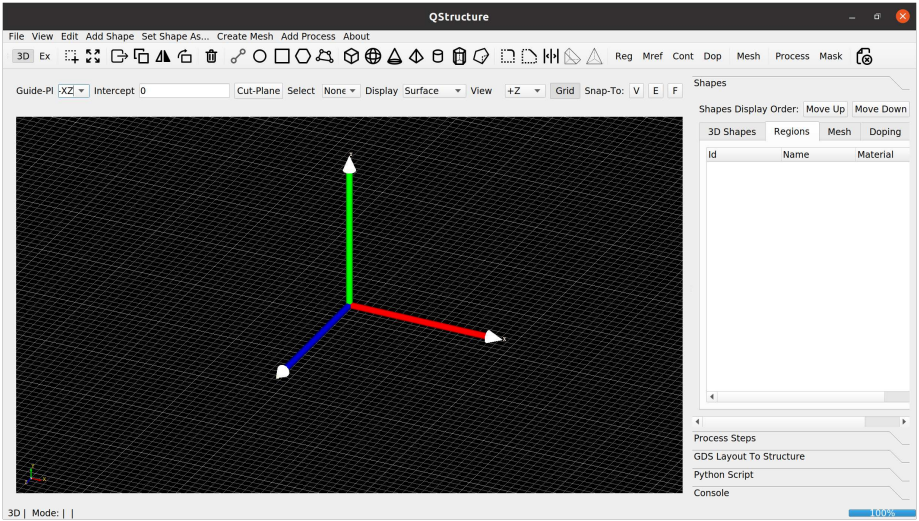


(a) cadassistant window

- In STEP file, material names appear in the bracket after region names.
- Material coloring: white - background gas, gray - oxide, yellow - Silicon.
- Doped zone coloring: Blue and its shades - p-doped. Red and its shades - n-doped.
- Click on the region/doped zone name to highlight it.
- ‘cadassistant’ allows you to take cross-section view of the 3D device.

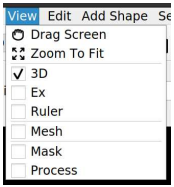
Table of Contents

GUI - Process window

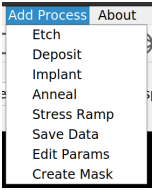


(b) Process window

GUI - Add Process Menu



(c) View menu



(d) Add Process menu

GUI and the menus of `processgen` are identical to those of `structuregen`. Users are requested to refer to the presentation/manual on structure generation for detailed information.

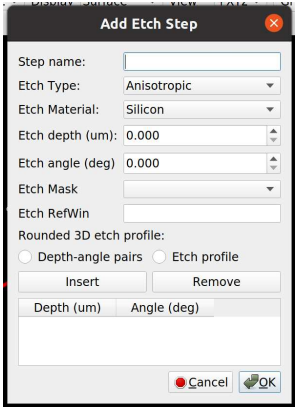
There are some additional capabilities in `processgen` menu as given below.

- File → Open : A structure '*.qstr' file can be opened to create intial structure for process simulations. Or a process '*.qprc' file can be opened to load previously stored `processgen` state.
- View → Process : Activate `Process` mode after generating initial structure.
- View → Mask : Activate `Mask` generation mode after generating initial structure.

A new Add Process menu lists functions to add process steps *only after* `Process` mode is activated.

- `Etch` : Add etch process to process-flow.
- `Deposit` : Add deposit process to process-flow.
- `Implant` : Add implantation step.
- `Anneal` : Add anneal step.
- `Save Data` : Add a placeholder to 'save data'.
- `Edit Params` : Add a placeholder to 'edit process parameters'.
- `Create Mask` : Create a mask from 2D `RefWins` in XZ plane.

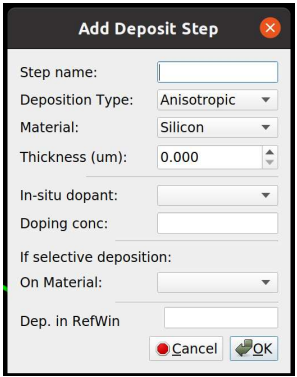
GUI - Add Etch Process



(e) Etch Dialog-box

- `Name` : Set unique name of the step.
- `Etch Type` : Specify etch-type, e.g. Isotropic, Anisotropic.
- `Etch Material` : Set etch material for selective etch.
- `Etch Depth` : Set etch depth in μm.
- `Etch Angle` : Set etch angle in degrees.
- `Etch Mask` : For masked etch, select predefined etch mask. If no mask is set, then unmasked etch is performed.
- `Etch RefWin` : If RefWin name is specified here, etch is performed *only* in the volume inside this `RefWin`.
- `Rounded 3D Etch Profiles` : To etch a 3D shape using an etch profile, select which profile-type you would like to input. Then input the profile in the table below.

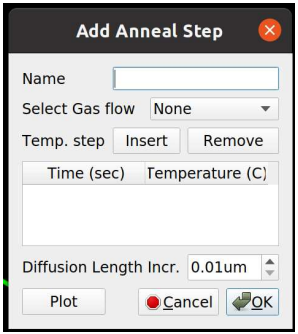
GUI - Add Deposit Process



(f) Deposit Dialog-box

- `Name` : Set unique name of the step.
- `Deposit Type` : Specify deposit-type, e.g. Isotropic, Anisotropic.
- `Deposit Material` : Set etch material for selective etch.
- `Deposit Thickness` : Set thickness of deposit layer in μm.
- `In situ Dopant` : Set whether the deposit layer is *in situ* doped by specifying the dopant.
- `Doping conc` : Set doping concentration of the dopant.
- `Deposition on Material` : If selective deposition is to be done, select material on which selective deposition is to be performed.
- `Deposit in RefWin` : If RefWin name is specified here, deposition is performed *only* in the volume inside this `RefWin`.

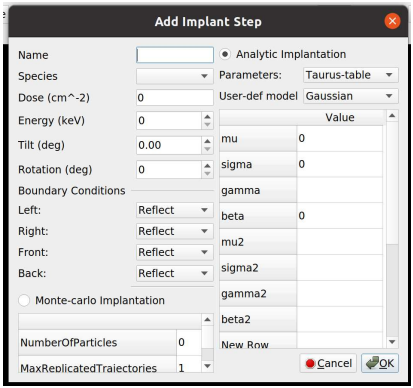
GUI - Add Anneal Step



(g) Annealing Dialog-box

- `Name` : Set unique name of the step.
- Insert : Insert a blank row at the end of the table.
- Remove : Remove selected row from the table.
- Temperature Ramp : Set temperature ramp in the table by specifying time (seconds) and temperature at that time.
- `Select Gas Flow` : Select gas. A new column with the gas name will appear. Then specify gas flow at each time-point in that column.
- `Diffusion Length Incr` : Specify approx diffusion length in μm due to current annealin step.

GUI - Add Implant Step



(h) Implant Dialog-box

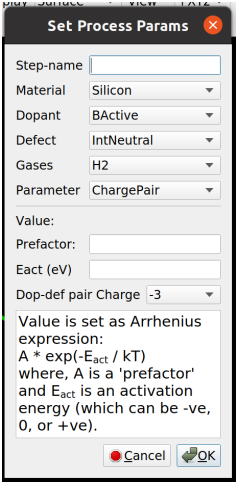
- `Name` : Set unique name of the step.
- `Species` : Select species to be implanted.
- `Dose`, `Energy`, `Tilt`, `Rotation` : Specify dose (1/cm^3), energy (keV), tilt and rotation (degrees)
- `Boundary Conditions` : Set left, right, front, and back boundary conditions. Available options `Reflect`, `Extend`, `Periodic`.
- Select implantation method – `Monte Carlo`, `Analytic`.
- For `Monte Carlo`, specify parameters such as number of particles, etc.
- For `Analytic`, specify implant-table from which to read and interpolate the parameters. Alternately, `User Defined` parameters can be specified in the table below.

GUI - Create Mask

(i) Create Mask Dialog-box

- `Name` : specify unique mask-name.
- Select desired `RefWins` from the list of suitable RefWins.
- For 3D process, 2D shapes in XZ plane are suitable.
- For 2D process, 1D segments along X direction are suitable.

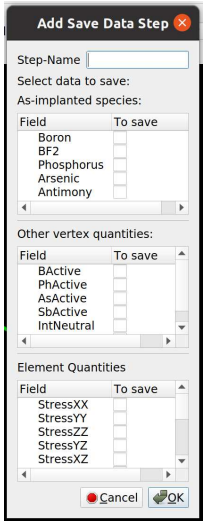
GUI - Set Parameters



(j) Set Parameters Dialog-box

- `Name` : specify unique step name.
- `Material` : Select material whose parameter is to be changed
- `Dopant`, `Defect`, `Gases` : Select the entities whose parameters are to be changed. Any one or two quantities need to be set.
- `Parameter` : Once entities are set, corresponding parameters are updated in the list.
- Set Arrhenius by specifying `prefactor` and `activation energy`. If a number is to be set, then $E_{\text{act}} = 0$.
- Specify `charge` on the dopant-defect pair species.

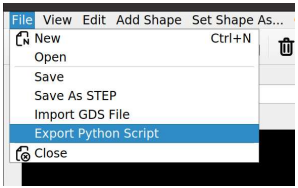
GUI - Save Data



(k) Set Save Data Dialog-box

- `Name` : specify unique step name.
- Select `As Implanted Species`, `Vertex Quantities` which need to be saved.

GUI - Run process simulations



(l) Export Python Script

- Alternately, you can export python script to a text file by File → Export Python Script and run the script from the terminal. (Preferred)

Table of Contents

Conclusions

- The process solver can simulate etching, deposition, implantation, and annealing steps on 2D/3D structures.
- Gas flows can be specified in the annealing step to create oxidizing environment. In such conditions, the process solver also simulates the Silicon oxidation step.
- Implant tables are provided to extract analytic implant parameters at the given implant energy and tilt.
- Enhancement of dopant diffusivity due to the defects and the electric field can be included in the dopant diffusion by using a *three-stream model*. Simpler and faster models for the dopant diffusion are also provided.
- Python interface enables convenient scripting of the process flow.
- The process simulator is built on *the structure generator and mesher*. Therefore, python interface of that tool can be readily used in the process simulator, too.
- Pseudo-process mode enables running fast 2D and 3D process simulations by performing pure geometric operations. Thus it assists in *iterative* process-flow development.
- A graphic-user-interface is provided for convenient creation of the structure and the process-flow.

Table of Contents

Easy-to-install licenses

For ordering a license, along with the name and the organization details, please also provide –

- For the node-locked licenses: Ethernet mac address of the client machine on which the software will run.
  OR
- For the server licenses: Ethernet mac address of the server machine at the client organization.

If you purchased one or more node-locked licenses, you will receive the following license file by secured email.

1. NodeLocked License_<id>_<Info>.lic, where <id> stands for license id and <Info> stands for customer identification in short.

Copy the license file to `/var/local/oesoft/licenses/` on the machine whose mac-address has been provided and change its access rights to `777`.

If you purchased one or more server licenses, you will receive the following license file by secured email.

1. Server License_<id>_<Info>.lic

Copy the license file to `/usr/share/oesoft/licenses/` on the *server machine* whose mac-address has been provided.

The End

Questions? Comments?