Lecture 18: Modeling with Athena and Atlas Part II

Last Time
- We loaded up an example program in Deckbuild
Lecture 18

- Silvaco Code.
- Defining the Mesh.
- Defining the Substrate.
- Oxidation.
- Etching.
- Doping.

Silvaco Code
Silvaco code.

- Last lecture we saw that code is entered into the DeckBuild GUI:

- Athena and Atlas use the same syntax.
- In fact, both the Athena and Atlas parts of code are in the same file.

Syntax

- The code runs sequentially, from top to bottom.
- The line numbers are not show by default, but you can turn them on from the main menu bar: “View” → “Show Line Numbers”.
- We will run code from a single file, with no functions, classes etc.
- Comments start the line with “#” and will be skipped over by the interpreter.
- There is no explicit line termination (c.f. “;” in c/c++ etc.). The new line character indicates a new line.
DeckBuild Commands

- Luckily you do not have to actually write too much code yourself.
- DeckBuild has a few built-in functions and dialogs which will help you to generate relevant code.
  - We will use these later.
- All they do is automatically create code.
- You could do the same thing by typing the code.

Interpreter

- DeckBuild uses an interpreter rather than a compiler.
- This means that rather than converting the entire program to machine code (1's and 0's) then executing it, DeckBuild will convert each line to instructions, line-by-line.
- You can run code line by line by hitting the “run to line button” then hit F10 to run the code one line at a time.
- There are various other buttons such as Kill, which will stop the program running.
Output

- The bottom part of DeckBuild shows the output.
- This will provide any important information that occurs while the code is running.
- If DeckBuild encounters a Syntax Error, an error box will be shown:

![Error Box](image)

Empty Program

- We will start by running an empty project.
- Write the following into DeckBuild:

```plaintext
go athena
quit
```

- The output will look something like the following:

![Output](image)
Defining the Mesh

Discretization

• We will to a certain-extent treat Silvaco like a “Black Box”.
• I.e. we specify a set of parameters and do not worry about the exact models used to get the results.
• But we do need to know a couple of things about the calculations however.
• The first is how space and time are discretized.
• A lot of methods (Monte Carlo, Finite Difference) rely on approximating a continuous function as a discreet one.
• E.g.:

\[
\int_a^b f(x) \, dx = \lim_{n \to \infty} \sum_{i=1}^{n} f(x_i)\Delta x
\]

\[
\Delta x = \frac{b - a}{n}
\]

\[
x_i = a + i\Delta x
\]

\[i \text{ is an integer (not the imaginary unit)}\]

https://en.wikipedia.org/wiki/Riemann_sum

Discretization

• So to employ these methods we must divide up continuous space and time into finite sections (e.g. $\Delta x$ and $\Delta t$).
• As long as these sections are small enough, the calculations will be accurate.
• We will first define a 2-dimensional spatial grid (in $x$ and $y$).
• Each position on the grid we define as a node.
• We must make the grid fine enough that computation is accurate, but not too fine as the calculation takes too long.
• For most Athena operations we can approximate the processing time to scale with $(N_p)^\alpha$, where $N_p$ are the total number of nodes and $\alpha$ is normally between 1.5 and 2.0.

Define the Mesh

• To define the mesh, first place your cursor where you want the code to appear:
• Then click “Commands” → “Mesh Define”
• The dialog should appear something like this:
Define the Mesh

- The first thing to be aware of is that all dimensions are microns (despite appearing unlabeled).
- We will start by creating a simple $1\,\mu m \times 1\,\mu m$ simulation area with grid spacing's of $0.1\mu m$.
- First, click the ViewGrid checkbox so the grid is shown:

In the "X coordinates" group box enter:
- Location = 0. Spacing = 0.10. Click insert.
- Location = 1 Spacing = 0.10. Click insert.
- Do the same in the "Y coordinates" group box.
- You should see something like the following:
Define the Mesh

- If you click the “write” button it will add the following code where your cursor was:

```
line x loc=0 span=0.10
line x loc=1 span=0.10
line y loc=0 span=0.1
line y loc=1 span=0.1
quit
```

- You could have equivalently written the above directly into DeckBuild.
- If you run this code the software will generate the mesh then quit.

Non-Uniform Mesh

- We can add more detail to certain parts of the mesh than others if we want to.
- E.g. we may want a finer mesh close to where we expect the channel to be.
- Let’s delete the code we just produced, then come back to the Mesh Define Dialog box. Enter the following:
Non-Uniform Mesh

• If you define different spacings at different positions, the software will ensure the spacing is correct at the position you specify, and interpolate the space between the other points.

• The code will come out as expected:

Non-Uniform Mesh

• You can make more very complicated meshes by adding more definitions of spacing:

• The number of points is given below the preview of the mesh.
Our Mesh

- For our example we want a large distance on the surface of the wafer, and fine resolution in the \( y \)-direction.
- These are outside the bounds allowed by the dialog box, so we just enter the code directly:
- After entering a section of code, it is good practice to run the program and make sure you have not made any errors.
- We can’t plot what we have yet.

```plaintext
#define the Mesh
line x loc=0 spac=10
line x loc=100 spac=1
line y loc=200 spac=1
line y loc=200 spac=5
line y loc=0 spac=0.025
line y loc=1.5 spac=0.0625
line y loc=6.75 spac=0.0625
line y loc=10 spac=1.875

quit
```

Defining the Substrate
Substrate

- The next step in our simulation is to define the wafer.
- In particular we are going to define:
  - The crystal orientation.
  - The dopant type.
  - The resistivity / dopant concentration.
- This is the properties of the wafer as we receive it (i.e. before we do anything to it).
- We can either specify the doping concentration as provided by the manufacturers, or we can define the resistivity (as measured) and the software will work out the concentration.

To define the substrate properties go to “Commands” → “Mesh Initialize”.

- This should bring up the dialog box shown to the left.
- There are a lot of options, we will just enter the parameters as shown.
- See p33 of the Athena User Manual for more details.
Substrate

- If you hit “write” it should produce something like the following:
- As usual, it is good practice at this stage to run the code and make sure you have not made any typos.
- We have now provided Athena with enough information to plot a graph.

Plotting Doping Concentration

- Add the following line to output the data to a structure file:
  ```apl
  structure outf=DopingConc.str
  ```
- DeckBuild show now look something like the following:
Plotting Doping Concentration

- If we go out of DeckBuild and back to the UNIX file system and use the `ls -l` command we should see that we have now created this file (among several other files):

```
Dopingconc.str
```

- Go back into DeckBuild and add the following line of code below where we define the structure:

```plaintext
tonyplot DopingConc.str
```

- This will call TonyPlot to plot the DopingConc structure.

---

Plotting Doping Concentration

- The code should now look like this:

```plaintext
# Define the mesh
line x loc=0 spacing=40
line x loc=10 spacing=2
line x loc=200 spacing=2
line y loc=0 spacing=0.025
line y loc=0.5 spacing=0.025
line y loc=0.75 spacing=0.025
line y loc=1 spacing=0.025

# Define substrate resistivity is in Ohm cm
init silicon boron resistivity=1e14 orientation=111
structure unit=plottingConc.str
tonyplot input=plottingConc.str
quit
```

- A few seconds later the plot should be displayed.
Plotting Doping Concentration

- The data should look like this:
- We have doping concentration (cm\(^{-3}\)) on the y-axis and depth (in \(\mu m\)) on the x-axis.
- The wafer is described as uniformly doped.
- Since we only have one dopant type, the boron concentration is here the same as the net doping concentration.
- This plot can then be exported as described in the last lecture.

Oxidation
Thermally Grown Oxides

- We are roughly following the process you followed in the laboratory.
- The next step is to grow an oxide.
- There are two ways you can grown an oxide in Athena:
  - Diffusion.
  - Deposition.
- We will carry it out using diffusion first. We talk about deposition later.
- Place the cursor where we want then click “Commands” → “Process” → “Diffuse”.

Thermally Grown Oxides

- The dialog has a few options. We will just cover a few basics.
  - Consult the manual for more details.
- The first tab looks like this:
- Hopefully all these parameters are self-explanatory.
- We will just carry out a simple anneal at 1100°C for 60 minutes.
Thermally Grown Oxides

- We will use a dry $O_2$ atmosphere at 1atm, with no impurities.

- You can also select specific models here.
- It is best to just leave the values as their defaults unless you have a reason not to.
- The best way to understand the effect of different parameters is to run the simulation and compare results.
Thermally Grown Oxides

• With the settings we described, we output this line of code:

```python
diffus time=60 minutes temp=1100 dryo2 press=1
```

• The TonyPlot output will then look something like this:

![TonyPlot output image]

Thermally Grown Oxides

• So far we should have the following code:

```plaintext
# Define the Mesh
line x loc=0 spac=50
line x loc=100 spac=3
line x loc=200 spac=3
line y loc=50 spac=5
line y loc=0 spac=0.025
line y loc=2.5 spac=0.025
line y loc=4.75 spac=0.025
line y loc=10 spac=1.875

# Define substrate (resistivity is in Ohm cm)
init silicon boron resistivity=10 orientation=111

# Thermally grow the oxide
diffus time=60 minutes temp=1100 dryo2 press=1

# Move and plot the data
structure out=DopingConc.str
tonymplot DopingConc.str
quit
```


**Etching**

---

**Simple Etch-All**

- There are a few different ways to etch a film off the surface of a wafer. Here we will just use the simplest approach.
- We will just remove all of the specified material (oxide) in the positions we define.
- Make sure you cursor is where you want the code to appear.
- On the main menu click “Commands” → “Process” → “Etch” → “Etch...”
Simple Etch-All

- This brings up the simple Process Etch dialog box.
- We will define use the “Geometrical” etch method, just allowing us to define where we want the etch to occur.
- If you just leave the “Geometrical Type” as “All” then click the “Write” button the following code will be generated:

  ```
  etch oxide all
  ```

- I.e. all the oxide is removed from the surface of the wafer.

Simple Etch-All

- If you run the program now, the following plot will be created:

  - I.e. we have removed all the oxide, so we are back to our doping concentration diagram.
  - Notice that thanks to oxidation, the doping is no longer uniform with depth.
Etch Specified Regions

- We can instead remove just some of the substrate.
- In this example, we just remove the oxide to the left of $x = 50 \, \mu m$.
- This would create the following code:
  
  ```
  etch oxide left p1.x=50
  ```

- You can edit the code by hand if the values are outside normal limits.
- We will change the position to $100 \, \mu m$:
  
  ```
  etch oxide left p1.x=100
  ```

If you run the program now, the following plot will be created:

- By default, TonyPlot will output the most logical information given what you provide.
- Here we see that we have etched all the SiO2 to the left of $x = 100 \, \mu m$. 
Etch Specified Regions

- Either by using the dialog or by editing the code directly, let's do the same to the right side of the wafer.
- The code should now look like this.
- The TonyPlot should look like:

Doping
Implantation

- The easiest way to dope a wafer in Athena is via ion implantation (Lecture 8).
- This can be done using a single line of code, or the “Implant” dialog box.
- Make sure you have the cursor is where you want the code to appear.
- On the main menu click “Commands” → “Process” → “Implant”

![Implantation dialog box in Athena]

- This brings up the simple Process Implant dialog box.
- Most settings are self explanatory.
- We will just accelerate the ions perpendicular to the wafer, and ignore damage.
- The accelerating voltage is a way to control the depth of the peak ion distribution.
- With these settings you should get the following line of code:

```python
implant boron dose=1.0e14 energy=100 tilt=0 rotation=0 crystal
```
Implantation

- If you run the code, you should get something that looks like the following:

- I.e. not much has changed.
- To see the doping concentration, you need to change the display settings.

Output

- To see the doping concentration, right click on the graph and click “Display” from the menu.
- This will then bring up the Display menu:

- Click the “Contours” button so it is depressed then click Apply to show doping concentration as a function of position.
- Click “OK” to accept and close the dialog box.
Implantation

- Using the code we have so far you should see something like following.

- This is not what we wanted: the mask has not worked.
- This is because the high-energy ions can penetrate deep, through the oxide, into the silicon.

Implantation

- We can have a more effective mask by making it thicker.
- We can easily achieve this by making the oxidation duration 10× longer.
- The oxide thickness should now be about 0.5 μm.
- Alternatively we could have used a wet oxide or a higher temperature.
Implantation

- Now we get the expected behavior.

- The depth of the peak in the dopant distribution (and its width) will depend on the ion type and the accelerating energy (see Lecture 8).

Diffusion Doping:

- Athena is not really designed to model fluid dopants unfortunately.
- It is more designed for ion-implantation (as discussed previously).
- There is a way it can be done however:
  - We are just going to deposit some SiO₂ with a very high impurity concentration.
  - We will then drive in the dopant and remove the SiO₂ afterwards.
  - It is not a perfect approach, but good enough for what we want to achieve.
Diffusion Doping:

- As usual, make sure the cursor is in the correct position first.
- To open the dialog box click “Commands” → “Process” → “Deposit” → “Deposit”.

We will just deposit 0.5 μm of SiO₂ (just labelled “Oxide”).
- We will set the grid layers (divisions) to 10.
- We will set the impurity concentration to 10²¹ cm⁻³.
Diffusion Doping:

- We will also adjust our code so we can see the output before and after the deposition.

- Now, the program should output two separate TonyPlot windows by the time it completes.

Output

- You should see two windows appear that will look something like the following:

- So by default TonyPlot will just give us a two-dimensional map of which material is located where.
Output

- As before we need to change the settings to display doping concentration:

  Before
  ![Before](image1)
  ![After](image2)

- So you will notice that all of the dopant resides in the new oxide layer we have deposited on top of our old one.

Drive-In

- To drive in our dopants we anneal the wafer again.
- As before, place the cursor where we want then click "Commands" → "Process" → "Diffuse".
- Here we are going to anneal for 30 minutes, at 1100°C, but this time we will anneal in N₂ (not O₂, H₂O as for SiO₂ growth).
Output

• This time we will look at the dopant distribution before and after drive in. The code should look like:

```plaintext
Before

1. Define the mesh.
2. Specify the dopant distribution.
3. Create the contours.
4. Plot the results.

After

1. Define the mesh.
2. Specify the dopant distribution.
3. Create the contours.
4. Plot the results.
```

• After turning on “Contours” in TonyPlot, as we described before, you should see two windows appear that will look something like the following:

**Before**

**After**
**Etching**

- The oxide we had previously on the substrate was used as a diffusion mask.
- Since we have now deposited the dopant and carried out the drive-in diffusion process, we can now remove it.
- You could either use the dialog as before, or simply write the following line of code:

  ```
  etch oxide all
  ```

- This just removes the oxide from all positions on the surface of the wafer.

**Output**

- After removing the oxide the dopant concentration and code should now look something like the following:
Summary

- We have looked at the basics of the Athena language.

Next Time

- We will calculate current-voltage behavior using Atlas.