# EE 3300 Lab 9

Fall 2024

# Digital Synthesis and Automatic Layout

## Table of Contents

**Notice**

Be aware, in this lab, that naming matters. The name of your files *and Verilog modules* will be important as you go through the lab. We designed the lab to have them all be named "boolean_function." If you name them something different, the commands in this document will not work verbatim. If you use something different, that is fine, but use your name *the same way every place it is used.*

## Objective

The purpose of this experiment is to develop methods for using Hardware Description Languages for the design of digital circuits.

## Checkpoints

The checkpoints for this lab are as follows:

1. QuestaSim Testbench Results
2. Synthesized Circuit Testbench Results
3. Valid Innovus Connectivity and Geometry Reports
4. Valid DRC and LVS Results

As with all labs, these checkpoints must be shown to a lab TA before the end of your next lab section. You should include these checkpoints in your lab report.

## Background

A Hardware Description Language (HDL) is a type of programming language that is intended for describing how a digital system works, either at the behavioral or structural level. If appropriately represented, a system written in an HDL can not only simulate the functionality of the logic gates, but also provide information on the anticipated timing and power consumption of the system when implemented in silicon.

For this reason, HDLs are used extensively in the design of digital systems.

The two most widely used HDLs today are Verilog and VHDL. There is considerable similarity between these two languages and engineers are expected to be proficient in both. In this lab

experiment, we will limit our discussion to Verilog. Specifically, we will focus on how an HDL can be used for design and simulation of digital integrated circuits.

The basic process of going from an arbitrary digital system to a complete implementation of the system in silicon can be broken into six steps:

1. Design in HDL (creating the system in Verilog)
2. Logic Synthesis (using an RTL Compiler to synthesize a schematic from Verilog)
3. Logic Import (importing the synthesized schematic into Cadence)
4. Layout Synthesis (using Innovus to synthesize a layout from Verilog)
5. Layout Import (importing the synthesized layout into Cadence)
6. LVS & DRC Checks

In lab 5, you created your own Boolean function circuit in Cadence. This was a tedious and time-consuming process. Today, we will re-create that circuit by following the six steps described above.

Appendix A of the Weste and Harris text has a brief discussion of Verilog and students should become familiar with the material in this appendix. There are also numerous books and websites devoted to a discussion of Verilog. Beyond the basic introduction to Verilog discussed in this laboratory experiment, students will be expected to take the initiative to develop their own HDL skills to the level needed to support the digital design component of this course.

## Step 1: Design in HDL

Select one of the following Boolean Functions to use in this experiment

- $Y1 = \bar{A}BC + A\bar{B}C + AB\bar{C}$
- $Y2 = \bar{A}\bar{B}C + A\bar{B}\bar{C} + \bar{A}B\bar{C}$
- $Y3 = (\bar{A} + B + C)(A + \bar{B} + C)(A + B + \bar{C})$
- $Y4 = (\bar{A} + \bar{B} + C)(A + \bar{B} + \bar{C})(\ \bar{A} + B + \bar{C})$

Implementing the same Boolean expression in Verilog. Name the Verilog file “boolean_function.v” and the module, “boolean_function”. We recommend using a structural implementation instead of behavioral implementation.

Before proceeding, make a testbench for the Verilog implementation. Show your TA the testbench results, proving that the implementation works correctly.

## Step 2: Logic Synthesis

### Introduction

The purpose of this section is to introduce the process of synthesizing a digital system's schematic from an HDL description of the system. This is the second step in the six-step process described above.

### RTL Compiler

An RTL Compiler is a tool that synthesizes and then optimizes behavioral circuit elements. For this lab, the Cadence Genus software will be used for logic synthesis. Other vendors also provide synthesis tools which you may encounter in industry.

Logic synthesis translates textual circuit descriptions like Verilog or VHDL into gate level representations. Optimization minimizes the area of the synthesized design and improves the design's performance. The HDL description can be synthesized into a gate level netlist composed of instances of standard cells.

Once the gate-level netlist is finished, it can be imported into Innovus and used to create a layout. Consistency between the layout and schematic can then be verified with an LVS within Cadence.

### Standard Cell Files for RTL Compiler and Innovus

Now, the necessary files for use in Genus and Innovus must be obtained to run the digital design flow. We will be using the gsclib_svt standard cell library for this lab. This library is made using the same gpdk045 library that we have begun using in lab.

If you want to know more about the gsclib045_svt library, you can access all of the standard cell library files at the location:

/remote/cadencelib/gpdk045/gsclib045_svt_v4.7

and the documentation at:

/remote/cadencelib/gpdk045/gsclib045_svt_v4.7/doc

From the course website or Canvas, download the rtl.tar.gz file which contains a folder called 'ee330_lab9. This folder contains the LIB, LEF, and Verilog files for the gsclib045_svt standard cells. Move rtl.tar.gz into your gpdk45 folder and extract it using the following commands.

```
cd ~/Downloads
```

```
mv rtl.tar.gz ~/gpdk45

cd ~/gpdk45

tar -xvzf rtl.tar.gz
```

Now the files are downloaded and extracted, which can be confirmed by looking for the *ee330_lab9* folder in the location used above. Throughout this lab, carefully watch for path names and make sure they are pointing to the correct directories and files.

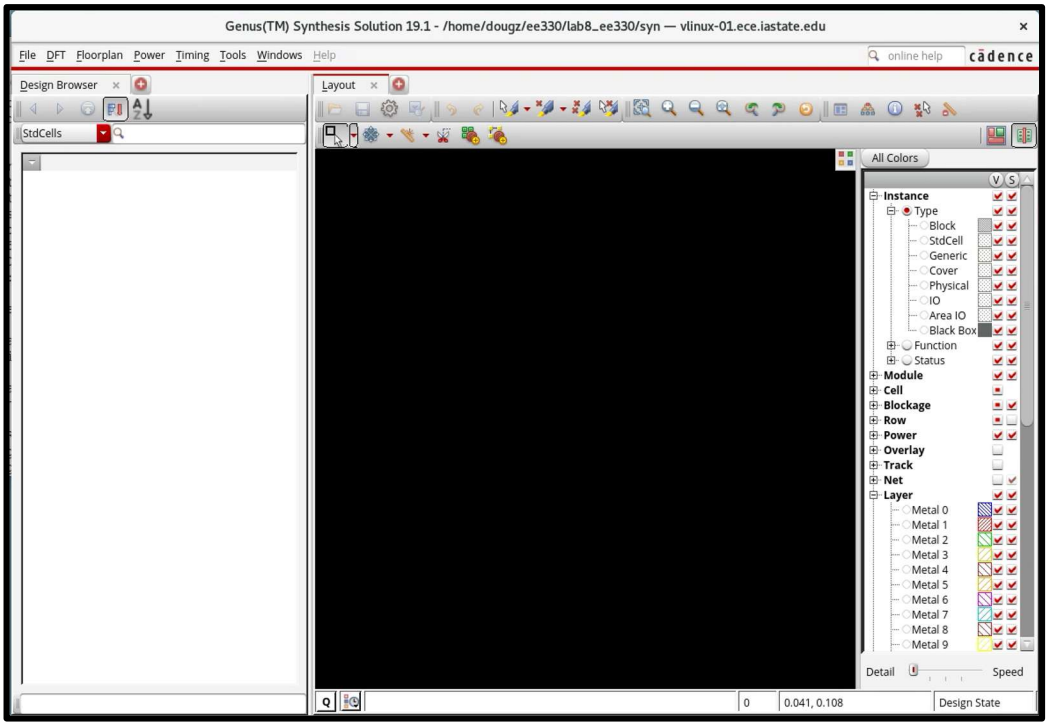
Now, copy over the Verilog file (boolean_function.v) which was created and tested earlier in this lab. The file should be located in your *ee330/verilog* folder if you used QuestaSim. Copy the file to gpdk45/*ee330_lab9/rtl/* either using the GUI or the following set of commands:

```
cp ~/verilog/boolean_function.v ~/gpdk45/ee330_lab9/rtl/
```

Change directories to the *ee330_lab9/syn/* directory and enter the following commands to begin the RTL Compiler:

```
cd ~/gpdk45/ee330_lab9/syn

genus

gui_show
```

A GUI will show and your terminal will change to the Genus command line.



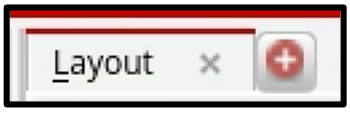
In the terminal window that has a “genus:/>” prompt, type the following to load the correct standard cell library files:

```
set OUTPUT_DIR ../synthesis_outputs

set_db / .init_lib_search_path {../lib}

set_db / .library {basicCells.lib}

set_db / .lef_library {../lef/gsclib045_tech.lef ../lef/gsclib045_macro.lef}
```

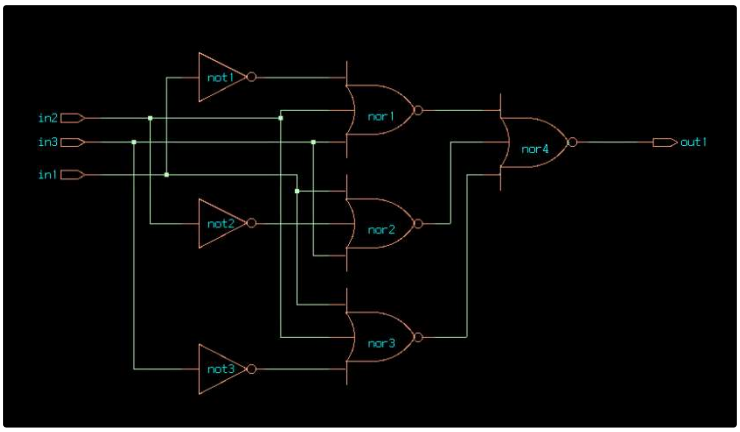
You will get multiple warnings, but no errors. Once the library is set, you need to load and elaborate your Verilog files. In this case, it was placed in the rtl folder and can be processed with the following commands:

```
set_db / .init_hdl_search_path {../rtl}

read_hdl “boolean_function.v”

set DESIGN “boolean_function”

elaborate $DESIGN
```

When Verilog is elaborated, each line is evaluated and replaced with a combination of logic gates. Go to your GUI window, there should be an empty “Layout” view. Add a “Schematic” view by clicking on the “Plus” button next to the layout view.




You should get a window with circuit blocks that represent the input Verilog. Depending how you write your code, you will get varying results. It may look some something like this:



At this point, the RTL Compiler has generated an unoptimized version of the Verilog you provided. The final step is "synthesizing" the elaborated design, which optimizes the schematic already created and integrates your design with the gsc045_svt standard cell library. While there are many optimization options available to you, we will skip them for this tutorial. The design can be synthesized with these following commands:

```
syn_gen
```

(Synthesis for a generic technology target.)

```
syn_map
```

(Synthesize from generic gates to gates from the chosen target technology library.)

```
syn_opt
```

(Optimize the mapped design.)

The GUI window will not automatically update to show the optimized design. To refresh the GUI you need to select your design from the left menu (it will be named "Hier Cell"), then right click -> Schematic View -> In Main. Exporting the synthesized Verilog file as well as a timing file used for automatic layout is done with these two commands:

```
write_hdl > ${OUTPUT_DIR}/${DESIGN}_synth.v
```

```
write_sdc > ${OUTPUT_DIR}/${DESIGN}.sdc
```

Genus also can create reports which give estimates about the timing performance, power usage, and area of the circuit (there are also many more available). If you take EE4650, you will learn more about these reports. For now, just take a look at them to get familiar. To view these reports, use the following commands:

```
report_timing
```

```
report_power
```

```
report_area
```

Congratulations! You have now synthesized your Verilog into a netlist which corresponds to a standard cell library! You may now close the Genus GUI. Now, we need to load the synthesized code into Virtuoso for testing and verification that it works as intended.

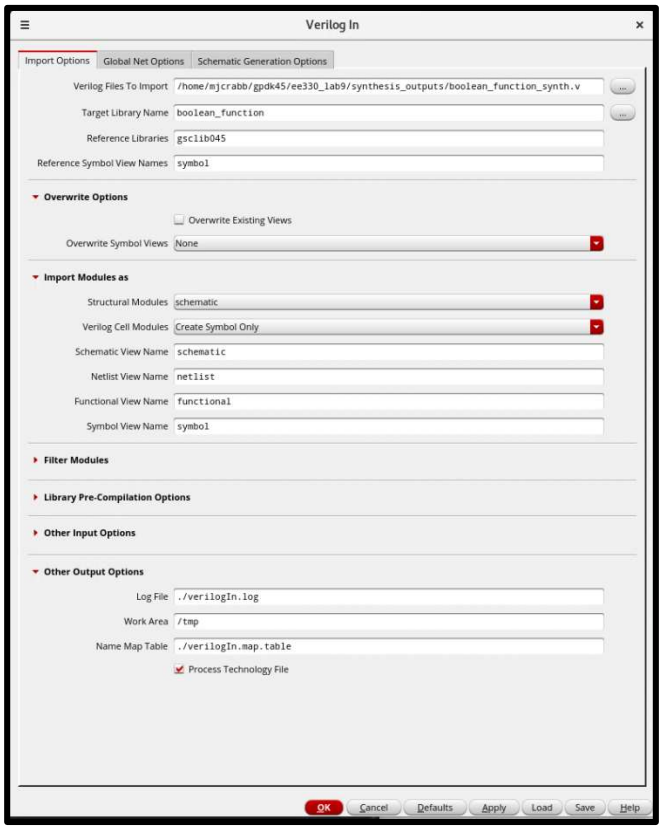
## Step 3: Logic Import

After synthesis, we want to import the gate-level netlist created in RTL Compiler into Cadence Virtuoso. This is the third step in the six-step process described above. Open Virtuoso and in the library manager create a new library with the name boolean_function (or whatever name you want to use for the library for this lab). When the technology file options pop up select "Attach existing technology library" and then select gpdk045.

Once the library is created, change to the CIW (where the errors and other messages appear) select File → Import → Verilog.

The following screen will appear. Fill it in as is shown in the screen shot below, except replace the Verilog file with your synthesized Verilog file created by Genus and use your own library name. This file will end with _synth.v if following the above steps.

**Make sure all text fields match, with only differences existing if you have named your files differently.**



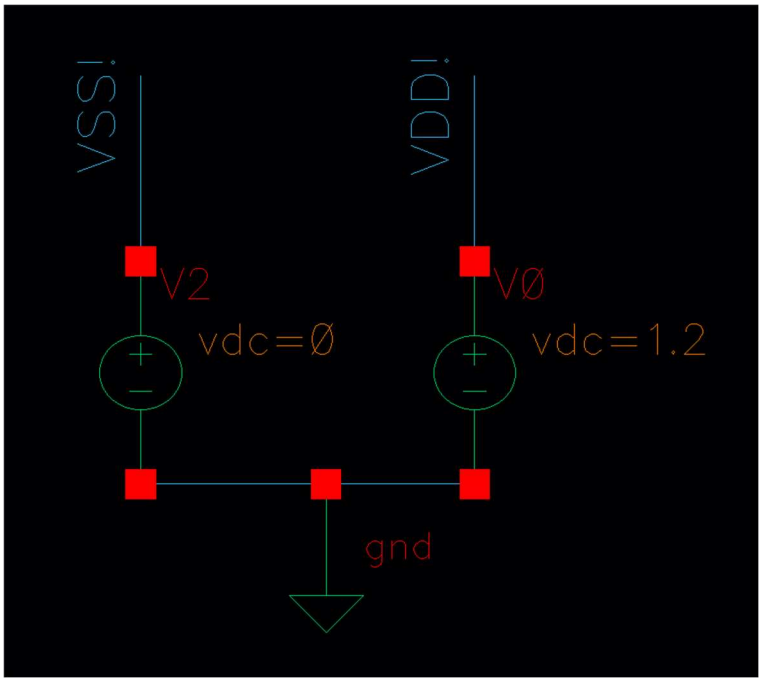

The import process should complete and may have a few errors or warnings. Ignore for now. Go to the schematic cell view created by the import process and open in Virtuoso. The synthesized optimized design should appear with the lowest level of hierarchy being the standard digital blocks

(gates) made of transistors in the gpdk045 process. If not, something is wrong. **Retry the import process or possibly synthesis before asking for help from the TAs.**

## Validation of Schematic

A symbol for your boolean_function should have been created. Create a testbench for the import boolean_function to double check that the generated netlist is functionally equivalent to the Verilog that you wrote and simulated in the previous lab.

The standard cells use global nets named "VDD!" and "VSS!" for $V_{DD}$ and $V_{SS}$. To supply the cells with the proper supplies, create voltage sources to generate $V_{DD}$ and $V_{SS}$ and label the outputs "VDD!" and "VSS!" – an example is shown below. The standard cells in this library are made for a 1.2V supply. This means $V_{DD}$ = 1.2V and $V_{SS}$ = 0V.




**Show your testbench schematic and correct results to your TA for the second checkpoint.**

## Step 4: Layout Synthesis

### Introduction

In a typical digital design flow, a hardware description language is used to model a design and verify the desired behavior. Once the desired functionality is verified, the HDL description is then taken to a synthesis tool such as RTL compiler. The output of RTL compiler is a gate-level description of the desired circuit. The next step is to take this gate-level description to a place-and-route tool that can convert it to a layout level representation. This is the fourth step in the six-step process described above. In this section of the lab, you will be introduced to a place-and-route tool called SoC Innovus. For the purposes of this tutorial, it will be assumed that you have a synthesized Verilog file ready to be placed and routed.

### SoC Innovus

SoC Innovus, in its simplest form, allows a user to create a layout from a synthesized HDL file. Once the synthesized file is imported, Innovus provides a variety of tools to create a floorplan for your design, place standard cells, and automatically connect power rails and local routing. While this lab only provides a brief overview of Innovus, there are many other optimization and layout options available. If you are interested in learning more about Innovus, you may be interested in pursuing EE4650.

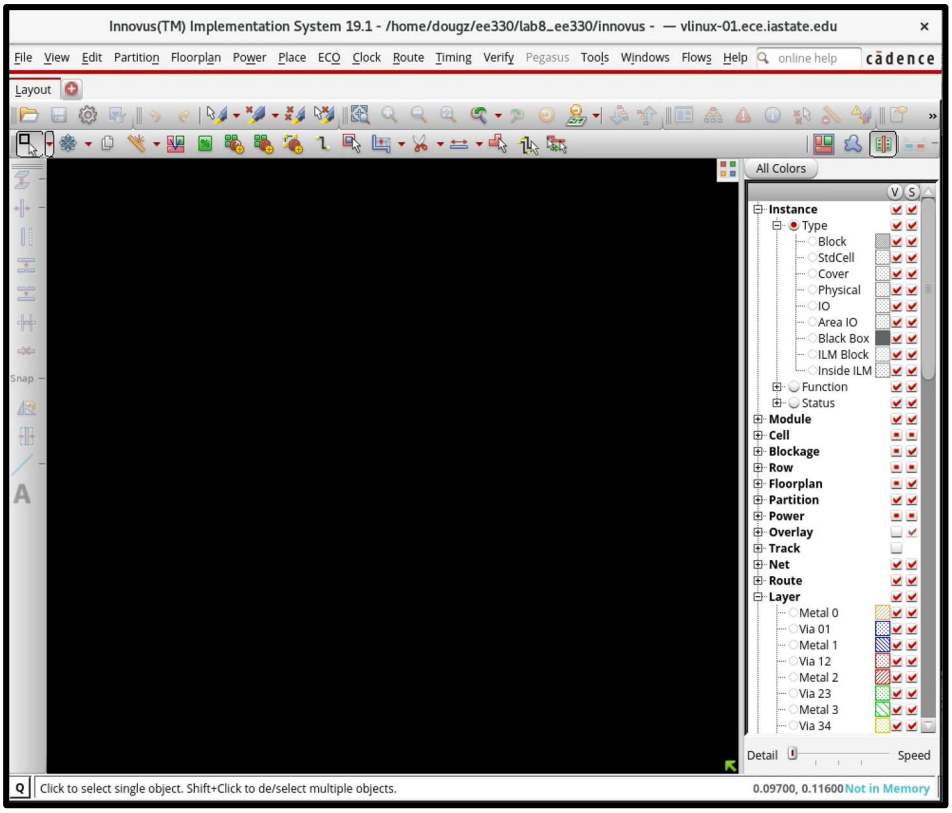
### Running SoC Innovus

Move into the *innovus* directory in the *ee330_lab9* folder and start innovus using the following commands in the terminal:

```
cd ~/gpdk45/ee330_lab9/innovus
```

```
innovus
```

The command prompt will change to innovus> and you will be presented with a GUI, which has been shown below.

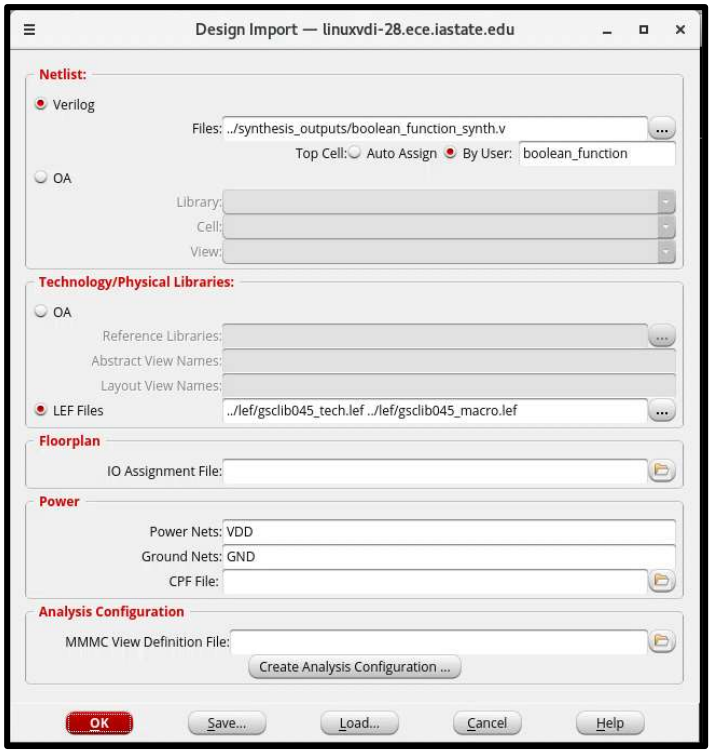



## Importing Synthesized Verilog

Once Innovus is initialized, you need to import your synthesized HDL description file. Go to File → Import Design and under Files, select ../synthesis_outputs/boolean_function_synth.v

Do not import the OA under Technology/Physical Libraries, instead clock the LEF Files button and enter ../lef/gsclib045_tech.lef and ../lef/gsclib045_macro.lef separated by a space. Make sure the "…_tech" file is listed first.

Under Power Nets put VDD and for Ground Nets put GND. The form should be filled out like the one below. Click OK to import the design.




## Floorplanning

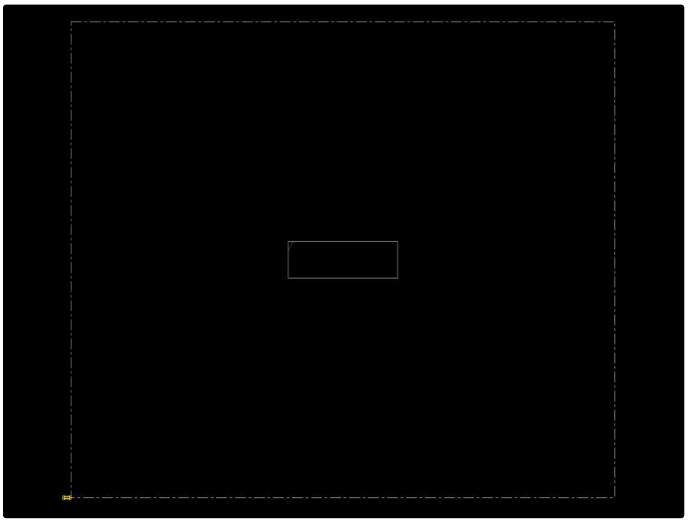
The Innovus GUI should now display an empty "die" and the information box in the bottom-right corner should read "Design is: In Memory." The next step is to specify a floorplan for your layout. Floorplanning is done to specify the dimensions of your layout and spacing between the core (or area where the standard cells are placed) and power/signal routing. Open up the floorplanning options by clicking on the *Floorplan* menu and selecting *Specify Floorplan*.

Options that you will need to pay attention to are:

- Aspect ratio – height/width ratio of the core.
    - Suggested: 1.0 (this will change to the closest workable number)
- Core Utilization – the amount of core area used for placing standard cells. A larger value means that your design will be fairly compact but routing may be difficult, if not impossible. Smaller values will drastically increase the design area.
    - Suggested: 0.5 (this will change to the closest workable number)
- Core Margins by – specifies the distance between the core and edge of the die. The margin area needed is proportional to the number of input/output (I/O) pins.
    - Suggested: 10 for core to left, right, top, and bottom

You are encouraged to play around with these settings.

Once these values are entered, the layout die/core spacing and aspect ratio should be updated. It may look similar to the image below.



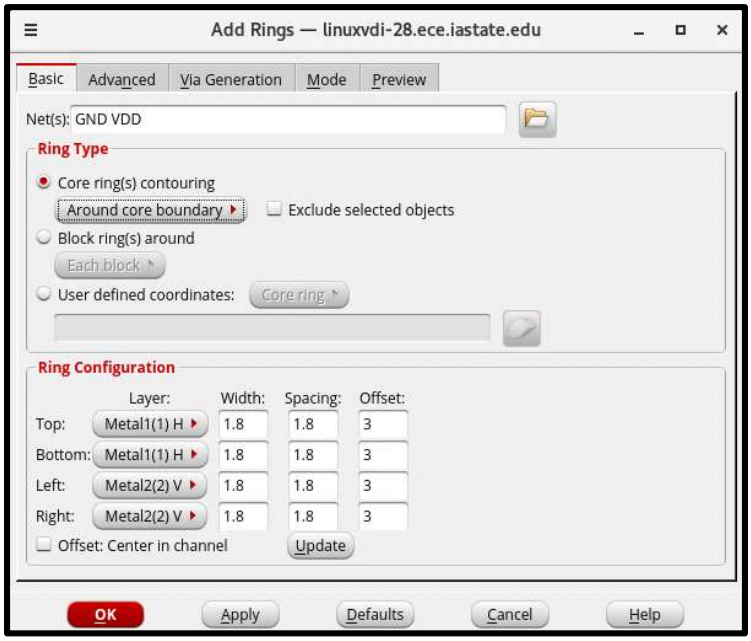
## Power Planning

In order to connect your supply voltages to the standard cells, power and ground need to be available on all sides of the die. This is done by adding power rings and specifying global net connections.

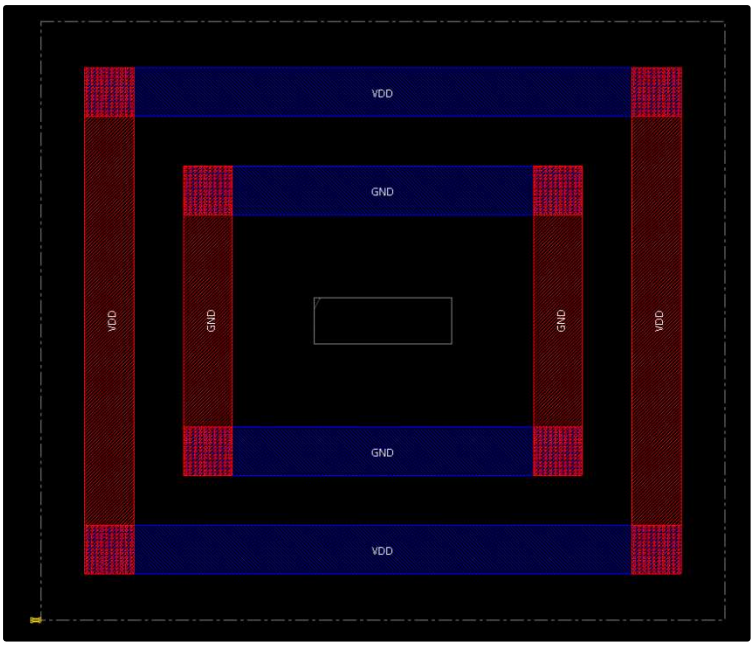
Select the *Power* menu and click on *Power Planning* → *Add Rings*.

A dialog box will pop up and allow you to specify the ring type and configuration. You can plan power in various modes to facilitate various design styles. Since it is preferable to have the routing done inside the power rings, it is a better idea to have them around the I/O boundary so choose this option. If it is not already entered, add "GND VDD" (without quotes) to the Net(s) field.

In the Ring Configuration section, replace the "Offset" for each side with "3". Leave everything else as "1.8"




Once finished, you should see two rings that run inside the I/O boundary. It should look like the following:




Connection to the global nets is done automatically if the nets are given the same names as was done in the setting of the power nets above. If all was done correctly, power planning should be finished.

## Placement

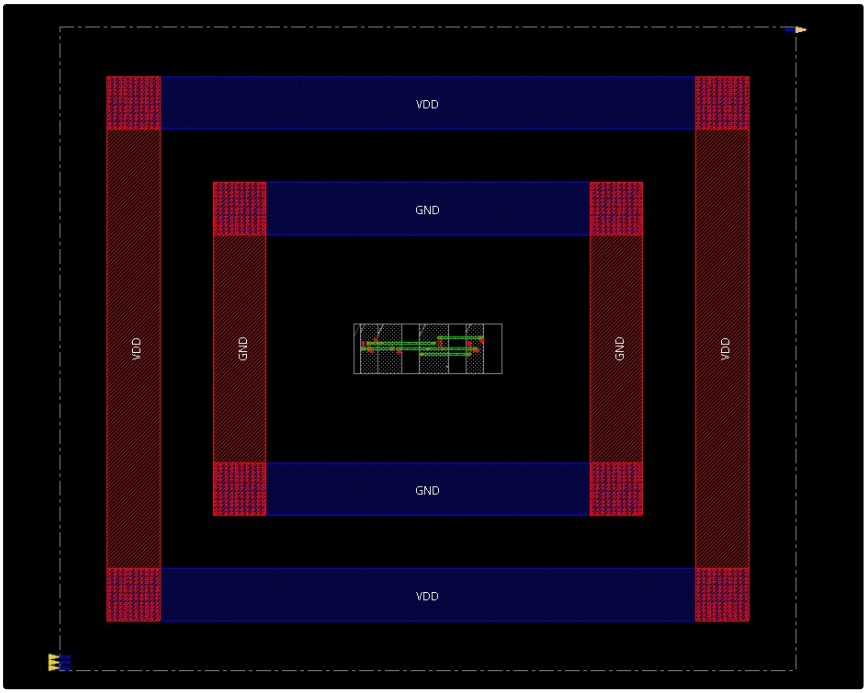
Now we are ready to perform the placement. At the end the cells used in the design will be placed so that they don't overlap and there is enough space for routing in between them. Select the *Place* menu and then *Place Standard Cells*. In the dialog box that appears, leave the default settings and hit OK. In the main GUI window, you can verify that the standard cells were placed by selecting the Physical View Icon in the upper-right:


You should see the cells placed in the middle of the core. The IO pins have shifted position accordingly as well.

If placement fails, it is most likely because Innovus cannot place the cells in the given area without causing gate overlaps. To solve this problem, you must start over by reinitializing the floorplan with relaxed constraints. The best constraint to change is the row utilization factor. By lowering this factor, you can increase the likelihood of placement without overlaps.

## Pin Assignment

The next step is to assign pin locations for your IO pins. To do that go to *Edit* → *Pin Editor*. This opens up a window which allows you to choose exactly how your pin is setup (which metal layers, location on die boundary). For our purposes we will pick either metal layers 1 or 2, and place the pins anywhere across the boundary (specifying top, left, right, or bottom is more than enough). Select the pin from the menu on the left, then choose "Assign location" under Location tab and pick a location. We will need to do that for all 4 pins. If that is done correctly, arrows indication pin direction should show up on the die's boundary.

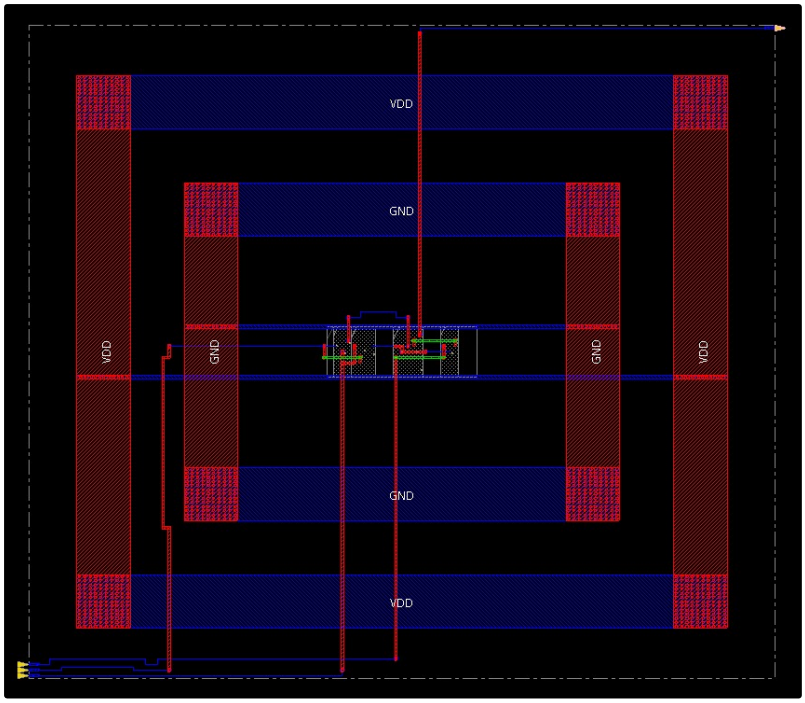



## Routing

The built-in routing options within Innovus can now be used to connect the supply nets and create interconnects between the standard cells.

Power and ground routing is performed with the “Special Route” tool. This can be found by selecting the *Route* menu and then *Special Route*. In the dialog box that appears, unselect *Block pins, Pad pins, and Pad rings*. After clicking OK, you should see horizontal and/or vertical metal power routing added to your design.

Once power routing is complete, the remaining connections between standard cells and the die I/O can be routed using the “NanoRoute” tool. Select *Route* and then *NanoRoute* and finally *Route…* to enter the dialog box. You do not have to make any changes and can click OK. Your design should now resemble the following

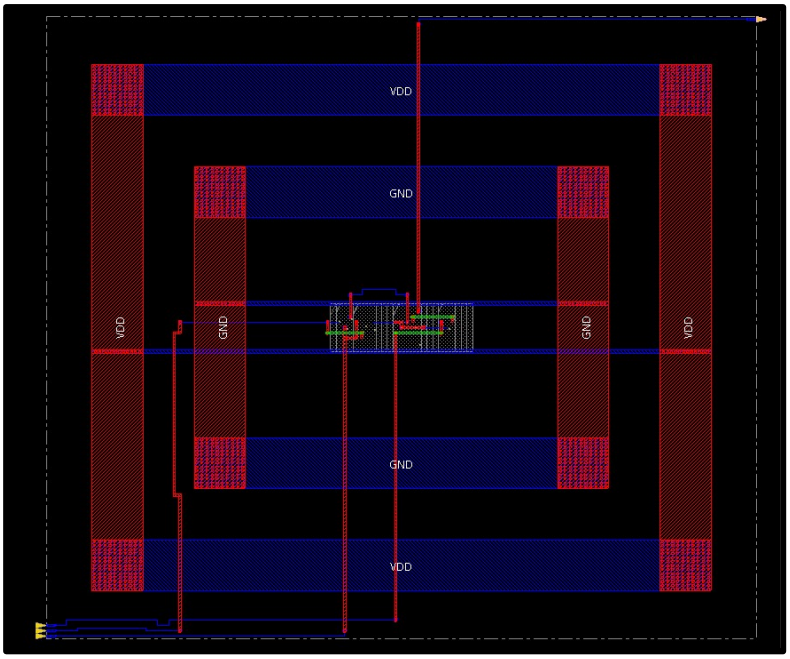



If you see white boxes and spacing errors when you route, your design is too constrained. Consider increasing the offset of your power rings by increments of 0.15.

## Filler Cells

If you import the complete layout from Innovus into Cadence at this point, you may have DRC errors due to nwell layers not being spaced far enough. You can fix these errors by hand in Cadence or you can add pre-designed filler cells with Innovus. These cells provide continuity for nwell layers and power rails between placed cells.

Filler cells are placed by selecting the *Place* menu and then *Physcial Cell* → *Add Filler*. In the dialog box that appears, click "Select" in the Cell Name(s) field and select the appropriate fill layers. Filler cells begin with "FILL". You can now click OK to fill in the blank regions in the core of your layout.




## Verification

Before exporting your layout, you need to verify that there are not any geometric errors that occurred during placement or connectivity errors from routing.

To run geometry and connectivity verification, go back to the terminal with Innovus running and enter the following commands:

```
verifyGeometry
```

```
verifyConnectivity
```

The results of the verification will be shown in the terminal and reports will also be generated in the directory where you are running Innovus.

If errors occurred, they are likely the result of your area being too compact. If this is the case, consider increasing your power ring offsets by an increment of 0.15 and/or increasing your Core Margins slightly.

**Show your TA the valid Innovus Connectivity and Geometry reports from the Terminal for the third checkpoint. Also include screenshots of these reports in your lab manual.**

### GDS Export of Layout from Innovus

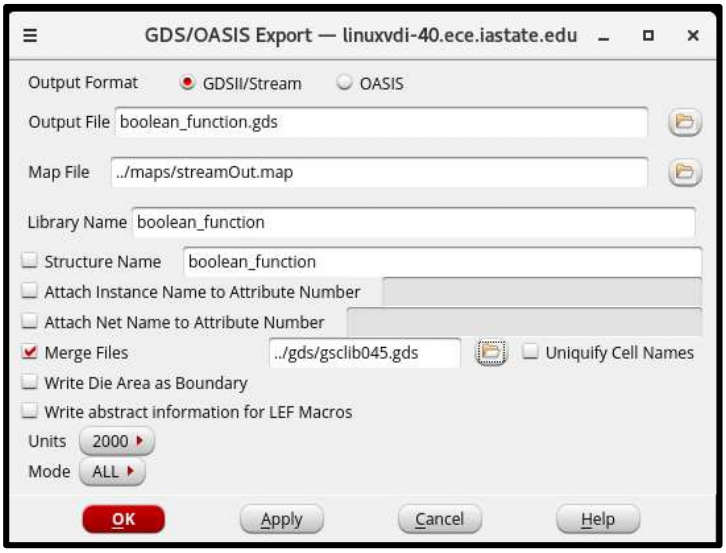
Once the layout is complete in Innovus (post-routing and post-filler cells), then you need to perform an export of the layout. This is done in Innovus by going to *File → Save → GDS/OASIS…*

In the window that appears, select the *GDSII* radio button. Then type an output file name in 'Output File' field such as *boolean_function.gds*. In the map field select the 'streamOut.map' file located in the *maps* directory.

In the Library Name, type the Library in Cadence you would like to add the design to. Use the same library you used to import the Verilog schematic.

Select the 'Merge Files' box and enter the *gsclib045.gds* file from the directory *ee330_lab9/gds*

Leave all other fields as default making sure the units are '2000' are mode is 'ALL'. Click OK. Look at the terminal and make sure Stream out has finished successfully.




## Step 5: Layout Import

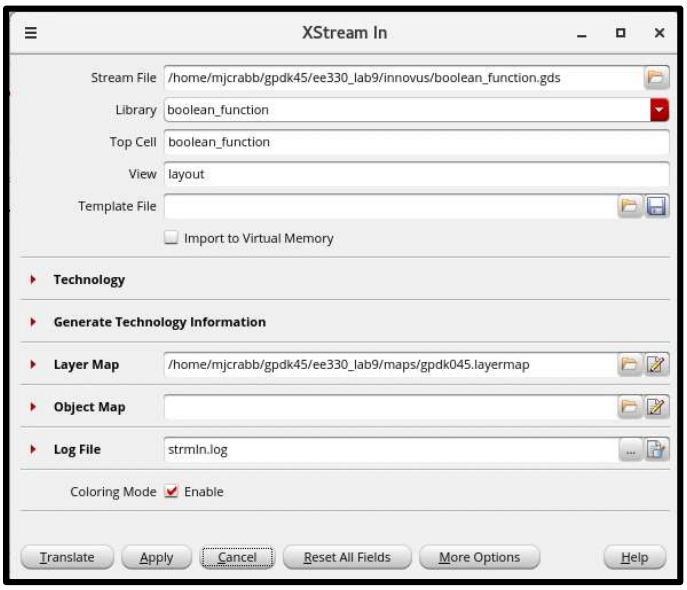
### GDS Import of Layout into Virtuoso

After creating a layout in Innovus and creating the GDS file, open Virtuoso. Click on the CIW (Virtuoso window where errors and messages appear) and click *File → Import → Stream*. This may produce an error the first time. Try again and it should work and bring up an import window.

On the "stream in" window choose the GDS file created in Innovus as the 'Stream File' file.

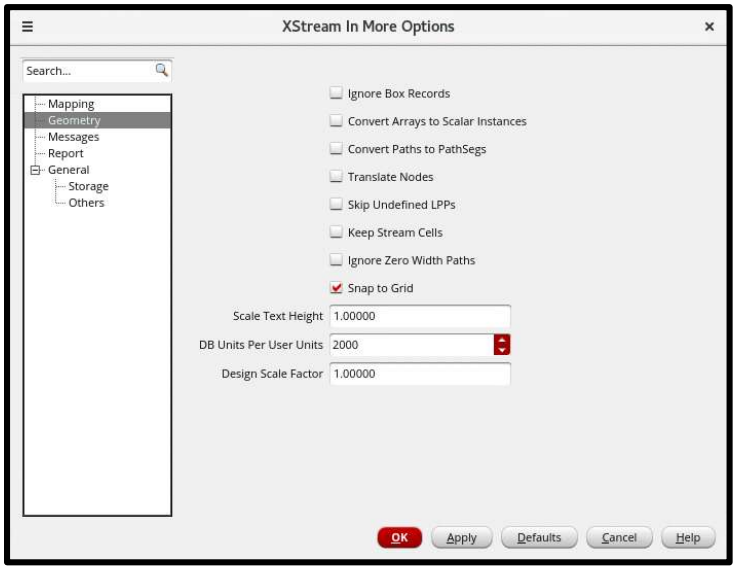
Under library, type the name of the library you would like to save the module in. Note, this does need to be the same as the name of the library given in the stream out in Innovus. Then give a name to the top level cell once it is imported.

Under "Layer Map", select the *gpdk045.layermap* file. This is located in the *ee330_lab9/maps* directory.




**Do not click Translate!** First, go to *More Options*, then in the *Mapping* tab and check the "Replace [] with <>" box.


Now, go to the Geometry tab. Select Snap To Grid and enter the DB Units Per User Units as 2000. Click OK.




Now we are ready to translate the file and import the design. Click *Translate*. If a box to save the libraries comes up make sure to save the libraries as something. The process will take a little bit. When complete there should be a log file and a message box that appears. This should be 0 errors and only a few warnings, if not 0.

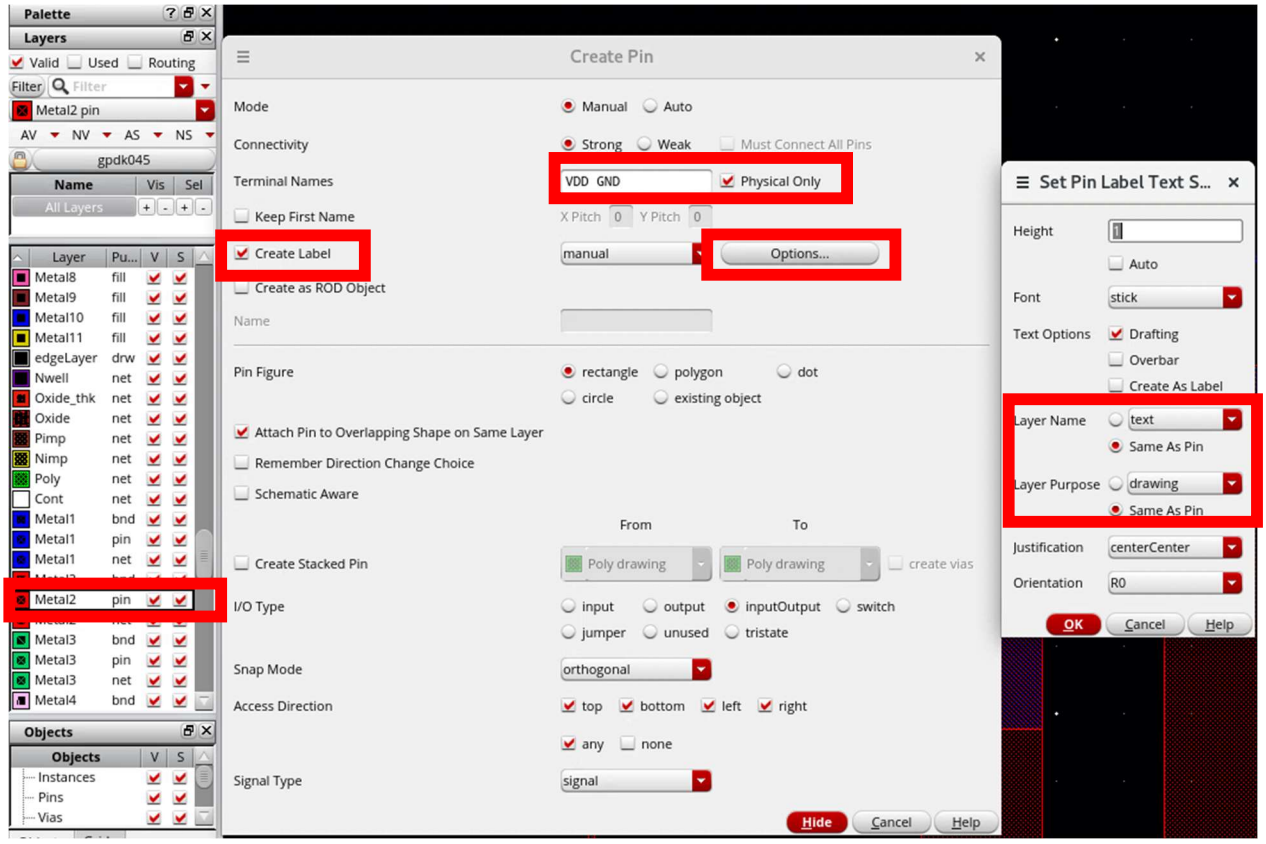
Go to the library created in the Library Manager and try to open the layout of the design. The standard cells should appear. Press *Shift + F* to see the layout view.

## Step 6: LVS & DRC Checks

### DRC and LVS error corrections

When the design is exported, Innovus does not make pins for the VDD and GND nets and the io pins are made as labels rather than pins. To add the VDD and GND pins, select Create → Pin. Add the names VDD and GND to the *Terminal Names* box and check the *Physical Only* box. Also check the *Create Label* box and click on the *Options…* button that is in line with the checkbox. Check the two *Same As Pin* options. Make sure you have the correct layer selected in the layout window (Metal1 Pin or Metal2 Pin) then click *Hide*. Create the pin like you would make a rectangle on the correct net. **Place the label on top of the pin you made.**

If you desire, you can also delete the labels for the other pins and repeat the process above to create them.




Now, you are ready to run DRC and LVS. The gpdk045 library uses Cadence Pegasus for DRC and LVS. See the additional guide on how to run these on the course website.

**Show your TA a valid DRC and LVS for your layout as a checkpoint.**

Congratulations, you have synthesized your digital circuit! In your lab report, please provide a summary of the steps required for synthesizing and performing layout with Cadence Genus and SoC Innovus. Explain what the main steps do and provide screen shots of major steps as well as the two verification steps at the end.

If you want to learn more about the digital design flow, what the purposes of the .lef, .lib, etc. files are, and how to perform further optimizations and analysis, check out this YouTube series:

https://www.youtube.com/watch?v=GIPhBfenqMc&list=PLZU5hLL_713x0_AV_rVbay0pWmED7992G

The Iowa State EE4650 class also covers more in depth elements of the digital design flow.