INSTITUTE FOR MICROELECTRONICS

360.246 Simulation of Semiconductor Device Fabrication Assignment 2 - Velocity Fields

Submission is due on Tuesday, May 17th 2021, 10am (submit to **tuwel.tuwien.ac.at**)

- Use C++ 11 or Python 3.6+ for all implementations.
- Binaries/Scripts must be callable with the command line parameters specified below.
- Submit your report as a single PDF file. There is no set length required for the report.
- Submit everything (Sources, Binaries, PDF) together as one zip-file to **tuwel.tuwien.ac.at**
- If you have questions regarding the assignment, you can reach us at **simfab-questions@iue.tuwien.ac.at**

General information

- You may use any compiler of your liking, however we recommend additional warning output. When using GCC, at least the following flags should be set: `-std=c++11 -Wall -Wextra -pedantic`
- Make sure you structure your programs in a readable fashion. It will make you more efficient and help us identify possible errors easily, i.e.: save you marks if there are problems.
- This exercise requires you to use the library **ViennaLS**. Install instructions for C++ and Python modules are available at `https://github.com/ViennaTools/ViennaLS`.

1 ViennaLS Basics

In order to get acquainted with the level set library **ViennaLS**, we will first cover some basics. We want to rebuild the snowman example from `https://github.com/ViennaTools/viennals-example`. You can follow the source file `Frosty.cpp` for reference in the following sections.

1.1 A Sphere / Snowball

The journey of a thousand snowmen starts with a single snowball.

- The first thing we need, is space to put a level set into. The simulation domain is represented by `lsDomain`. If nothing is passed to the class upon construction, the grid delta is 1.0 and there are no boundary conditions, because the domain is infinite and will extend as far the surface reaches.
- Now we need to put some surface into the domain. This is done using `lsMakeGeometry`. The Documentation contains all the shapes that can be created with this class. For simplicity, start with a sphere centred at the origin.
- Now use `lsToSurfaceMesh` to extract a surface from the level set. The resulting `lsMesh` object can then be written to a VTK file using the `lsVTKWriter`.
- In order to view the grid points of the level set and their level set values, use `lsToMesh` and compare it with the explicit surface. Notice that ViennaLS uses a sparse level set representation.
- Inspect your first level set sphere using ParaView.

1.2 Melting a Sphere

In order to simulate the melting of our sphere, we need to advect the level set:

- The class `lsAdvect` performs the advection for us. It needs all level sets which are involved in the advection and a class inherited from `lsVelocityField`, which describes the velocity field by which to advect.
- The best way to pass `lsDomains` to `lsAdvect` is using `insertNextLevelSet`:

```
lsAdvect<double, D> advectionKernel();
advectionKernel.insertNextLevelSet(myLsDomain);
```

- Now we just need a velocity field. The simplest velocity field, just returns a constant:

```
class velocityField : public lsVelocityField<double> {
public:
        double getScalarVelocity(
                        const std::array<double, 3> & /*coordinate*/,
                        int /*material*/,
                        const std::array<double, 3> &/*normalVector*/,
                        unsigned long /*pointID*/) final {
                return -1;
        }
};
```

  The above implementation means, that the velocity is -1 at every point on the surface, therefore our snowman is melting in the sun.

- A single `apply` call performs one advection step adhering to the CFL condition. If the physical advection time was set by `setAdvectionTime()`, several steps will be taken until the given time in seconds has passed. In the Example, this results in the surface shown in Fig. 1.

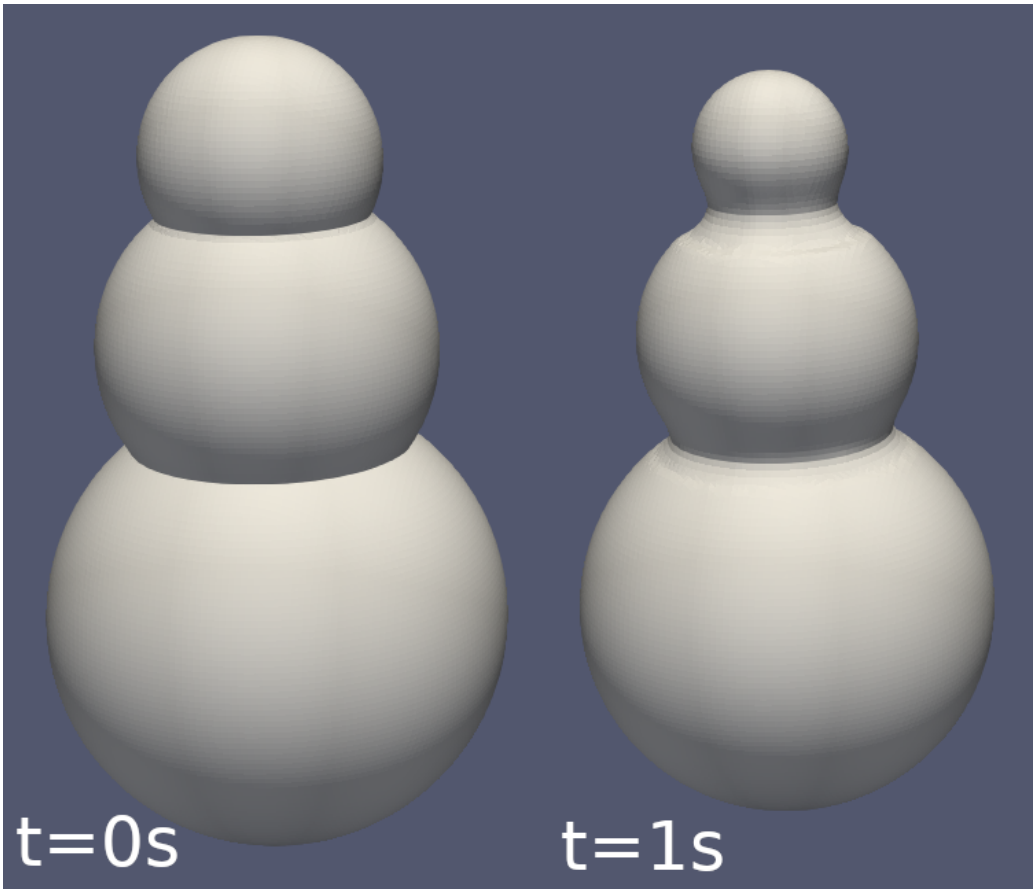

Figure 1: Evolution of the snowman level set by "melting" the snowman for 1 second.

1.3 Multiple Materials

Snowmen consist of multiple spheres, as shown in Frosty.cpp.

- In order to get stable numerics, layer wrapping is used during advection. In order to combine two level sets for a wrapped layer including both, `lsBooleanOperation` is used.
- During advection, only one level set can be moved at a time, the "top" level set. However, the wrapping approach makes it possible to include "lower" materials. If several level sets have the same level set value at the same point in space, the velocity for the "lowest" level set is taken, as shown in Fig. 2. Lower level sets are adjusted to the top level set after etching. During deposition, lower level sets do not change. See the lsAdvect Documentation for details.
- The different velocities can be set in the velocity field. See Frosty.cpp for reference.

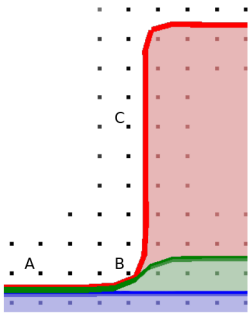

Figure 2: Multiple wrapped materials on the sidewall of a trench geometry. The coloured lines show the location of the explicit surface. Red includes green which includes blue. At the grid point A, the velocity of blue is applied, because it is the lowest layer present at that point. At grid point B it is green, and at C red.

2 Emulating a Real Process

The possibility to set a velocity field for different materials is already quite powerful in creating semiconductor structures. Now we can use one level set as a mask by setting its velocity to something very small (or 0).

2.1 Via Etch

Deep vias are usually fabricated by a series of different etch and deposition steps, known as deep reactive-ion etching (DRIE) or the Bosch Process:

- First, isotropic deposition of a passivating material protects all parts of the substrate.
- Etching of the passivating layer and the substrate takes place. This is almost perfectly directional for the passivating layer because it is mostly removed by high energy ions. For the substrate it is almost isotropic since the substrate is removed mostly chemically.

This process can be emulated, by considering only the velocities on the surface. The result should look similar to Fig. 3.

2.2 Task

- Create a three-dimensional simulation domain with two reflective boundaries and one infinite boundary. You can choose the size and grid spacing as you want, however we recommend a grid extent of $[-40\mu m, 40\mu m]$ in the x- and y-directions and a grid spacing of $1\mu m$.
- Starting from a single plane (substrate/silicon) with normal towards the infinite boundary, deposit a mask material on top.
- Use Boolean operations to generate a circular hole in the mask, but not in the substrate. Since the process will only be active in exposed parts of the substrate, the hole in the mask should span most of the simulation domain. (Hint: Use `lsMakeGeometry` using an `lsCylinder` to make a cylinder.)

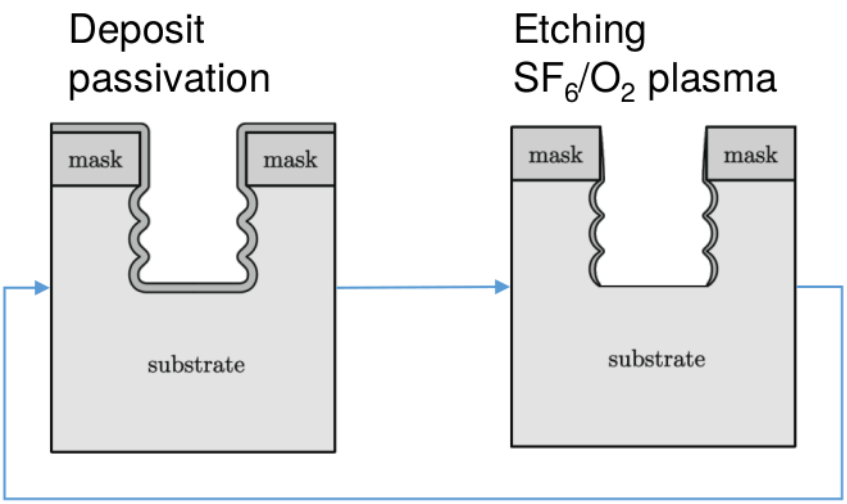

Figure 3: Schematic of a cycle of DRIE.

- Emulate several cycles of the Bosch process to create a deep hole in the substrate. Assume perfectly isotropic deposition and perfectly directional etching for the passivation layer. Assume perfectly isotropic etching of the substrate. Be aware that deposition creates a third material, not silicon or mask, so it will need different velocities than the other two! Describe in your report, which deposition/etch rates and distributions lead to satisfactory results.
- Evaluate how increasing/decreasing the cycle times changes the profile. How does the sidewall roughness (horizontal deviation) change? How deep is the etched hole per cycle in relation to the used rates? What is the benefit/drawback of using longer but fewer cycles?
- Try changing the etch process to be more isotropic instead of perfectly directional. How does the distribution of the etch process (more isotropic versus more directional) influence the final structure?
- What properties of the process influence the size of a single scallop in the sidewall and how?
- Present your results in your report, including figures of the final structures, making sure you include necessary information, such as dimensions and axis labels.

3 Creating Transistor Structures

In the above Task, you have created a single via through a silicon substrate. Although it is a complex process with many steps, it does not look particularly thrilling. Now, we want to create a full transistor using just a chain of simple addition/removal steps. During actual fabrication several complex processing techniques are required, such as sophisticated plasma etching for directional etch steps, which we will try to mimic using simple velocity profiles. Chemical Mechanical Planarization (CMP) is commonly used to flatten the top of structures. CMP should be approximated by simply defining a flat plane and removing material above it using Boolean operations.

3.1 FinFET Transistor

FinFET transistors have a "fin" connecting the source and drain, which is raised above the substrate, as shown in Fig. 4. Such a raised conductive channel leads to an improved electrical response in the transistor channel.

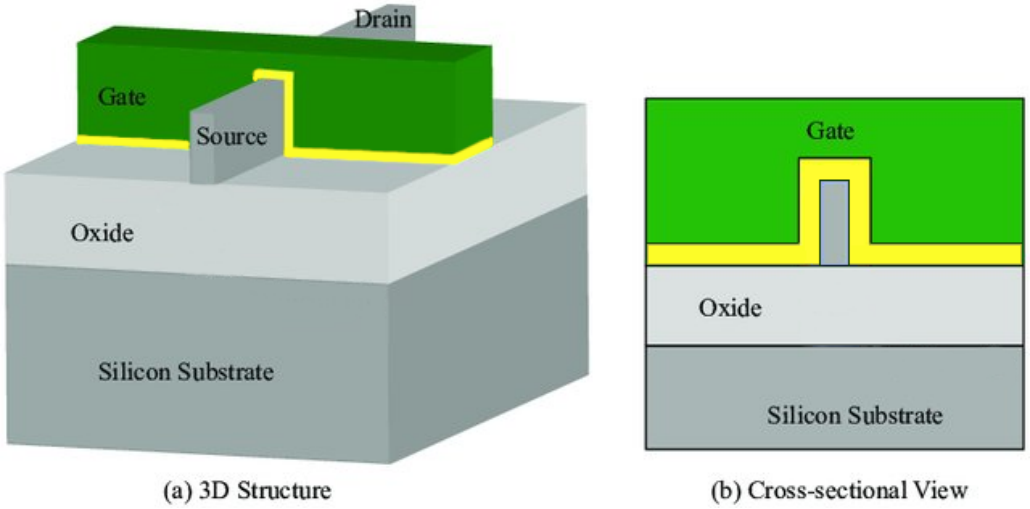

Figure 4: Schematic of a single FinFET on an insulating oxide layer. The insulating spacer between channel and gate is shown in yellow, Silicon is shown in grey and the gate is shown in green.

3.2 Task

Build a FinFET structure using the deposition/etch steps listed in Table 1, which will lead to the structure shown in Fig. 4. Be sure to wrap the level set layers correctly to advect the top level set as expected. Mask layers should not be simulated here, but are assumed to be perfect boxes on top of the material (Hint: Create them with Boolean operations). Work in units of nanometers for easier calculations and fewer rounding errors.

Process	Properties
Oxide Deposition	Start from a single 200x200nm plane (this is the Oxide).
Silicon Deposition	Deposit 50nm of Silicon
Mask Creation	Create a mask (a box) on top of the silicon with 20nm width.
Fin Creation	Directional Etch of Silicon down to the Oxide
Mask Removal	Simply delete the Mask level set
Spacer Deposition	Isotropic deposition of 5nm of Spacer (yellow in Fig. 4) everywhere
Gate Deposition	Deposit 80nm of Gate Material everywhere
Gate CMP	Use a plane 70nm above the Oxide to flatten the Gate Material
Gate Mask	Add a 10nm wide mask (box) perpendicular to the fin on top of the Gate
Gate Patterning	Etch Gate and Spacer directionally, resulting in a 10nm wide Gate
Mask Removal	Delete the Mask material

Table 1: Process Sequence for the creation of a silicon fin and gate.

- Generate a FinFET structure using the process steps in Table 1 In your first simulation assume etching is perfectly directional and deposition is perfectly isotropic. In your report, include the parameters you have used and describe how each of them influences the result.

- Introduce some isotropy to the directional processes. How does it affect the final structure?
- What happens when directional etch processes (Fin creation, Spacer Etch and Gate Patterning) have positive isotropic components? Can you think of a process which can achieve this?
- On the edges of wafers, the plasma of plasma etch steps may not be perfectly uniform. Try changing the direction of the directional processes to be slightly skewed. What happens?
- The simple Engquist-Osher advection scheme, we have discussed is often not enough to describe directional etch steps sufficiently. Try changing your advection scheme to the more sophisticated Lax-Friedrichs scheme for the directional etch steps. Usually, this integration scheme, would need calibration to the specific process, however the default settings work well for $V = \vec{n_z}$, so a simple directional etch. Hint: This can be done by using
  `lsAdvect::setIntegrationScheme(lsIntegrationSchemeEnum::LAX_FRIEDRICHS_1ST_ORDER)`