TiberCAD flow-chart

Meshing
Modeling

GMSH, ISE-TCAD

Mesh file

Input file

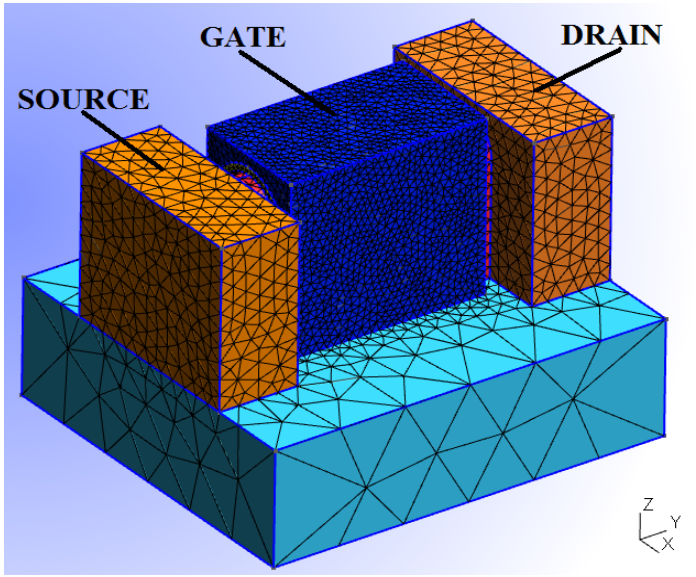
Solvers

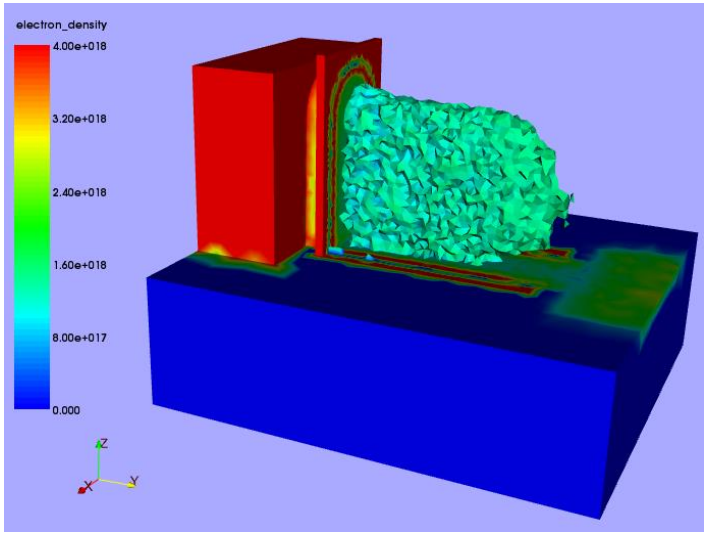
TiberCAD

Output

Visualization
Post-processing

(paraview, tecplot)





GMSH Input file

```
Point(1) = {-25,0,0,0.5};
Point(2) = {0,0,0,0.002};
Point(3) = {25,0,0,0.5};
Line(1) = {1,2};
Line(2) = {2,3};
```

***Geo* file for GMSH: geometrical model**

```
Physical Line("p_side") = {2};
Physical Line("n_side") = {1};
```

Physical entities: *mesh regions* to which external program can refer

```
Physical Point("cathode") = {3};
Physical Point("anode") = {1};
```

Boundary regions: *n-1* dimension, for boundary conditions (contacts)

GMSH Input file

```
lc = 0.001;
```

Definition of a ***variable***

```
Point(1) = {0,0,0,lc};
```

Point is defined by a list of four numbers: three coordinates (X, Y and Z), and a ***characteristic length*** (***lc***) that sets the target element size at the point

The ***distribution of the mesh element sizes*** is then obtained by interpolation of these characteristic lengths throughout the geometry.

GMSH Input file

```
Line(1) = {1,2} ;
Line(2) = {3,2} ;
Line(3) = {3,4} ;
Line(4) = {4,1} ;
```

A ***straight line*** is defined by a list of point numbers.
Here the line 1 starts at point 1 and ends at point 2

```
Line Loop(5) = {4,1,-2,3} ;
```

A line loop is a list of connected lines

```
Plane Surface(6) = {5} ;
```

A surface is a list of line loops

GMSH Input file

Plane Surface(127) = {126};

……..

Surface Loop(128) = {127, 119, 121};

Volume(129) = {128};

A ***surface loop*** is a list of plane surfaces (“shells”)

A ***Volume*** is a list of surface loops

GMSH Input file

Physical Point(1) = {1,2} ;

Physical Line(“MyLine”) = {1,2,4} ;

Physical Surface("My_surface") = {6} ;

Physical Volume ("My_volume") = {…} ;

Physical entities will group elements belonging to several elementary entities by giving them a common ID (a ***region name***).

This ***region name*** will be referred to in tiberCAD input file

GMSH Input file

Extrusion

```
Extrude {0, 0, 10} {Surface{11}; }
```

extrudes the surface 11 along the z axis and automatically
creates a new volume (as well as all the needed points, lines and surfaces

```
Characteristic Length {103, 105, 109} = lc * 3;
```

The following command permits to manually assign a characteristic length to some of the new points

GMSH Input file

Mesh Extrusion

```
Extrude {0,0,h} {Surface{6}; Layers{ 7}; }
```

by specifying ***'Layers'*** in ***Extrude*** command instead of only extruding the geometry, we extrude also the 2D **mesh** along z axis

```
Extrude {0,0,h} {Surface{6}; Layers{ {8,2}, {0.5,1} };}
```

2 layers in this case, the first one with 8 subdivisions and the second one with 2 subdivisions, both with a height of h/2

```
Extrude {0,0,h} {Surface{6}; Layers{ {8,2}, {0.5,1} }; Recombine;}
```

the resulting mesh can be recombined into prisms with the parameter ***Recombine***

GMSH Input file

Mesh Extrusion

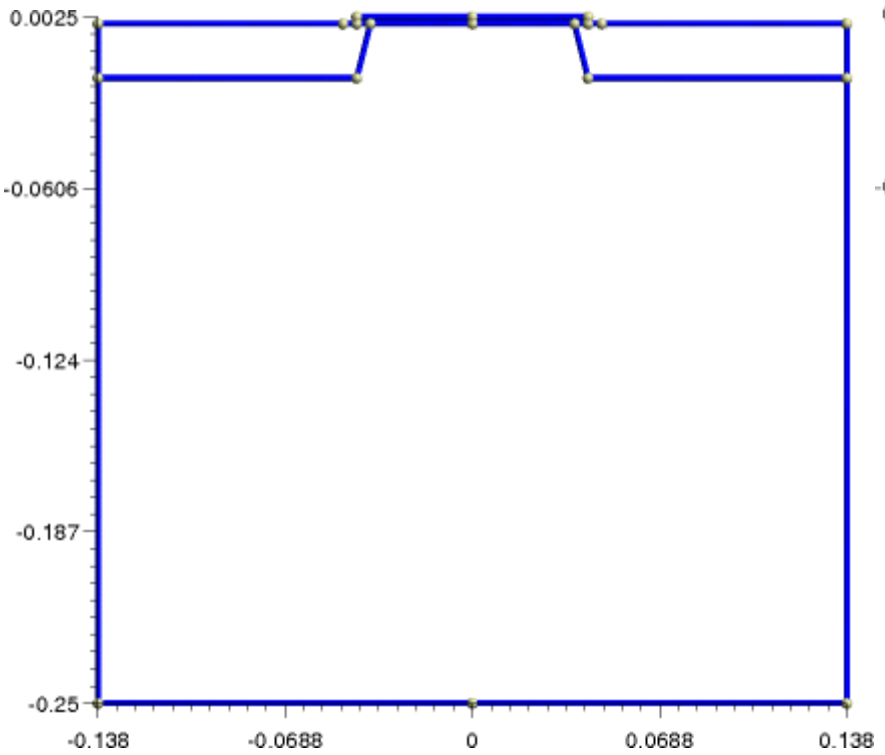
```
Out_list[] = Extrude {0,0,h} {Surface{6}; Layers{ {8,2}, {0.5,1} }; Recombine;};
```

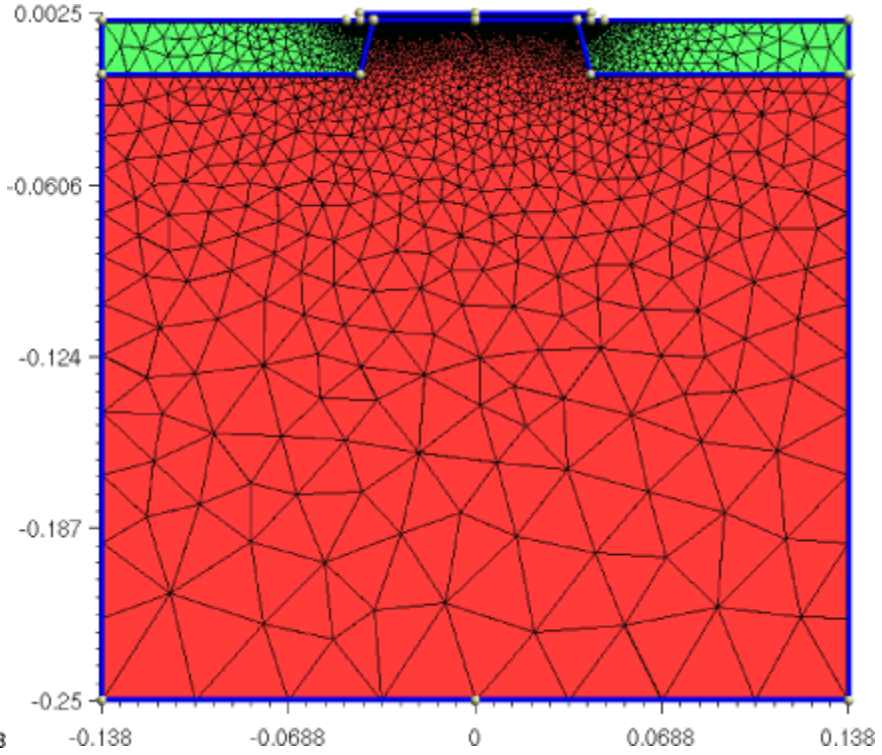
we can retrieve the volume number by using the return value (a list) of the ***Extrude*** command. This ***list*** contains the "top“ extruded surface (in out[0]), the newly created volume (in out[1]) and the ids of the lateral surfaces (in out[2], out[3], ...)

```
Physical Volume(101) = {1, 2, out_list[1]};
```

we can then use the volume obtained from the extrusion through the list *out_list*

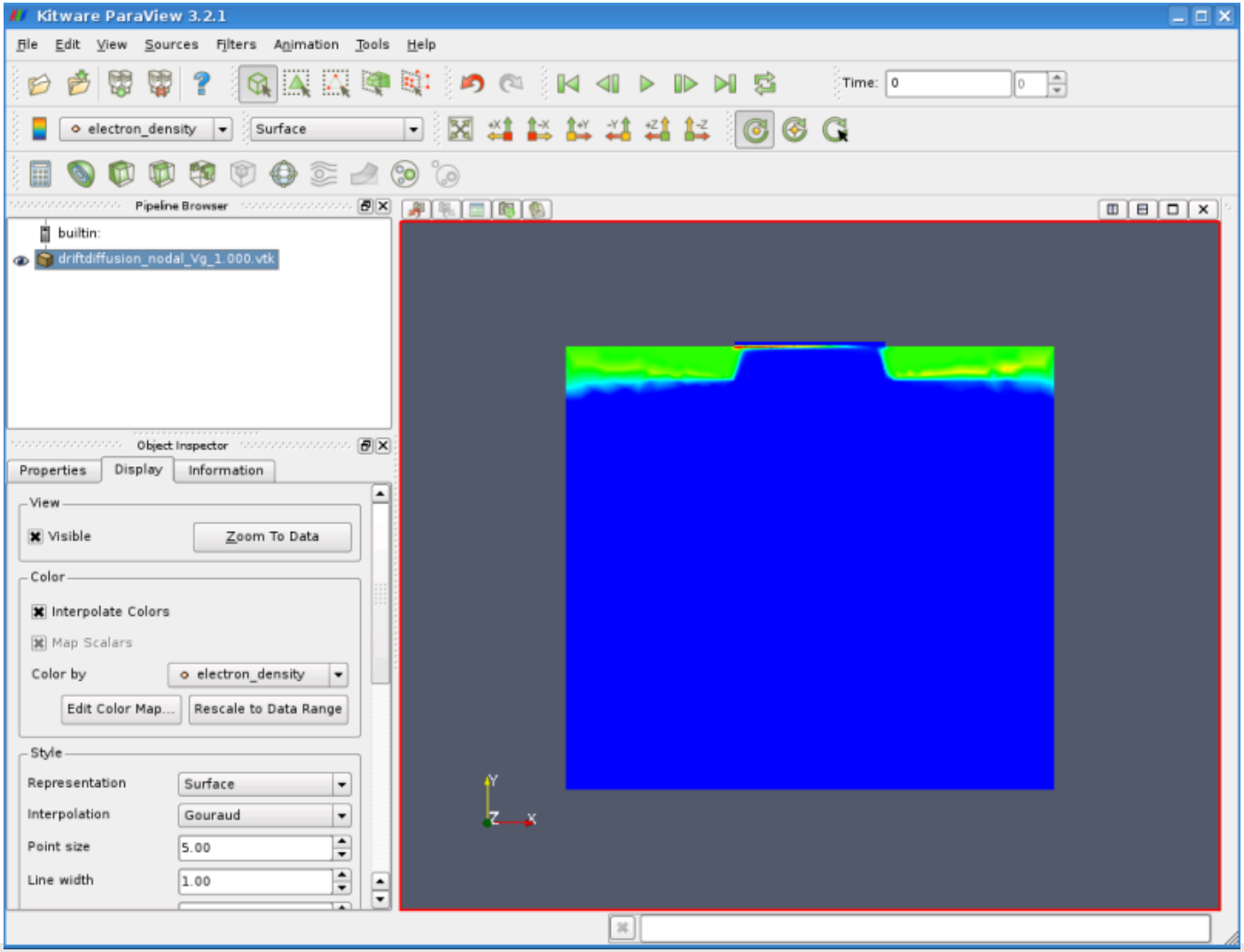
Modeling / Meshing





gmsh mosfet.geo -2 -o mosfet.msh

Output Visualization: Paraview



Input file structure

```
Device
{
meshfile = InGaAs_1D.msh

Region  n_side
{
  material= Si
  Doping
    {
     density = 1e18
     type = donor
    }

}
```

***Device* description:**
mesh file reference,
regions definition,
material,
doping,
etc.

Input file structure/2

```
Module driftdiffusion
{
   name= dd
   regions = all
   plot = (Ec, Ev, Eg, ElField,...)

Physics
  {
    strain_simulation = strain
    polarization (piezo, pyro) {}
    recombination srh {}
  }

  Contact cathode
  {
    type = ohmic
    voltage = 0.0
  }
```

***Modules* definition: associated regions, physical models, boundary conditions (contacts), etc.**

Input file structure/3

```
Simulation
{
  dimension = 1
  temperature = 300
  solve =  (strain, driftdiffusion,
     quantum_electrons,quantum_holes )

  resultpath =  output

  output_format = grace

}
```

***Simulation* definition: simulation flow, output control**

Block
Device

in GMSH units are **not** defined !

identifier of a GMSH region or user defined name

to collect several GMSH regions

```
Device
{
    meshfile = mesh_file
    mesh_units = 1e-9 (default 1e-6= [μm])

    Region  region_name
    {
      [mesh_regions = (list_gmesh_regions)]

      material = material_name
      [x = alloy molar_fraction]
```

Block
Device

A common material, crystal structure and growth directions can be defined for all device regions by defining them outside of the **Region** blocks.

```
Device
{
  [material = material_name]

  Region  region_name
  { material = material_name
    x-growth-direction = (1,0,0)
    y-growth-direction = (0,1,0)
    z-growth-direction = (0,0,1)
```

x-growth-direction = *(1,0,-1,0)*
y-growth-direction = *(-1,2,-1,0)*
z-growth-direction = *(0,0,0,1)*

Bravais vectors with Miller indices for wurtzite (4-tuple) or zincblende (3-tuple) crystal along the x, y and z directions.

Block
Device

```
Region  region_name
{
    ……
     Doping
{
     density = doping conc. [cm -3 ]
     type = donor|acceptor
     level = dopant energy level [eV ]
}
```

Doping associated to a Region

Block
Device

Cluster groups different regions with possibly different materials in a logical unit with a common name.
(used e.g. for a definition of a quantum region)

```
Cluster cluster_name_1
{
  regions = (list_of_regions)
}

Cluster cluster_name_2
{
  regions = (list_of_regions)
}
```

The regions of the device associated to the Modules simulation will be indicated by means of ***Region*** and ***Cluster*** names.

Block ***Simulation***

To define general parameters such as the temperature and the output settings for the actual calculation to be run, that is the process-flow of the simulation.

```
Simulation
{
  temperature = temp
  verbose = info level

  solve = (list of simulations)
   resultpath = output path
  output_format = vtk|grace
}
```

Process-flow of the simulation:

- ❑ Simulations will be performed in the order defined in ***solve***
- ❑ Special simulations are ***sweep*** and ***selfconsistent***