Simulation Standard

Engineered Excellence

A Journal for Process and Device Engineers

Using VICTORY Process for Plasma Etching Simulations

Introduction

VICTORY Process, the 3D process simulator now includes a module for plasma etching. The module is designed to simulate plasma etching processes at the feature-scale size. The simulation in the reactor-scale region is out of the scope of VICTORY Process. All transport characteristics data (as functions of reactor parameters needed for the feature-scale simulation) are modeled by user definable C-functions and are supplied to the module. The plasma etching simulator shares many elements with the standard physical etching /deposition module such as:

- The topology of a given feature, defined by its layers of various materials, as described by level set functions given on Cartesian meshes
- All fine details of the structure are captured on imbedded finer meshes, and automatic and/or manual adaptive mesh refinement is available
- Particles fluxes, etch rates for different materials and types of particles involved in the process are modeled by appropriate functions, which are implemented in the user accessible C-Interpreter model library
- All feature topological changes caused by the plasma etching process during a given time are captured by the solution of the corresponding partial differential equations acting on the level set functions
- A feature's structure is automatically updated after the simulation time has expired

However, there are some important differences:

Plasma reactor generates two different types of particles: ions and neutrals, characterized by different flux distribution functions. The immediate consequences are:

- Each plasma species has its own etching rates, sticking efficiencies and other parameters, given as single values or functions, for every material present in the simulated structure
- At a given surface point, all plasma particles react with material in a complex way, and the resulting etching rate has to be properly modeled

All of these differences are reflected in the syntax of input deck commands.

The plasma etching module is capable of simulating plasma etching processes by giving the user access to various models of plasma particles' fluxes, etching rates and reactions. This paper shows the most important details and the relevant input deck commands.

Define the Plasma Etching Process Characteristics

Two different etching agents are recognized and implemented in the plasma etching module: ion and neutral particles, representing all ions and all neutrals produced in a plasma reactor, respectively. Distinguishing ions and neutrals is important in implementing and defining etching rates, sticking efficiencies, fluxes and other parameters. Note that in the current version of the plasma etching module all ions (as well as all neutrals) in the plasma are represented by a single type of ion / neutral particle.

In order to run simulations, a user issues a list of commands to initialize various parameters and access algorithms. One such command defines how a single plasma particle (ion or neutral) reacts with materials in a given

Continued on page 2 ...

INSIDE

How to Mesh in 2D TCAD 5

Hints, Tips and Solutions 14

structure. It is similar to the chemical definition in the physical etching/deposition module, but with extended capabilities. The command is PLASMAETCHPROPERTIES, and is typically used as follows:

```
PlasmaEtchProperties plasmaname="my_ions" \
  particles="ions" \
  material="silicon" rate=0.45 sticking=1.0 \
  material="photoresist" rate=0.0005 \
  sticking=1.0 density=1.
```

for ions and similarly for neutrals. The command line defines the rate and the sticking efficiency for all materials in a structure, related to a single active particle acting on a flat horizontal wafer surface, and the the particle density in the reactor ambient.

The particles parameter is mandatory. It specifies the type of plasma particle (ion/neutral) related to the list of material properties defined by this command. Any plasma etching simulation must have two PLASMAETCHPROPERTIES commands, one for each type of plasma particle.

A typical example of syntax to define the particles' flux distributions in space is:

```
PlasmaFlux name="my_ionflux" \
  particles="ions" pri_c \
  function="plasmaEtchIons" pridep1="theta" \
  pridep2="surfacematerial" pripar1=2.0
```

for ions, and similarly for neutrals. Again, the particles parameter is mandatory and it defines the type of plasma particle the flux distribution definition refers to. In any simulation two such PLASMAFLUX commands must be issued for the two types of plasma particles. The most frequently used distribution functions, namely cosine for neutrals and cosine-to-power for ions, are implemented in the C-Interpreter model library and have been used for the simulations shown in this paper, but any others distribution function can be defined by the user.

Within the code the calculation of a total particle's flux at a given surface point is carried out by integrating the flux distribution function along all directions from the point to all points in space at infinity (reactor domain). Any direction which is obstructed by the structure is ignored. This means that its contribution to the total flux is zero. The calculated flux is normalized with a value of a flux at the highest surface point of the structure where all directions toward the space are unobstructed. Thus the flux value for a surface point can be interpreted as a probability for active particles to reach the surface point. Since there are ions and neutrals in the plasma, the probability is multiplied with normalized densities:

- *Nion / (Nion + Nneut)* for ions and
- *Nneut / (Nion + Nneut)* for neutrals, respectively.

Here *Nion* and *Nneut* represent ion and neutral particles' average densities in the plasma. These parameters are provided by the command PLASMAETCHPROPERTIES by means of the parameter density as shown above.

Neutral particles play an important role in surface chemistry in plasma etching processes. The sticking efficiency for such particles is usually < 1 or even $\ll 1$, which has to be reflected in the flux calculation. Therefore, VICTORY Process calculates secondary flux values for surface points for any type of plasma particles. For any surface point the algorithm takes into account the flux contribution (reflection and re-emission) from all other surface points, provided that the line connecting a given point and any other surface point is not obstructed by the structure. All surface points thereby serve as micro-sources of bounced off plasma particle(s). Those micro-sources are taken into account together with the ambient (plasma reactor) source. The syntax of how to switch on this optional modeling feature will be shown later in this paper.

Besides the particle fluxes, the total etching rate at a surface point, which describes a complex surface chemistry, must be calculated. This rate is modeled as a function of various plasma particles' parameters, such as

- Flux
- Sticking coefficient
- Rate and other values

Various models found in literature are implemented in the open C-Interpreter model library. The command to be issued to select a specific surface reaction model for the plasma etching simulation is PLASMAETCHREACTION. The syntax for examples presented in this paper reads:

```
PlasmaEtchReaction name="my_plasma_etch" \
  c_function="plasmaetch_haugutheta1" \
  dep1="Flux" dep2="Rate" dep3="Sticking" \
  dep4="Surfacematerial" \
  par1=4.0 par2=20000.0 par3=0.075 par4=5.0
```

In this example the rate model defined by the C-Interpreter model library function "plasmaetch_haugutheta1" is chosen. Here the dependencies (dep1, dep2, dep3 and dep4) are properties at the local surface point position, while the parameters (par1, par2, par3, par4) are global modifiable surface reaction model parameters. Note that the dependencies "Flux", "Rate" and "sticking" refer to both ions and neutrals. Therefore any of these dependencies in the function's definition must have two entries. In other words two values are passed to the surface reaction model function for each of the dependencies. The simulator automatically supplies the appropriate values whenever the surface reaction model function is called.

The flux models and the etch rate calculation models described above are linked by the command PLASMAETCHTOPOGRAPHYMODEL. For the examples in this paper we link the previously defined models by:

```
PlasmaEtchTopographyModel
  name="my_plasma_etch_model" \
  reactionmodel="my_plasma_etch" \
  fluxmodel1="my_ionflux" \
  fluxmodel2="my_neutralflux"
```

Note that both ion and neutral flux models must be selected by this command to setup an appropriate plasma etching model.

Once all the process models and characteristics are defined as shown above, one can issue the command PLASMAETCH to run the actual plasma etching simulation. In the cases presented here, one example reads:

```
PlasmaEtch model="my_plasma_etch_model" \
  ionname="my_ions" \
  neutralname="my_neutrals" \
  time=10. autoref \
  NeutralSecSource="CONST"
```

By this command all above defined models and parameters are put together. As shown in the example, it is possible to switch on the secondary flux calculation, both for ion and/or neutral particles, by the parameter NeutralSecSource, followed by the name of micro-sources shape function. In the case of ions, the parameter IonSecSource should be used. It only makes sense to switch on the secondary flux calculation if the sticking efficiency for a chosen particle is not equal to 1, because the existence of bounced-off particles can be simulated. It follows from the fact that for any surface point the amount of flux available for re-emission is proportional to (1 – sticking efficiency).

Examples

The properties of the new VICTORY Process module for plasma etching simulation are demonstrated in the examples shown further in the text. They are based on results published in a paper by Hauguth et al, Microelectronic Engineering 86 (2009) 976-978. Not all parameters required for setting up the plasma etching simulation could be obtained from the publication directly, but could be adjusted by means of calibration. Several parameters were varied in ranges reported in literature, and their effects are presented in the simulation results shown here. In all these cases, some parameters were kept constant, such as:

- Etching rates for materials and single plasma particles
- Sticking efficiency and density for ions
- Global parameters defining flux and rate functions
- Total simulation time

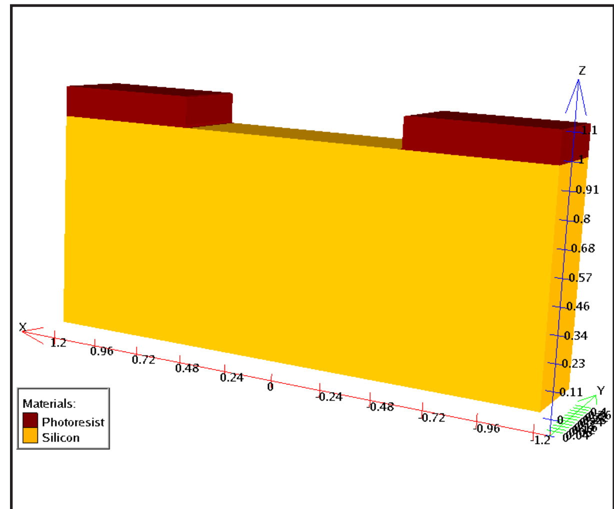

Figure 1. Initial structure before starting the plasma etching simulation.

Note that the etch rates for the mask material are very small in comparison with the silicon ones.

The parameters which were varied are sticking efficiency and density for neutrals.

The common initial structure for all cases is shown in Figure 1.

The structures after performing plasma etching simulation with various parameter combinations are shown in Figures. 2-5.

The densities of ion and neutrals particles are the same for the case shown in Figure 2. Although in real experiments usually *Nneut* >> *Nion*, this case is used for comparison with cases which are much closer to realistic situations.

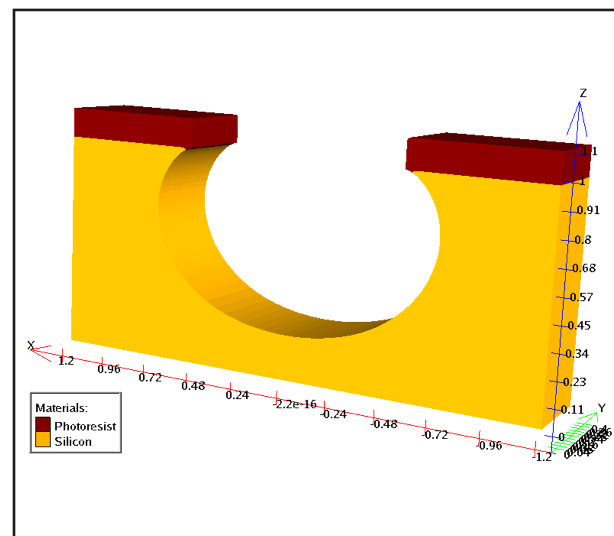

Figure 2. Structure after performing the plasma etching simulation by assuming *Nneut* = *Nion*.

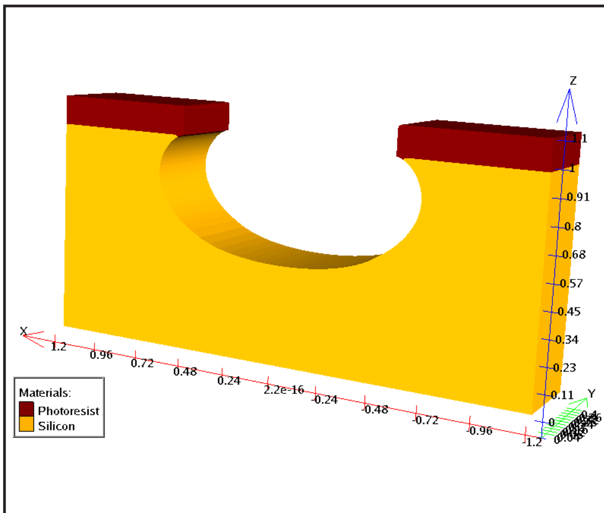

Figure 3. Structure after performing the plasma etching simulation by assuming weakly ionized plasma (*Nnet/Nion*=10000) and a neutral sticking efficiency of 1.0.

In the cases shown in Figures 3, 4 and 5 a high ratio between the neutral and ion flux was applied (Nnet/Nion = 10000). This corresponds to typical plasma etching conditions of low ionization degree. The sticking efficiency for the neutral particles was varied for these cases:

- Figure 3 - neutral sticking efficiency : 1.0
- Figure 4 - neutral sticking efficiency : 0.5
- Figure 5 - neutral sticking efficiency : 0.2

There is a significant qualitative difference in the cases shown in Figures 4 and 5, where the sticking efficiency < 1, in comparison to the case shown in Figure 3. Only these two cases show the experimentally observed under-etching below the mask. Because these domains are invisible for the main source (reactor domain) of active particles, their appearance is only due to bounced-off active particles from the rest of the structure.

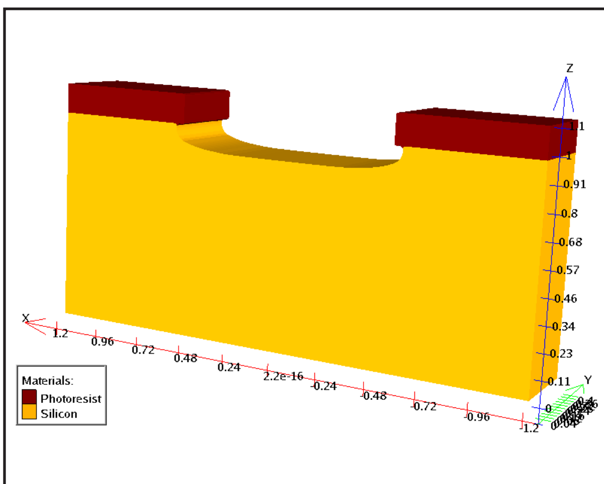

Figure 5. Structure after performing the plasma etching simulation by assuming weakly ionized plasma (*Nnet/Nion*=10000) and a neutral sticking efficiency of 0.2.

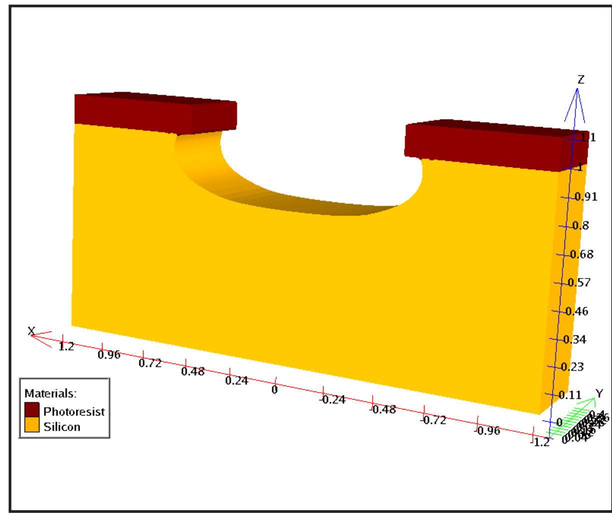

Figure 4. Structure after performing the plasma etching simulation by assuming weakly ionized plasma (*Nnet/Nion*=10000) and a neutral sticking efficiency of 0.5.

Conclusion

The plasma etching module in VICTORY Process is capable of simulating complex processes within a feature scale for three-dimensional structures. The technological parameters from the reactor scale can be included by means of transport characteristics (plasma particles' distribution functions, etching rates etc.) via user's definable C-functions stored in an open model library. The distinction between plasma ion and neutral particles is fully implemented into the code, which allows complex models to be applied. One of the main results of the example presented in this paper is the module's ability to predict the etching of domains covered by the mask, which is caused by bounced-off active particles.

How to Mesh in 2D TCAD

Introduction

Meshing for process in ATHENA or electrical characterisation in ATLAS is one of the biggest challenges that a user faces. Poor meshing in both ATHENA and ATLAS can lead to simulation failures. Poor meshing can also lead to inaccurate electrical results, which is potentially even worse. The effect that the mesh can have on the electrical results has been illustrated in [1]. An optimum mesh will have a sufficient number of points to ensure accuracy yet it will not have an excessive number of points as this will lead to an increase in simulation time.

For a more fundamental discussion on meshing in ATHENA an excellent article can be found in [2]. Further information on base mesh definition in ATHENA can be found in [3].

In this article a more applied approach is considered with discussion given to a selection of mesh enhancement techniques and situations the user might face. A very simple structure is used throughout this article to highlight these techniques.

The deck used to create the basic structure used in this article is shown in Figure 1a. Images of the structure at key processing steps are shown in Figures 1b – e. Firstly, a trench region is etched from the silicon (Figure 1b), a liner oxide is then deposited (Figure 1c), followed by a poly trench re-fill (Figure 1d), finally the poly silicon is cleaned off (Figure 1e).

Please note that extremes and simplifications have been used in this article so as to clearly highlight and introduce a basic range of techniques.

Simple Half Cell Structure

```
Deckbuild V3.40.8.R - Half_Tr.in, dir: /home/davidg/
File  View  Edit  Find  Main Control  Commands  Tools

go athena

#TITLE: Simple Half Trench

# X-Mesh with uniform spacing
line x loc=0.00 spac=0.1
line x loc=3 spac=0.1

# Y-Mesh with uniform spacing
line y loc=0.00 spac=.2
line y loc=3.00 spac=.2

# Define initial starting silicon
init orientation=100 c.phos=1E13

# Etch out the trench. We are not limited to 4 points.
# Here we have given a slight chamfer to the trench corner
etch silicon  start x=0  y=-1
etch  cont  x=1 y=-1
etch  cont  x=1 y=1.9
etch  cont  x=0.9 y=2
etch  done  x=0 y=2

# Deposit the trench liner. Note Athena is left to
# auto select the number of DIVISIONS
dep oxide thick=0.1

# Deposit the poly refill
dep poly thick=2

# Etch back the poly
etch poly thick=2

# Save out the structure
struc outf=Std_Half_Tr.str

quit
                                                    ATHENA
```

Figure 1a. ATHENA syntax used to create the basic half-cell structure.

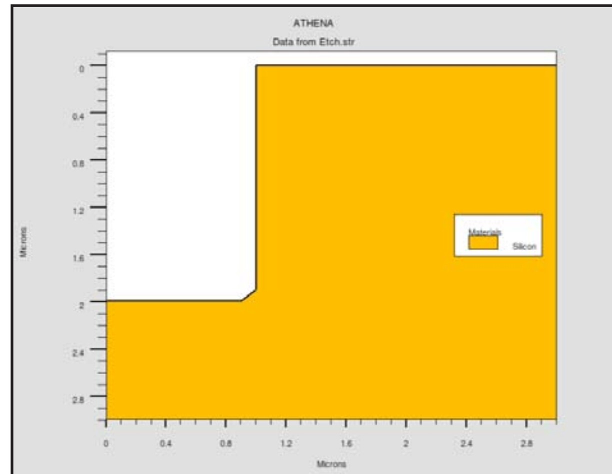

Figure 1b. Starting structure with trench etch.

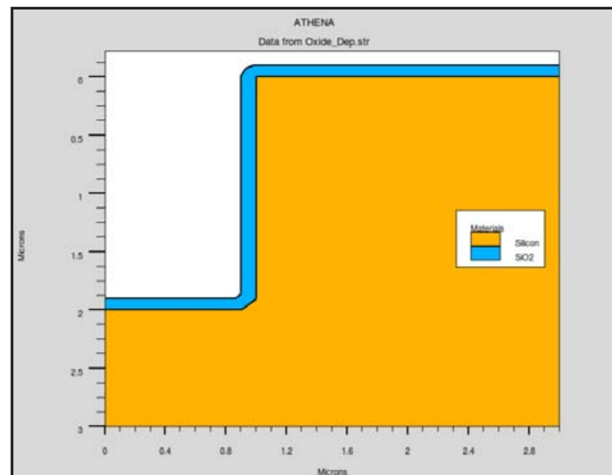

Figure 1c. Deposit of 1um oxide liner.

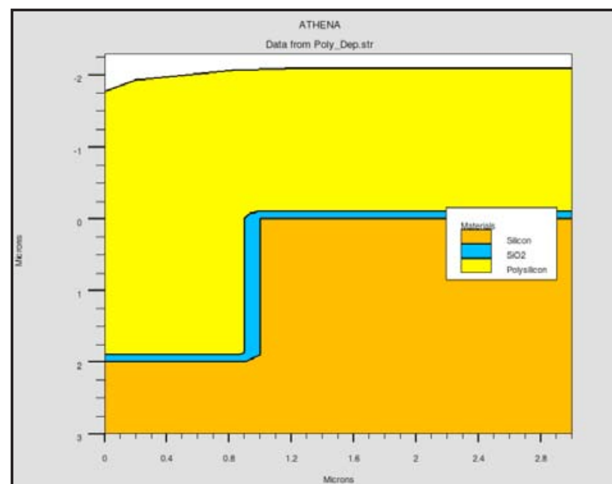

Figure 1d. Poly trench refill.

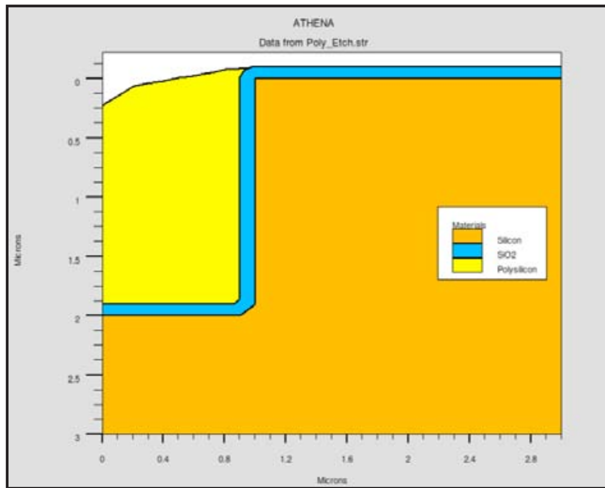

Figure 1e. Poly clean back.

Mesh Definition in layers Deposited in ATHENA

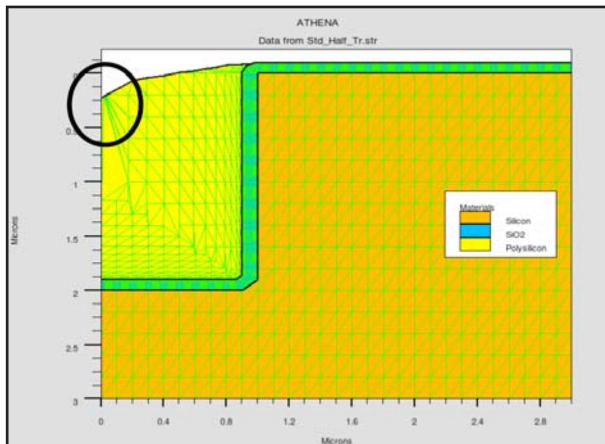

Figure 2. Auto-mesh definition of the deposited poly layer.

In Figure 2 we have allowed ATHENA to automatically choose the number of mesh layers in the deposited layer. This is achieved by not using the DIVISIONS flag on the relevant DEPOSIT statement. Overall the mesh is satisfactory except in the top left corner, as highlighted. Although it is not severe in this instance, a number of triangles are sharing a common vertex or point. This is not desirable and can subsequently cause numerical instabilities. Furthermore, you can see that some of the triangles in this region are very thin which can also contribute to numerical instabilities. Ideally the ratio of the longest to shortest edge should be no greater than 10 [2].

The issues with the structure highlighted in Figure 2 can be resolved by increasing the mesh density in the deposited layer, as shown in Figure 3. The mesh density in a deposited layer is set using the DIVISIONS flag on the DEPOSIT statement as in:

```
DEPOSIT POLY THICK=2 DIVISONS=31
```

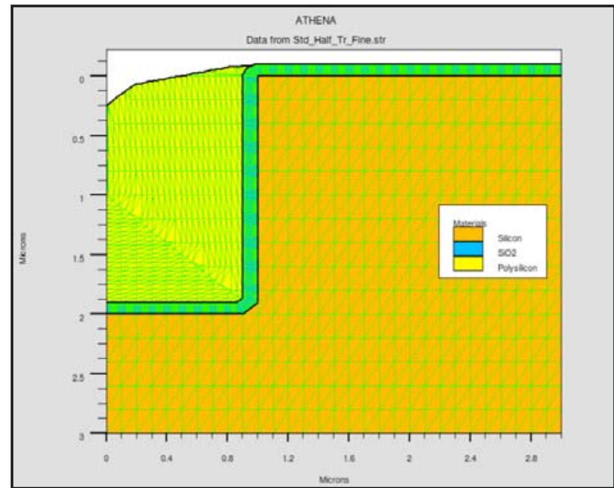

Figure 3. Increased number of mesh points in the deposited poly, enhancing the mesh quality.

Comparing the two structures, it is clear that all of the triangles in Figure 3 show a good ratio between side lengths and no node has a high number of triangles attached to it. Although the mesh issues are now resolved it should be noted that the mesh density is now somewhat excessive.

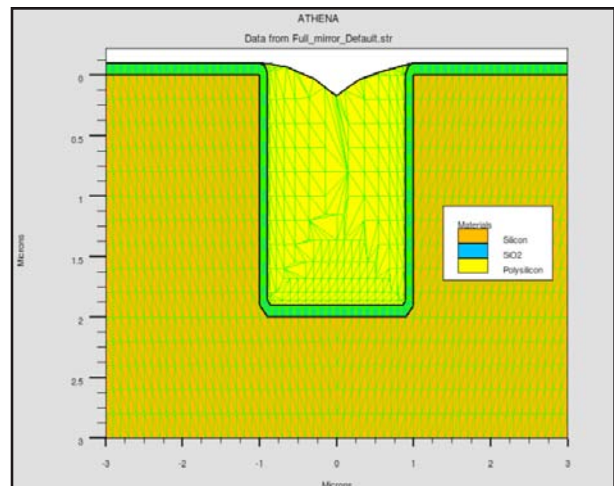

Figure 4. Half-cell mirrored in ATHENA with default DIVISIONS.

Usually the most numerically efficient method is to simulate a half-cell. However to enhance the numerical stability in certain circumstances it is better to simulate a full cell. In the previous images we have shown a half-cell. Shown in Figure 4 is a full cell. It is very simple to turn a half-cell structure into a full cell structure in ATHENA using the command:

```
# Using the LEFT flag mirrors on the
  left vertical axis,

# RIGHT mirrors on the right vertical
  axis

STRUCTURE MIRROR LEFT
```

You can use this command at any point in the deck. The structure shown in Figure 4 uses the automatic DIVISIONS value set by ATHENA in the poly deposition step. Consequently the structure is displaying a number of narrow triangles and triangles on a common vertex as was observed in Figure 2. Although a full cell can be used to overcome a numerically challenging step, the structure can easily be converted back to a half-cell at any point to speed up the simulations.

There are two different ways we can do this. The first is simply using ATHENA syntax:

```
ETCH LEFT P1.X=0 NOEXPOSE
```

This statement will remove all of the materials to the left of X=0. The second option is to use DevEdit (either syntax or via the GUI) and cut the structure in half.

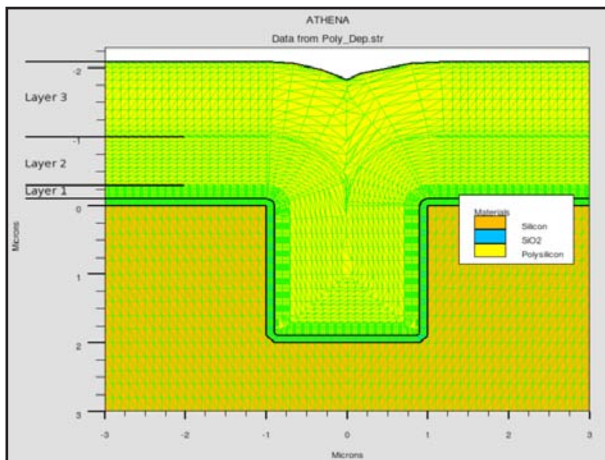

Figure 5. Multi layer deposition.

In the simulation environment there is no penalty if you make multiple discrete depositions of the same material. In situations with complex topography it is sometimes useful to split the deposition of a thick layer into a number of separate steps. This also has the advantage that different mesh densities can be used on different layers. Previously the polysilicon layer was deposited in a single step using:

```
DEPOSIT POLY THICK=2 DIVISONS=31
```

Using the multiple deposition technique we would use a sequence of statements similar to:

```
DEPOSIT POLY THICK=0.2 DIVISONS=11

DEPOSIT POLY THICK=0.7 DIVISONS=11

DEPOSIT POLY THICK=1.1
```

So the sum of the deposited layers is equal to the total thickness specified when just a single statement was used. Such a technique is illustrated in Figure 5. The three discrete layers of polysilicon can clearly be seen. The first, Layer 1, is relatively thin with a high mesh density. Layer 2 is somewhat thicker with a lower density and Layer 3 is thicker still with its mesh density further reduced.

Mesh Relaxation in ATHENA

The focus so far has been ensuring that the mesh is dense enough to accurately form the structure. It is always worth remembering that an excessive number of grid points will increase computation time. To optimise the mesh we could completely re-mesh the structure in DevEdit (which will be covered in the following section) or the mesh density can be reduced in defined areas simply using ATHENA syntax. To relax a mesh in ATHENA the following command is used:

```
RELAX X.MIN=1.5 X.MAX=3 Y.MIN=0.5
Y.MAX=3 SILICON
```

This relaxes the mesh in a box, the limits of which are defined by the X/Y max/min co-ordinates.

Figure 6a shows the original structure. The mesh density has been increased to accentuate the relaxation that is shown in Figure 6b.

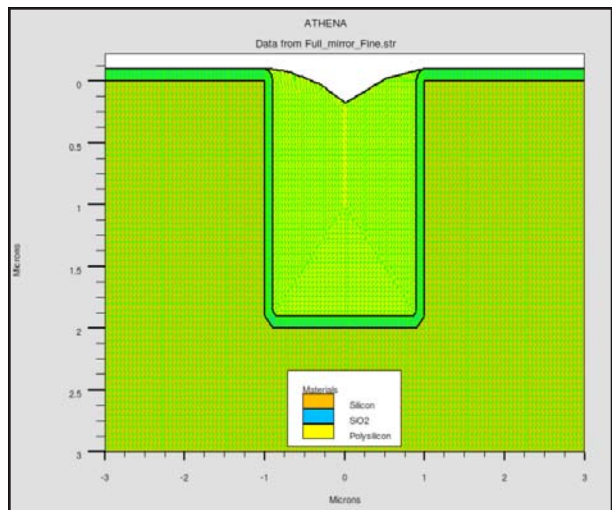

Figure 6a. Starting structure with dense mesh prior to relaxation.

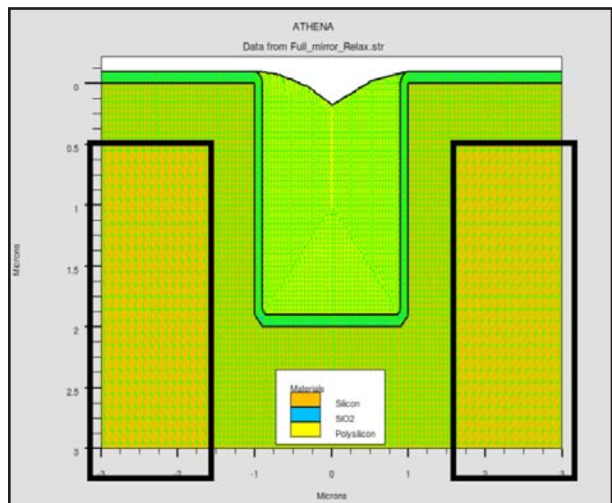

Figure 6b. Box mesh relaxation on the structure shown in Fig 6a using ATHENA syntax.

Mask Alignment to Mesh Lines

ATHENA hosts a set of highly advanced meshing algorithms. Even though these algorithms are very robust the user should try and adhere to certain 'best practices'. Although not mandatory, one such best practice is to explicitly define mesh lines on the edge of masked or etched regions. Otherwise inaccuracies can occur and obtuse triangles can be introduced which can potentially cause convergence issues in both ATHENA and ATLAS.

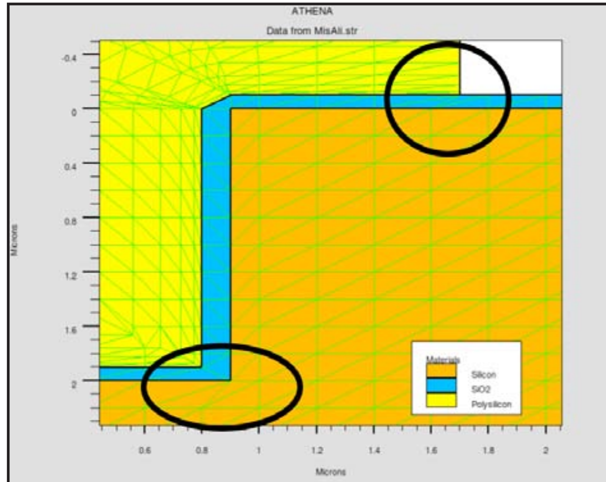

Figure 7a. The introduction of obtuse triangles due to non-alignment of mesh lines and masked edges.

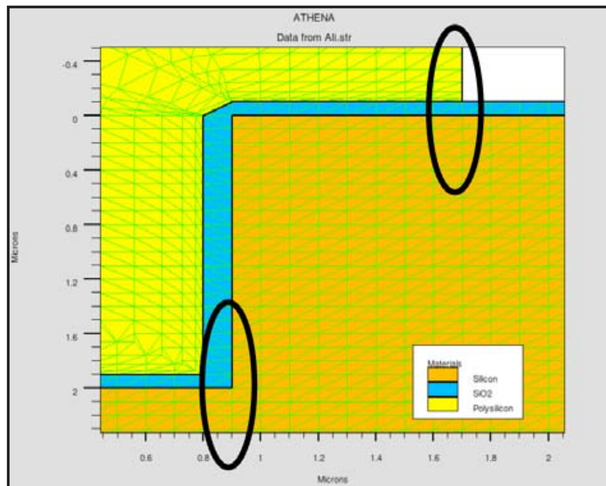

Figure 7b. Obtuse triangles can be minimised if mesh lines are explicitly defined at the edge of masked regions.

Shown in Figure 7a is a structure where mesh lines have not been aligned to the masked edges. The mesh was set at a uniform spacing of 0.2um, the trench was etched at X=0.9 therefore the vertical edge of the trench lies in between the X mesh line at 0.8 and 1.0. Similarly with the polysilicon at the surface, the material was etched to the right of X=1.7. Consequently the vertical edge of the poly silicon at the surface lies in between the X mesh lines found at X=1.6 and 1.8. Obtuse triangles can clearly be observed in the oxide at the edge of the etched poly.

Although not severe in this instance such situations are best avoided. These issues can easily be overcome by ensuring mesh lines are aligned to the edge of the masked region as shown in Figure 7b.

Global Mesh Refinement

The most common cause of convergence issues is almost certainly poor meshing. If you are facing convergence issues and you suspect that your mesh is not well defined you can carry out a rapid assessment of this by applying a global mesh density multiplication factor to your mesh. This feature is available in both ATLAS and ATHENA. The base mesh density will be globally increased by the user-defined multiplication factor. This factor can also be used to reduce mesh density.

In ATHENA, the multiplication factor is used on the INIT statement. For example:

```
INIT ORIENTATION=100 C.PHOS=1E13 SPACE.
MULT=0.5
```

In ATLAS the global multiplication factor flag is set on the MESH statement as in:

```
MESH SPACE.MULT=0.5
```

If the multiplication factor is set to less than one then the mesh density will be increased by that factor. If it is set to a value greater than one then the density will be decreased.

Figures 8a to 8c illustrate the effect on the mesh in ATHENA. Figure 8a shows SPACE.MULT=0.5, Figure 8b shows SPACE.MULT=1 and Figure 8c shows SPACE.MULT=2.

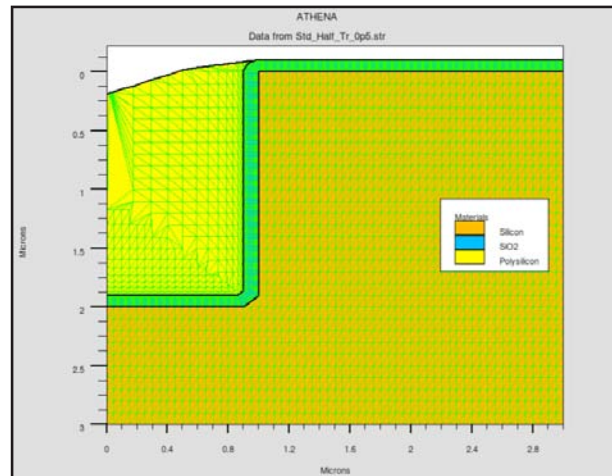

Figure 8a. SPACE.MULT=0.5. Starting mesh density doubled.

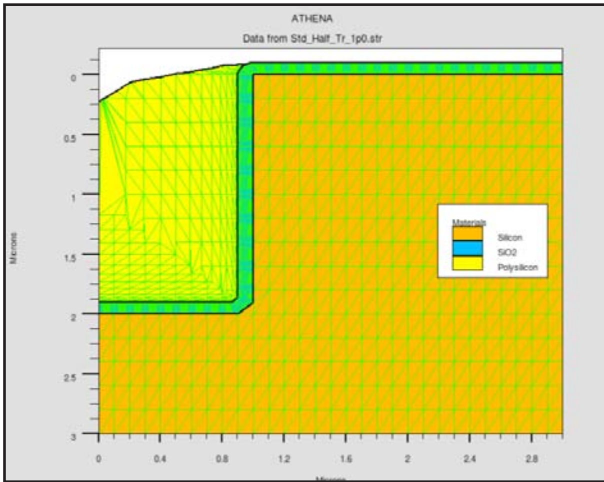

Figure 8b. SPACE.MULT=1. The 'standard' mesh density considered with no multiplication.

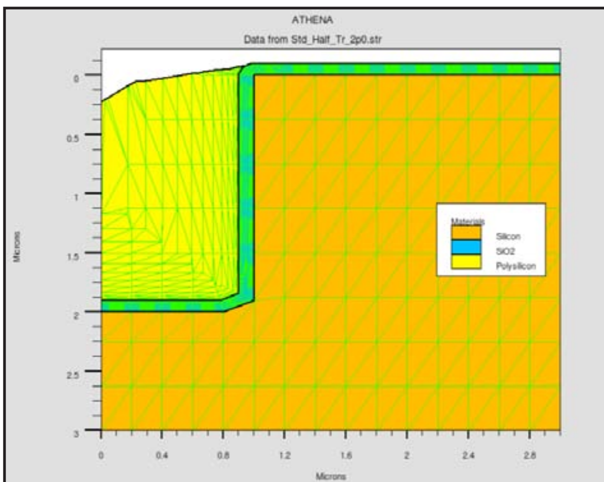

Figure 8c. SPACE.MULT=2. Starting mesh density halved.

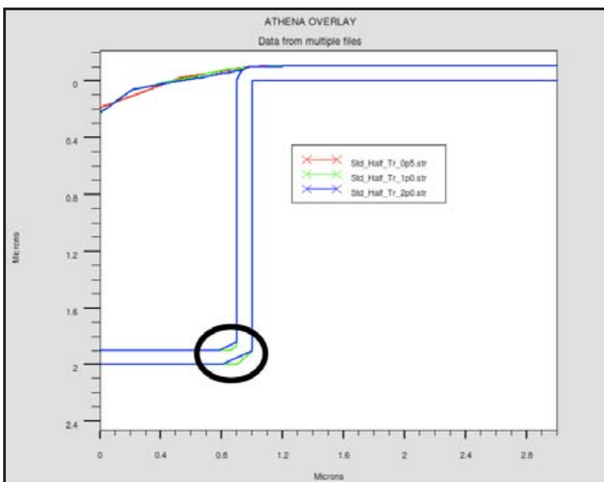

Figure 8d. Overlay of Fig 8a – 8c the coarse mesh of Fig 8c (blue) has lead to an inaccuracy on the internal corner of the trench.

The three structures shown in Figures 8a – 8c can then be overlaid to study for any effects due to the changing mesh density. The overlay is shown in Figure 8d. The region edges of the two structures with the finest meshes lie perfectly on one another (SPACE.MULT=0.5 and 1, denoted by the green and red lines in Figure 8d). However, reducing the mesh density (SPACE.MULT=2, Figure 8c) causes a slight inaccuracy on the internal corner of the trench. As highlighted, the region edge line does not lie in the same location as it does in the other two structures that have a higher mesh density.

Remeshing In DevEdit

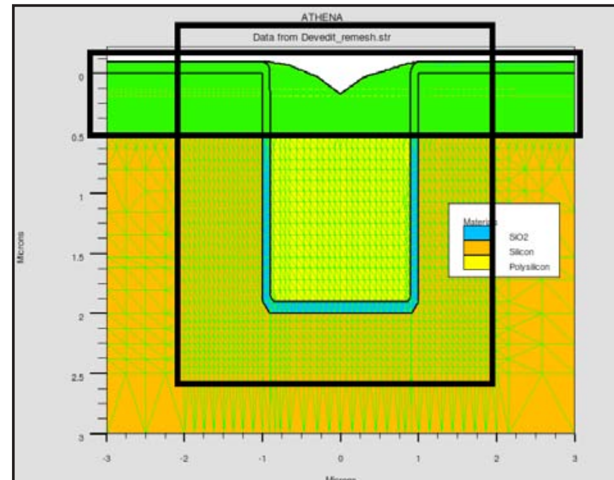

Figure 9. Box refinement in DevEdit.

A great feature about the Silvaco suite of TCAD tools is the auto interface. This means that a structure can be generated in one TCAD tool then passed seamlessly in the same deck into another TCAD tool. Just because a mesh is good for process simulation it doesn't mean that it will be good for electrical simulation in ATLAS. Therefore DevEdit can be used to re-mesh before electrical simulation. DevEdit can also be used part way through a process run to re-mesh or as a standalone tool to create structures from scratch.

DevEdit offers many ways to define your new mesh making it a great universal tool. For example, you can re-mesh on specific regions, materials, under regions, on volume data or in defined boxes.

Shown in Figure 9 is the re-meshed full cell structure. In this instance we have inserted DevEdit syntax part way through the ATHENA deck as shown in Figure 10. The syntax creates a moderately coarse base mesh then defines two mesh refinement boxes, as highlighted. The first is a relatively fine mesh covering the surface. The second is a coarser mesh covering the trench poly.

```
Deckbuild V3.40.8.R - Full_Tr_devedit.in, dir: /home/davidg/
File  View  Edit  Find  Main Control  Commands  Tools

go athena

#TITLE: Full Trench With Devedit Meshing

line x loc=0.00 spac=0.1
line x loc=3 spac=0.1

line y loc=0.00 spac=.2
line y loc=3.00 spac=.2

init orientation=100
#
etch silicon  start x=0  y=-1
etch  cont  x=1 y=-1
etch  cont  x=1 y=1.9
etch  cont  x=0.9 y=2
etch  done  x=0 y=2
#
structure mirror left

dep oxide thick=0.1

dep poly thick=2

go  devedit
# Define the base Mesh
base.mesh  height=1 width=1
bound.cond  apply=false max.ratio=300

# Define the first fine mesh over the surface. Max and min values are set equal
# to give a uniform mesh
constr.mesh id=1 x1=-3 y1=-0.5 x2=3 y2=0.5 default max.height=0.01 max.width=0.1 \
        min.height=0.01 min.width=0.1

# Define second mesh refinement box covering the trench region
constr.mesh id=2 x1=-2 y1=0 x2=2 y2=2.5 default max.height=0.1 max.width=0.1 \
        min.height=0.1 min.width=0.1

# Build the mesh
mesh

# Jump back to athena to etch back the poly
go athena

etch poly thick=2
struc outf=Devedit_remesh.str

quit
ATHENA
```

Figure 10. The use of DevEdit syntax part way through an ATHENA process run to remesh using boxes of refinement.

Mesh Refinement on Volume Data in DevEdit

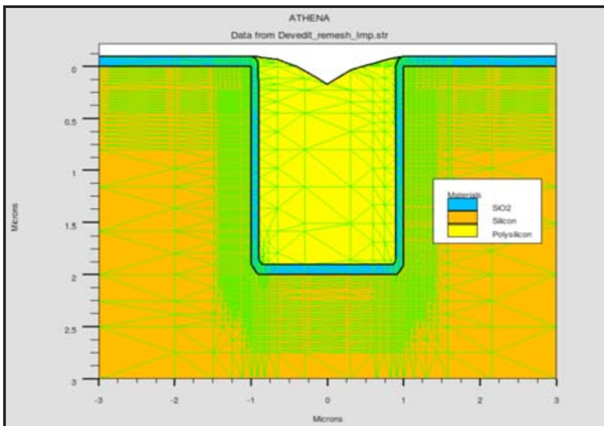

Figure 11a. Remeshed, refinement on quantities.

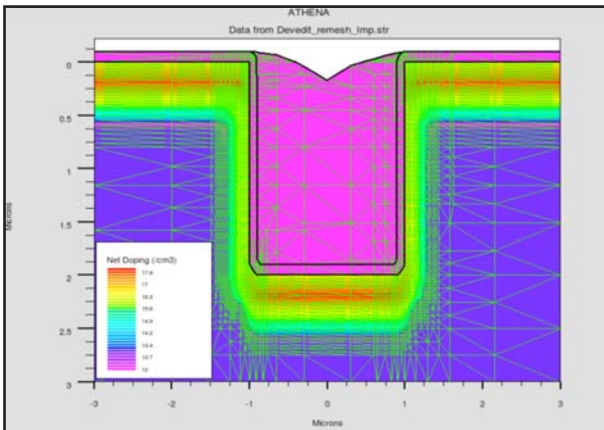

Figure 11b. Remeshed, displaying the mesh and volume data (Net Doping) that has been refined on.

```
Deckbuild V3.40.8.R - BaseMesh_RefImp.in, dir: /home/davidg/
File  View  Edit  Find  Main Control  Commands  Tools

go athena

#TITLE: Full Trench With Devedit remeshing on impurities

line x loc=0.00 spac=0.1
line x loc=3 spac=0.1
#
line y loc=0.00 spac=.2
line y loc=3.00 spac=.2
#
init orientation=100 c.phos=1e13
#
etch silicon  start x=0  y=-1
etch  cont  x=1 y=-1
etch  cont  x=1 y=1.9
etch  cont  x=0.9 y=2
etch  done  x=0 y=2
#
implant boron dose=1E13 energy=50

structure mirror left

dep oxide thick=0.1

dep poly thick=2

# Auto Interface into Devedit
go  devedit

# Define base mesh
base.mesh  height=1 width=1
bound.cond  apply=false max.ratio=300

# Define volume data to refine on - Net Doping
# and set remeshing control parameters
imp.refine imp="Net Doping" scale=log sensitivity=0.1 transition=1
imp.refine min.spacing=0.05

# Build mesh
mesh

# Auto interface, move seamlessly back to Athena
# to finish off process run
go athena

etch poly thick=2
struc outf=Devedit_Remesh_Imp.str

quit
```

Figure 11c. Deck and syntax used to auto interface between ATHENA and DevEdit showing the use of DevEdit to refine on volume data.

One feature that is commonly used in DevEdit is re-meshing on volume data, such as net doping, a specific impurity, electric field or potential. For accurate results and to ensure convergence in ATLAS it is critical that the volume data is well meshed. In areas where the volume data changes the mesh should be dense enough to accurately reflect these changes. Where there is no change in the volume data, such as deep in a uniformly doped bulk / substrate region, the mesh density can be quite coarse.

Shown in Figure 11a is the full cell structure we have previously considered. The mesh shown was created in Devedit. Refinement has been undertaken on Net Doping. Consequently, where the volume data changes the mesh is quite fine, and where the doping is uniform (bottom left and right) the mesh is quite coarse. Shown in Figure 11b are the net doping contours with mesh overlay highlighting the refinement on doping.

Structure Generation in ATLAS

The user should always consider very carefully if itis necessary to create a full process run. Typically it is far quicker both computationally and in user development time to generate the structure exclusively in ATLAS or using the DevEdit GUI. A full range of tools are available in ATLAS to ensure that the user can accurately model their device and doping profiles. Further realism can be achieved in ATLAS if the user, rather than defining their doping profiles in the deck, imports experimental (SIMS, SRP etc) doping profiles into the ATLAS simulation.

The half-cell ATHENA structure has been reproduced in ATLAS. The ATLAS deck is shown in Figure 12a and the resulting structure in Figure 12b. The ATHENA structure is shown in Figure 12c for ease of comparison. From this it is clear that fundamental replication is quite an elementary task. To turn the ATLAS structure into a pseudo-process structure, the doping profiles from the ATHENA structure have been extracted (although we are importing simulated profiles, as noted, the profiles could be sourced from experiments) and then imported into the ATLAS simulation. Figure 12d shows the doping in the ATLAS structure replicating, as intended, the doping in the ATHENA structure which can be seen in Figure 12e.

Deckbuild V3.40.8.R - Atlas.in, dir; /home/davidg/

File | View | Edit | Find | Main Control | Commands | Tools

```
go atlas

mesh

# Define uniform mesh in X-direction
x.mesh loc=0.0   spac=0.1
x.mesh loc=3.0  spac=0.1

# Define uniform mesh in Y-Direction
y.mesh loc=-0.1   spac=0.1
y.mesh loc=3.0 spac=0.1

# Define material regions
region  num=1 silicon
region  num=2 oxide x.max=1 y.max=2
region  num=2 oxide y.max=0
region  num=3 polysilicon x.max=0.9 y.max=1.9

# Call in experimental doping profile
doping  2d.ascii infile=doping.dat p.type

save outfile=Atlas_Comp.str
quit
```

Plotting... ATLAS

Figure 12a. ATLAS deck used to create equivalent structure.

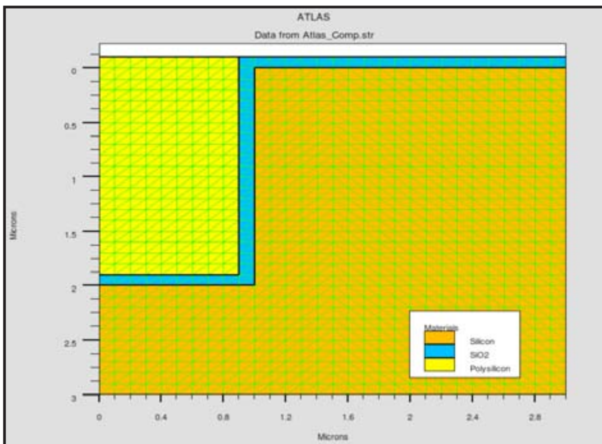

Figure 12b. Equivalent structure created in ATLAS.

Conclusions

Being able to rapidly generate efficient and robust meshes is a one of the biggest challenges to a TCAD user. In this article a selection of techniques to aide such endeavours has been presented.

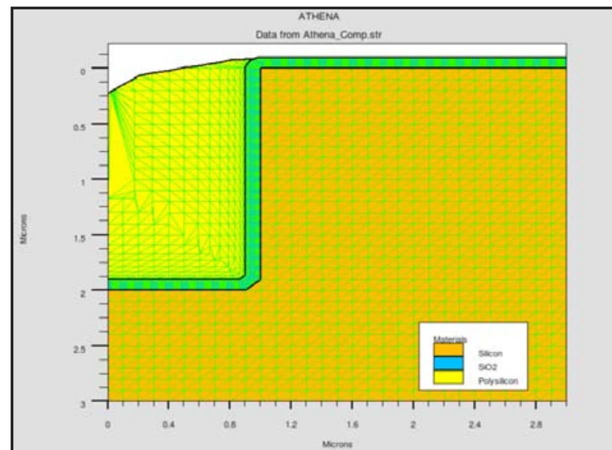

Figure 12c. ATHENA equivalent for comparison.

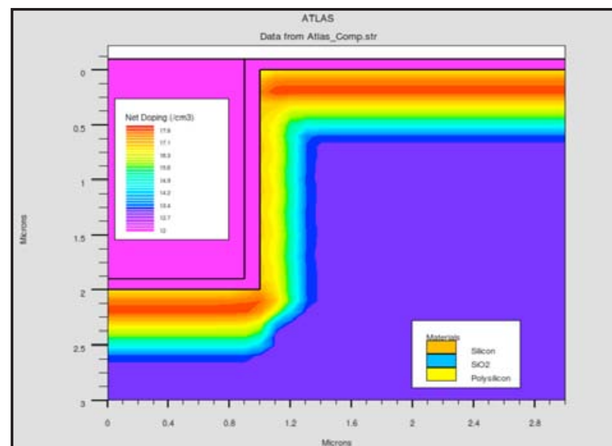

Figure 12d. Net Doping in the ATLAS structure. A more realistic pseudo-process structure is achieved as the doping is imported in from an experimental source.

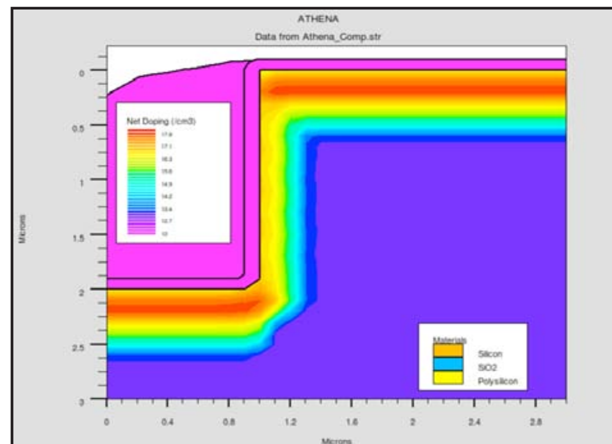

Figure 12e. Original experimental ATHENA doping that was extracted and mapped onto the ATLAS structure.

References

[1] "DevEdit: A Flexible Tool for Structure Editing and Mesh Generation" *Simulation Standard*, Volume 8, Number 8, August 1997. Available from: http://www.silvaco.com/

[2] "Mesh Control in ATHENA", *Simulation Standard*, Volume 9, Number 2, February 1998. Available from: http://www.silvaco.com/

[3] "Hints, Tips and Solutions", *Simulation Standard*, Volume 13, Number 11, November 2003. Available from: http://www.silvaco.com/

Hints, Tips and Solutions

Q. How do I convert a value from one set of units into another?

A. Use the DBINTERNAL CONVERT.UNITS command.

1) Introduction

The command CONVERT.UNITS was introduced in DBINTERNAL version 2.5.2.C. This allows the conversion of a value in one set of units into a value in another set of units that have the same dimension.

For example it can convert values from one set of units into another:

```
convert.units 0.1 in/s into um/min
0.1 in / s == 152400 um / min
```

Alternatively it can check that two different representations of a unit are the same thing (such as the viscosity coefficients in ATHENA and VICTORY Process)

```
convert.units 1 g/(cm.s) into (dyn.s)/cm2
1 g / cm.s == 1 dyn.s / cm2
```

2) Units and SI prefixes.

Units are usually indicated by their standard unit symbol: "m" for meter, "s" for second, "dyn" for dyne, etc. This symbol is case sensitive. The symbol for a liter is a lower case "l" (an upper case "L" is the symbol for the lambert). The symbol for an electron volt is "eV" (none of "ev", "Ev", or "EV" would be recognized as an electron volt).

Several standard symbols cannot be represented in the Latin alphabet and so have equivalent names: "Ang" for Å, "Ohm" for Ω, "degC" for °C, and "degF" for °F.

If the same unit means different things then the specific meaning of a unit must be indicated. For example "atu" is the standard symbol for atomic unit, but there are many atomic units. The specific atomic unit required must be specified by adding the type, in brackets, after the unit name.

```
convert 1 atu (time) into s
1 atu (time) == 2.41888e-17 s
convert 1 atu (energy) into eV
1 atu (energy) == 27.2114 eV
```

This also occurs with several units that are shared between the US and the Imperial measurement systems, see section 5.

Any of the SI prefixes can be used with any unit. There must be no space between the SI prefix and the unit it is associated with. The letter "u" is used instead of μ to indicate 10^{-6}. Tom Duff of Bell Labs observed that a nanocentury is about π seconds.

```
convert.units 1 ncentury into s
1 ncentury == 3.1557 s
```

If there is a clash between a prefixed unit and a bare unit then the parser will recognize the bare unit. The symbol for inch is "in", therefore "min" could be "milli-inch", but the unit "min" will be parsed as "minute".

Any number that comes immediately after a unit is assumed to be a power that the unit is raised to. This number can be a positive or negative integer, fraction, or decimal. Any space between the unit and the number is ignored. The numbers may, but don't have to be, enclosed in brackets. The "^" symbol can be used to make the expression more readable if desired. The following units all have the dimension of a volume ("b" is "barn", an SI unit of area); "m^3", "m 3", "m3", "b^3/2", "b(3/2)", "b 3/2", "b1.5".

Any two adjacent units are assumed to be multiplied. Units are adjacent if they are separated by spaces, by the "." sign, or by an exponent. Dynamic viscosity is "Pa.s" or "Pa s". Energy is "m2kg s-2" (where the exponent of "m" separates "m2" and "kg").

The solidus "/" (when not being used in an exponent fraction) switches from the numerator to the denominator (or back again) where any subsequent units are added. The energy units could have been written "m2 kg / s2". Spaces around the solidus are not needed but can be used to make the units easier to read.

A repeated solidus is allowed, but it probably doesn't do what is expected. An old style of writing the acceleration was "ft/s/s". In this parser the "ft" is added to the numerator, the first solidus indicates subsequent units should be added to the denominator, therefore the first "s" is added to the denominator, the second solidus indicates subsequent units should be added to the numerator, therefore the second "s" is added to the numerator. The two "s" will cancel leaving a length rather than an acceleration.

```
convert.units 1 ft/s/s into m
1 ft == 0.3048 m
```

The units of acceleration should be defined as "ft/s2", or "ft.s-2", or "(ft/s)/s".

3) The "D" prefix

Most of the units are "ratio scale" units which means that the origin is an absolute zero and a single unit has the same magnitude everywhere on the scale.

However the common temperature units, degrees Celsius and degrees Fahrenheit, are "interval scale" units, which means that a single unit has the same magnitude everywhere on the scale but the origin is not an absolute zero.

There is a potential for confusion when converting values between "interval scale" units and "ratio scale" units. For example, suppose a beaker of water was at 41 °F and it was warmed up by 45 °F to 86 °F. If these temperatures were converted into Kelvin then

```
convert.units 41 degF into K
41 degF == 278.15 K
convert.units 45 degF into K
45 degF == 280.372 K
convert.units 86 degF into K
86 degF == 303.15 K
```

Obviously 278.15 K + 280.372 K is not equal to 303.15 K.

The problem is that the difference (45 °F) should be measured on a "ratio scale" rather than the normal Fahrenheit "interval scale". The unit needs to be prefixed with a "D" to indicate this (there must be no space between the D and its associated unit).

```
convert.units 45 DdegF into K
45 degF == 25 K
```

And 278.15 K + 25 K = 303.15 K.

If the "D" is used with an SI prefix on the unit then the "D" must come before the SI prefix.

```
convert.units 9 DmdegF into uK
9 mdegF == 5000 uK
```

To convert into a difference unit the "D" prefix must occur on the first unit

```
convert.units 5 K into DdegF
5 K == -450.67 degF
convert.units 5 DK into degF
5 K == 9 degF
```

A "degF" or "degC" in a compound unit is treated as a "ratio scale" unit, rather than an "interval scale" unit.

```
convert.units 5 J/degF into J/K
5 J / degF == 9 J / K
convert.units 9 degF/W into K/W
9 degF / W == 5 K / W
```

4) Incompatible units.

If an attempt is made to convert one unit into an incompatible set of units then the conversion fails and a warning is issued.

```
convert.units 1 W into J
The conversion failed
The dimensions of "W"
   [LENGTH^2.MASS.TIME^-3]
Are incompatible with "J"
     [LENGTH^2.MASS.TIME^-2]
[LENGTH^2.MASS.TIME^-3] ==
   [LENGTH^2.MASS.TIME^-2] * [TIME^1]
```

The term at the far right of the last line, after the "*", indicates what dimensions need to be added to the required units to make them dimensionally compatible with the initial units.

```
convert.units 1 W into J.s-1
1 W == 1 J / s
```

This can be used to check the dimensions of a set of units: by attempting to convert one unit into a dimensionless unit (indicated by a unit of "1").

```
convert.units 1 W into 1
The conversion failed
The dimensions of "W"
    [LENGTH^2.MASS.TIME^-3]
Are incompatible with "1" []
[LENGTH^2.MASS.TIME^-3] ==
    [] * [LENGTH2.MASS.TIME^-3]
```

Hence the dimensions of "W" are "$L^2.M.T^{-3}$".

5) US and Imperial units.

Many of the traditional volume units, and various other units, have different magnitudes in the US and Imperial systems. For example:

```
convert.units 1 gal (us) into gal (imp)
1 gal (us) == 0.832674 gal (imp)
```

Initially all such unit names must be qualified by "(us)" or "(imp)". If it is known that one type of system is predominant then the `CONVERT.UNITS.OPTIONS` command can be used to indicate which system an unqualified unit should belong to.

```
convert.units.options us
convert.units 1 gal into gal (imp)
1 gal (us) == 0.832674 gal (imp)

convert.units.options imp
convert.units 1 gal into gal (us)
1 gal (imp) == 1.20095 gal (us)
```


Call for Questions

If you have hints, tips, solutions or questions to contribute, please contact our Applications and Support Department
Phone: (408) 567-1000 Fax: (408) 496-6080
e-mail: support@silvaco.com

Hints, Tips and Solutions Archive

Check out our Web Page to see more details of this example plus an archive of previous Hints, Tips, and Solutions
www.silvaco.com

USA Headquarters:

Silvaco, Inc.
4701 Patrick Henry Drive, Bldg. 2
Santa Clara, CA 95054 USA

Phone: 408-567-1000
Fax: 408-496-6080

sales@silvaco.com
www.silvaco.com

Worldwide Offices:

Silvaco Japan
jpsales@silvaco.com

Silvaco Korea
krsales@silvaco.com

Silvaco Taiwan
twsales@silvaco.com

Silvaco Singapore
sgsales@silvaco.com

Silvaco Europe
eusales@silvaco.com

SILVACO