EE 330 Lab 9

Spring 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. ModelSim 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.

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.

Setup Standard Cell Library

First, download the rc.zip file from the course website. Then, move them to the TSMC folder for use. They can be extracted through the GUI.

. Go into the "rc" folder. It should have 3 folders, named libdir, rtl and syn

• libdir: contains the library files the tool will use.

• rtl: contains the Verilog codes needed to be synthesized. Please copy the Verilog file which you made for the first checkpoint to this folder and rename it as "ALT_MULTADD.v". Please do not include your test bench file because that is for simulation only.

• syn: contains run_dir (which holds the results of running synthesis) and scripts (which holds the scripts for running synthesis). More importantly, in the scripts folder, there are 3 files.

They are:

1) design.sdc: contains the constraints you want to add to the design. They are already set. Please note in the Verilog file you made in Lab 1, if you changed the port names that are defined in the Lab 1 instruction, you need to modify this file to adapt to your port names.

2) read_rtl.tcl: is a script used to read in your Verilog file. Please make sure the names of all Verilog files that you have written are included in the script. If you have more than one Verilog files to be read in, you need to add lines in this file to read all your Verilog files.

3) run_synth.tcl: is the top level script to drive the synthesis tool. This file will use the other 2 files. Note that this startup process is meant to be done only once. The only time you need to make changes will be when you have a different Verilog file to synthesize. You are to copy the 'v' file into the 'rtl' folder and rename it as "ALT_MULTADD.v"

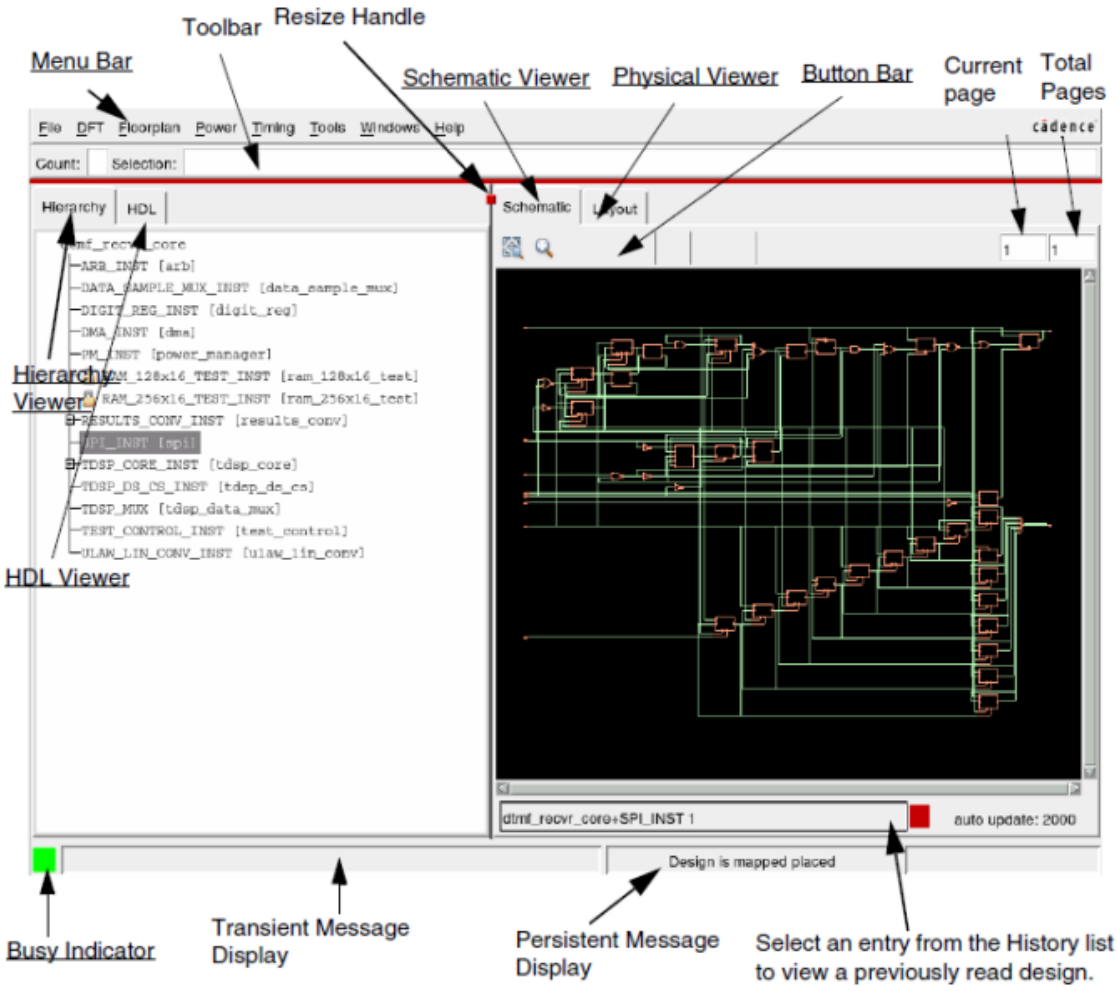
Starting Geus Synthesis

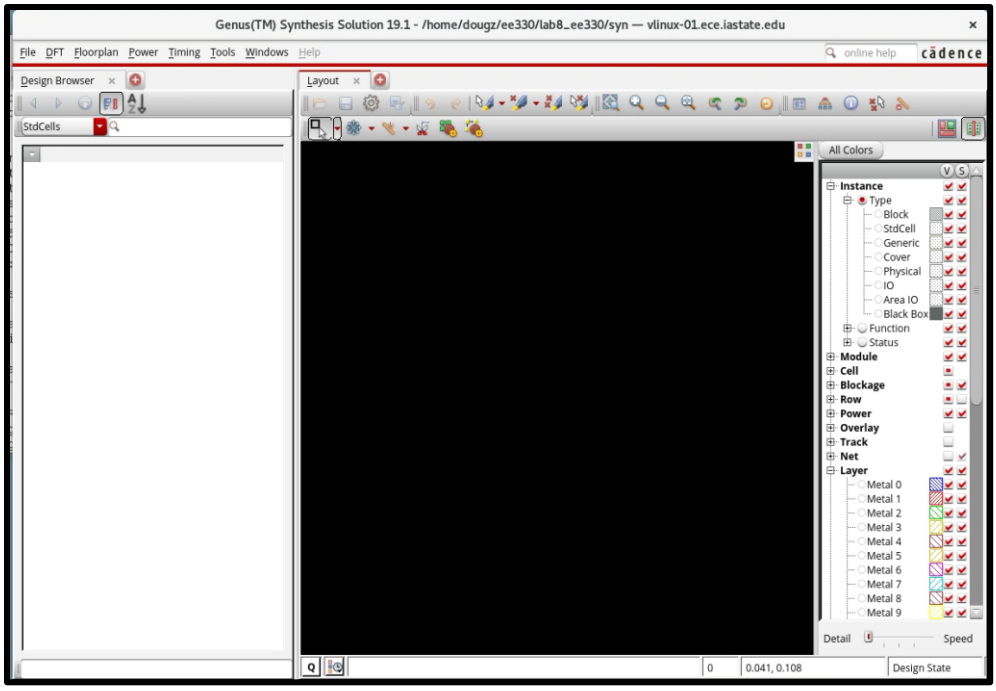
Navigate to the and enter the following commands to begin the RTL Compiler:

```
cd ~/TSMC180/rc/syn

genus -legacy_ui

gui_show
```

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

You can play around with the GUI after synthesis to see what various tools can do.


Toolbar
Resize Handle
Menu Bar
Schematic Viewer
Physical Viewer
Button Bar
Current page
Total Pages
File DFT Floorplan Power Timing Tools Windows Help
cadence
Count:
Selection:
Hierarchy
HDL
Schematic
Layout
Hierarchy Viewer
HDL Viewer
dtmf_recvr_core+SPI_INST !
auto update: 2000
Design is mapped placed
Busy Indicator
Transient Message Display
Persistent Message Display
Select an entry from the History list to view a previously read design.



You can perform synthesis by running the script that is already made for you. Ensure that you have placed the appropriate HDL file (Verilog file) in the right folder (rtl).

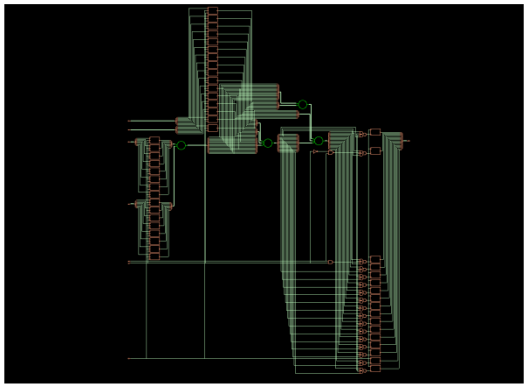
To load your design from the terminal, provide the following command:

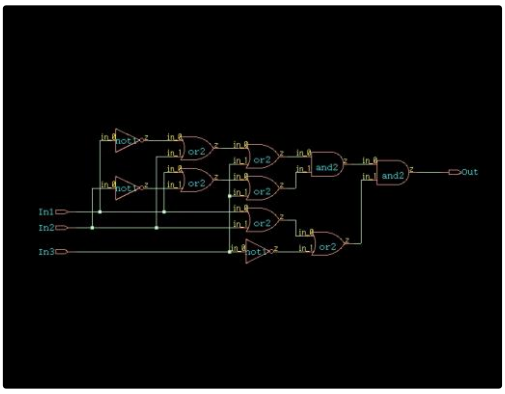
source scripts/run_synth.tcl

The tool will do the synthesis job for you. Just wait for the result (It might take some time). A gate level schematic will be shown in the gui window.

To see the synthesized schematic, right click on the “Heir Cell” Go to “Schematic view module” and then “in New” The schematic should show up. Zoom in to see the input and output ports. It should be similar to what you expect.

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, RTL Compiler has generated an unoptimized version of the Verilog you provided. 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 is done with these two commands:

```
write -mapped > ${DESIGN}_synth.v
```

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

Congratulations! You have now synthesized your Verilog into a netlist which correspond 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.

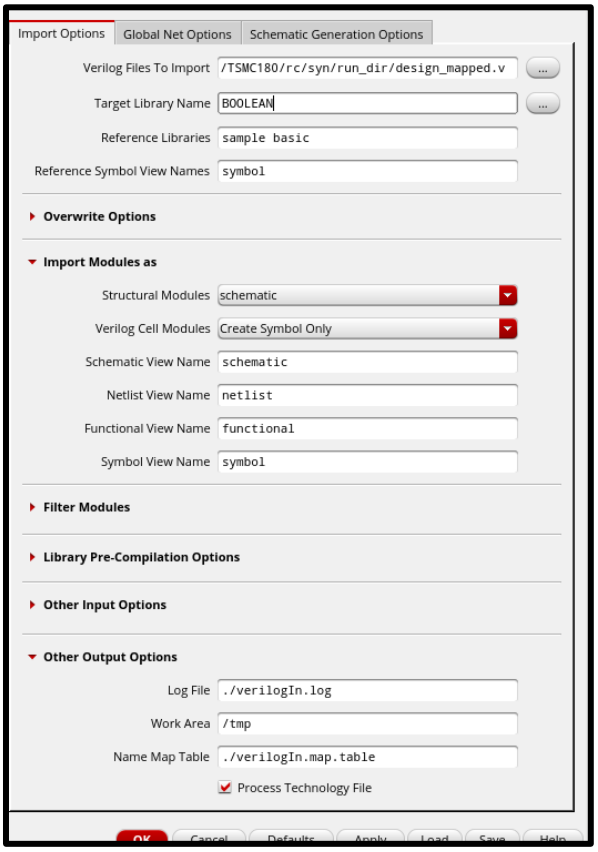
Step 3: Logic Import

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 'tsmc18rf'.

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 TSMC 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. For V_{DD}, use the "vdd" component discussed in previous labs. A V_{SS} component is not necessary (V_{SS} is, by default, shorted to GND by Genus). Submit your testbench results as a 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 CPR E 465.

Running SoC Innovus

In the *rc* folder, create an additional folder. Make note of this folder's name; "Innovus" would be good.

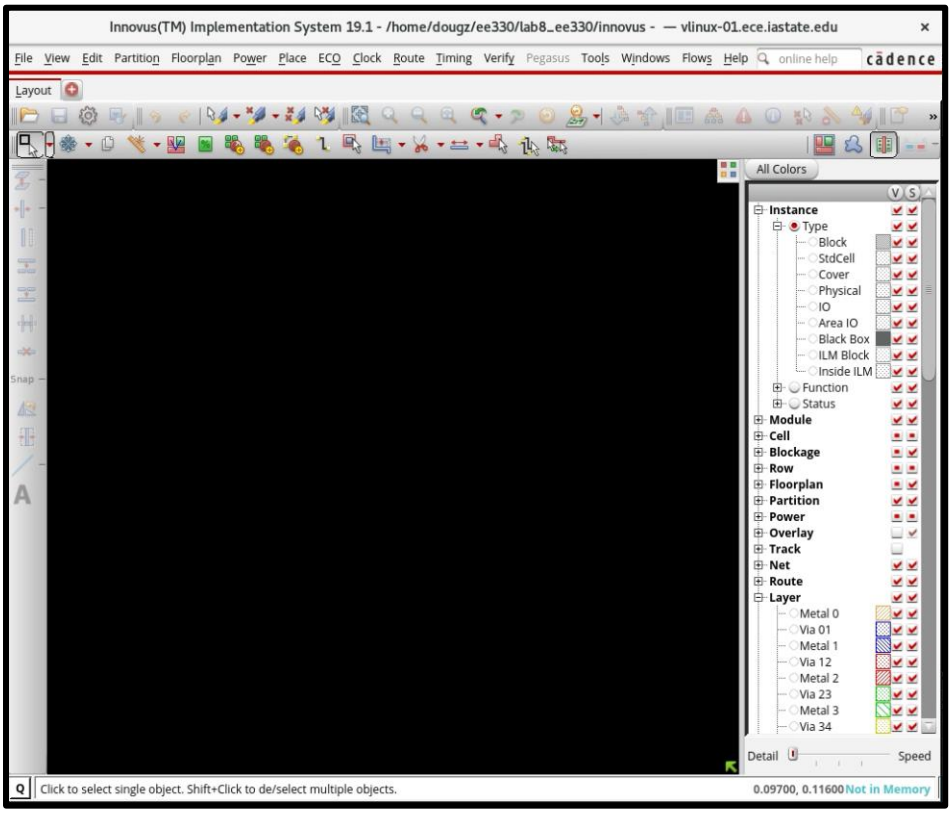
Download the innovus_lab zip file from the class webpage and extract its contents into the Innovus folder you just created.

Go to "run_dir" folder within innovus_lab

Type "innouvus" to start.

```
innovus
```

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



- We need to set the paths to the libraries and constraints in the MMMC.tcl file before we do anything.
- In order to do this, modify these lines of code from the tcl file:
- create_library_set -name typical_lib -timing {“Add path to tcbn65gpluswc.lib here”}
  create_constraint_mode -name typical_constraint -sdc_files {“Add path to design_mapped.sdc here”} -ilm_sdc_files {“Add path to design_mapped.sdc here “}

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 ../syn/boolean.v

Do not import the OA under Technology/Physical Libraries, instead clock the LEF Files button and choose ../lef/tcbn….

Under Power Nets put VDD and for Ground Nets put VSS

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*.

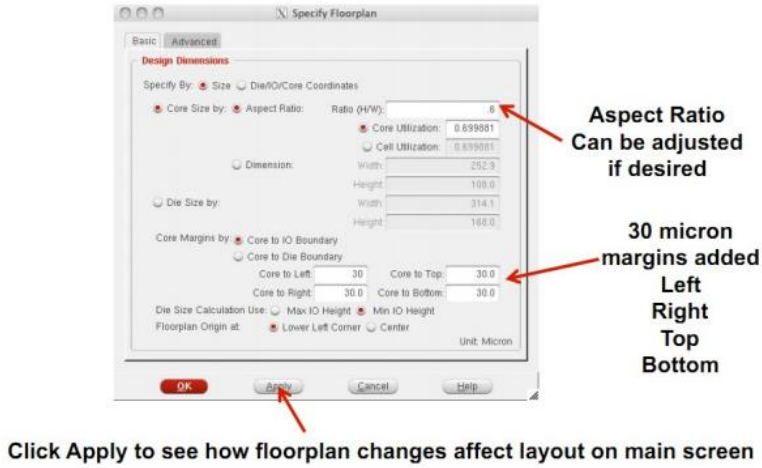
Options that you will need to pay attention to are:

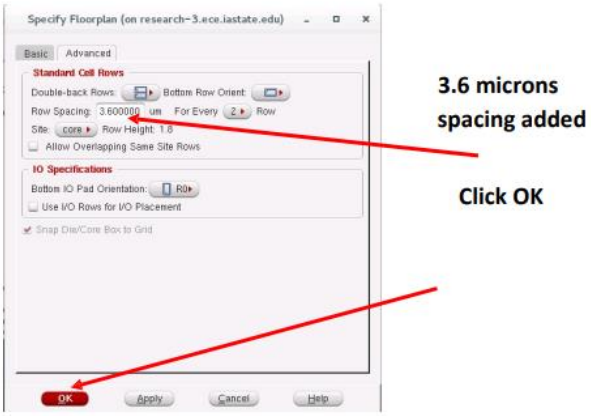
- Aspect ratio – height/width ratio of the core.
    - Suggested: 1.0 (this will change to the closest workable number)
- Core Utilization – the amount of core area used for placing standard cells. A larger value means that your design will be fairly compact but routing may be difficult, if not impossible. 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: 20 for core to left, right, top, and bottom

You may follow these steps

- Select Floorplan→Specify Floorplan Set the parameters and options for both "Basic" and "Advanced" tabs using the values as shown in the figures below. Please note that these parameters will affect your layout result.





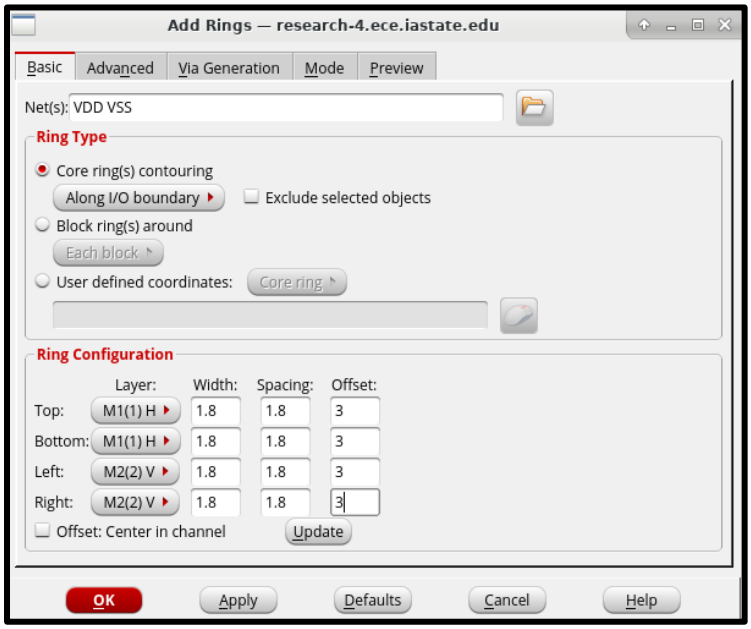
Power Planning

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.

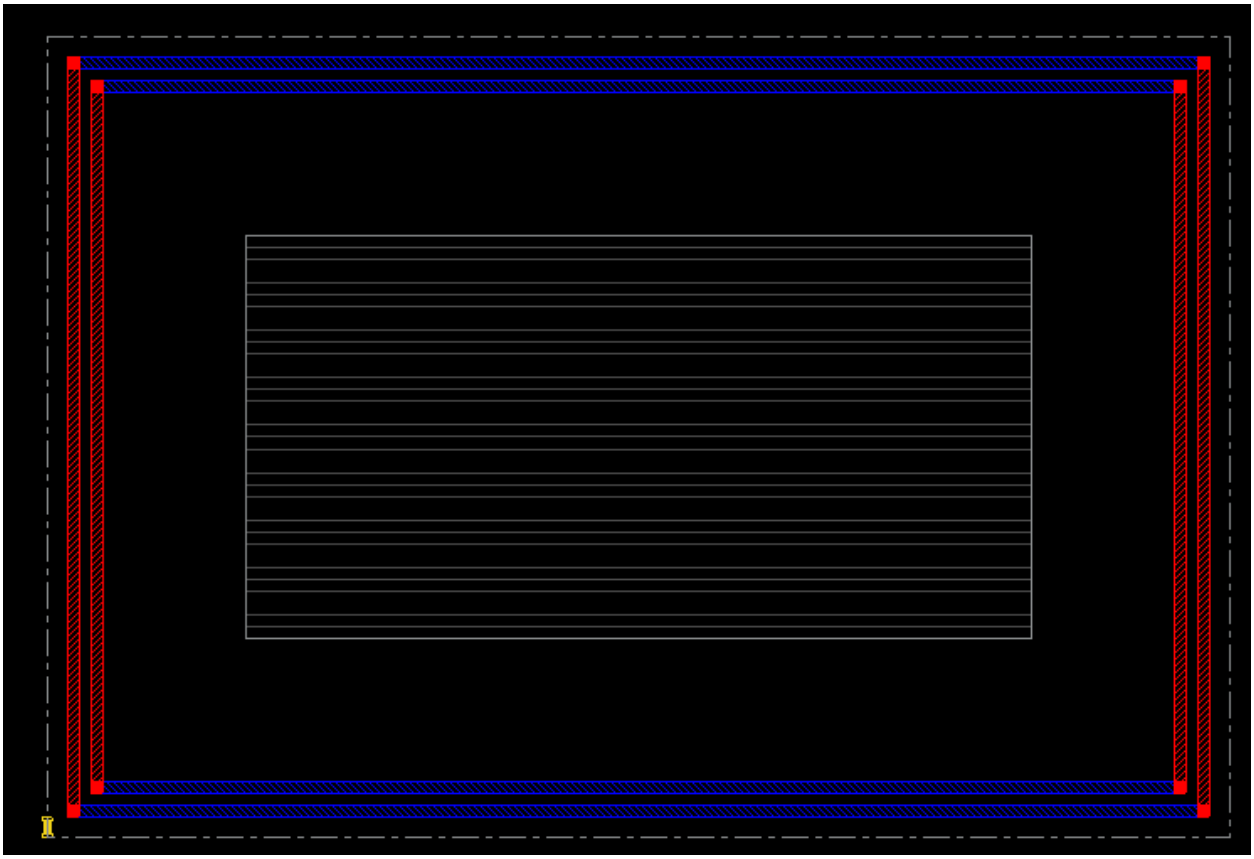
Select the *Power* menu and click on *Power Planning* → *Add Rings*.

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 "VSS 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:



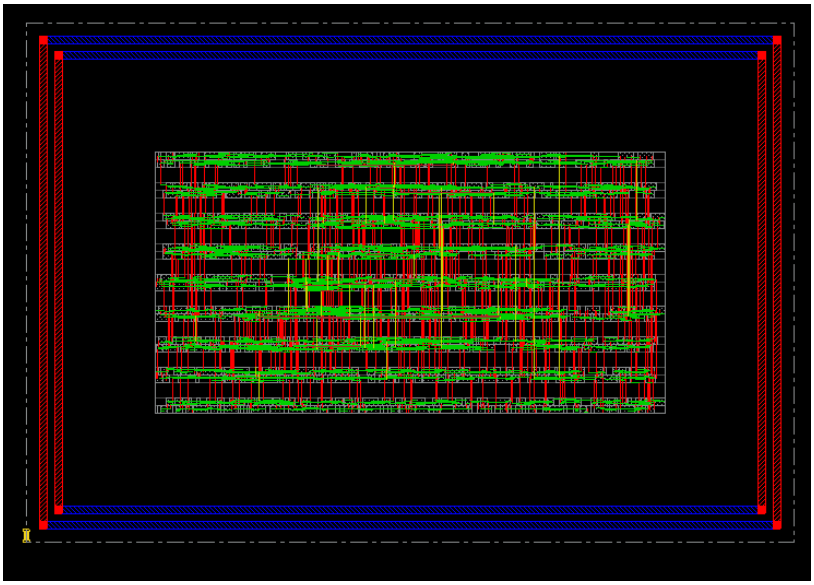
Connection to the global nets is done automatically if the nets are given the same names as was done in the setting of the power nets above. If all was done correctly, power planning should be finished.

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:

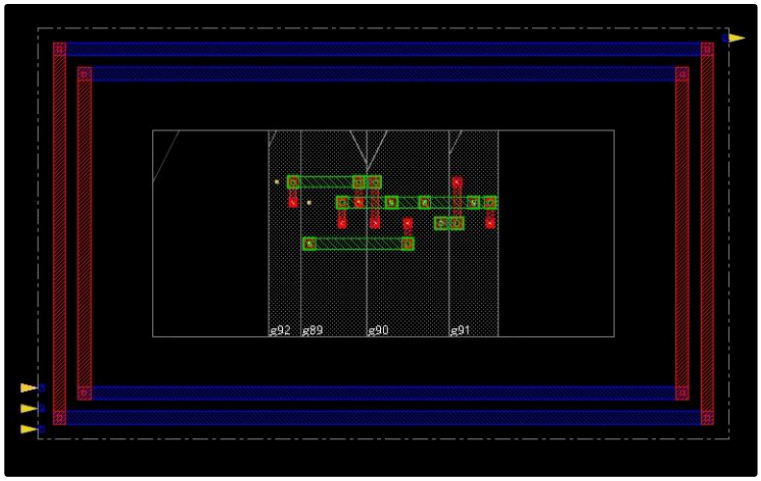
You should see the cells placed in the middle of the core. The IO pins have shifted position accordingly as well.

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.

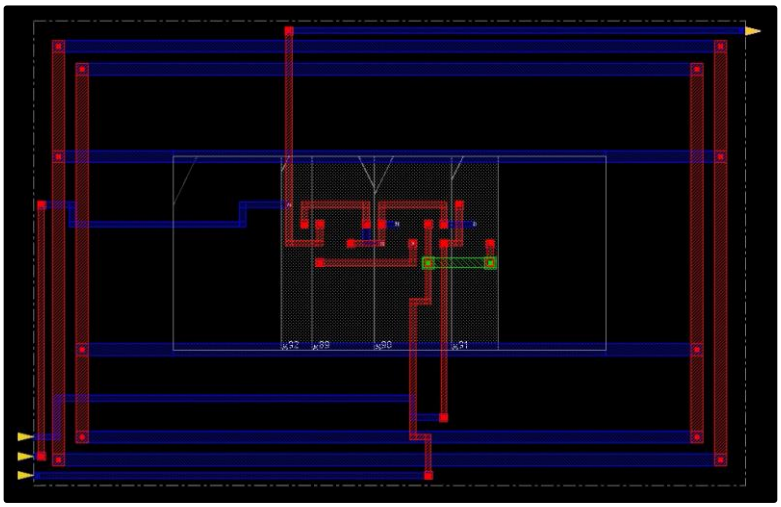


Routing

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

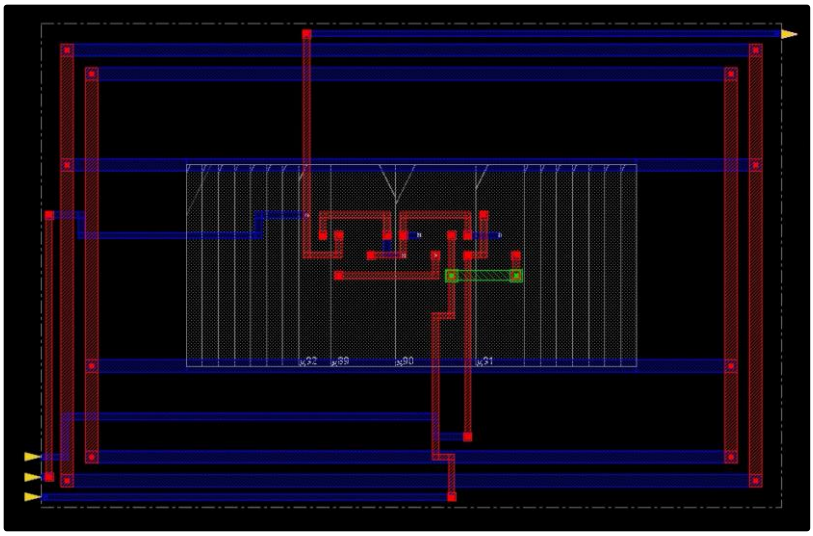


If you see white boxes and spacing errors when you route, your design is too constrained. Consider increasing the offset of your power rings by increments of 0.15.

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 IN THE OSU05 LIBRARY ARE JUST CALLED 'FILL' AND THERE IS ONLY ONE OF THEM.** 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.

This can be done by selecting *Verify Geometry* and *Verify Connectivity* in the *Verify* menu. Reports will be generated and placed in the directory that you started Innovus from. You can also look at the terminal used to start Innovus and see the results of the verification. 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.

You can run these tests from the Innovus command terminal as well. To *Verify Geometry* type "verifyGeometry -report <report file name>" in the Innovus command line. To *Verify Connectivity* type "verifyConnectivity -report <report file name>" in the Innovus command line.

Submit valid pictures of the Innovus Connectivity and Geometry reports from the Terminal as a checkpoint.

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 '*gds2_encounter.map*' file located in the *libdir* 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.

Leave all other fields as default making sure the units are '1000' 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

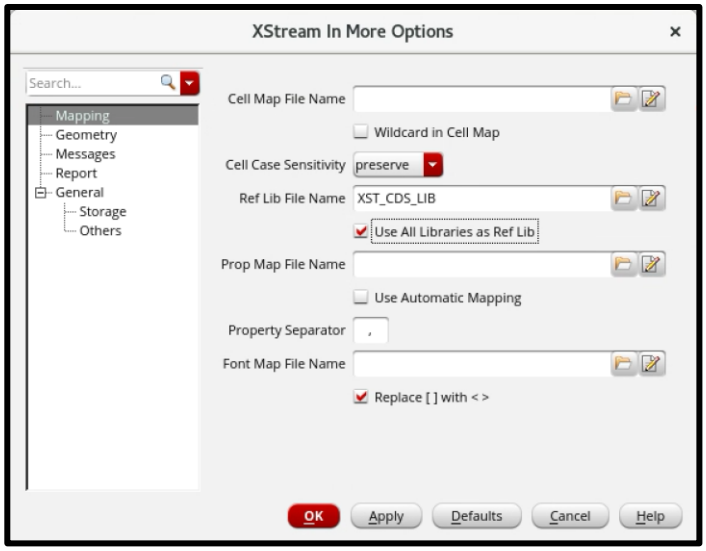
After creating a layout in Innovus and creating the GDS file, open Virtuoso. Click on the CIW (Virtuoso window where errors and messages appear) and click *File* → *Import* → *Stream*. This may produce an error the first time. Try again and it should work and bring up an import window.

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

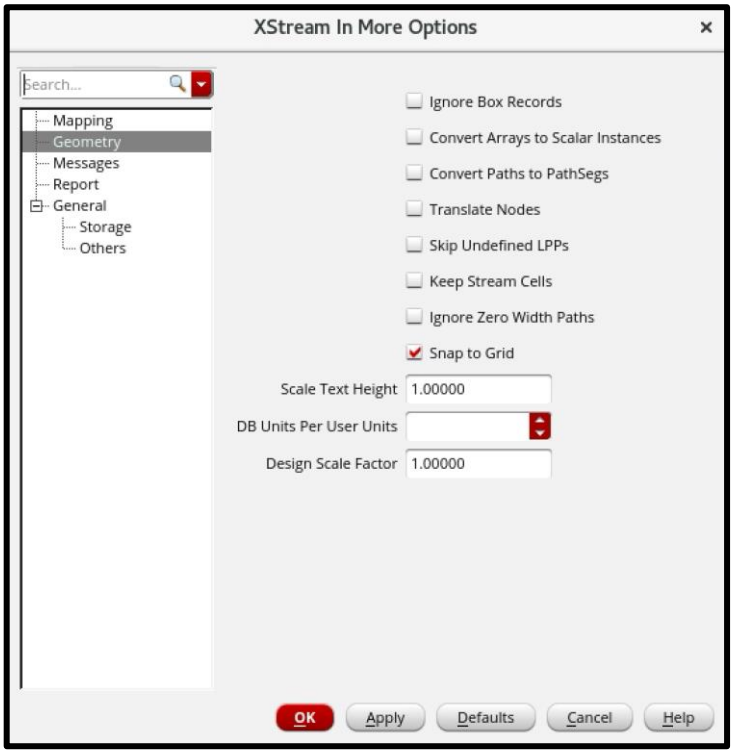
Under library, type the name of the library you would like to save the module in. Note, this does need to be the same as the name of the library given in the stream out in Innovus. Then give a name to the top level cell once it is imported.

Under "Layer Map", select the *gds2_icfb.map* file instead of the defaulted *NCSU_TechLib_ami06.layermap.* This is located in the *lab11_ee330/libdir/* directory.

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



Now, go to the Geometry tab. Select Snap To Grid.



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. Hit *Ctrl+F* to see the layout view.

Possible Errors

Red Cells with no Layers

If the cells appear but are red and disappear when you press Shift+F then there is a problem with the technology file in the folder. You should try copying the layout into a library with a known good technology file. If that does not work delete the library and attempt to stream it in again.

This can be solved through a slightly complicated process.

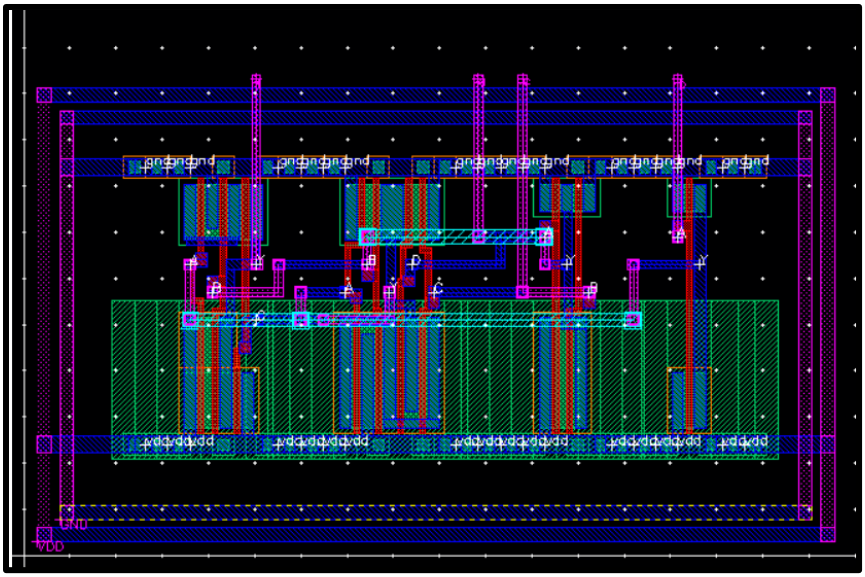
1. Copy the current library. This is just good practice.
2. Delete the original library.
3. Create the library from in Cadence (with the proper name, normally boolean_function)
4. Import the schematic created from RC5.Stream in the file from Innovus, but do not check (or uncheck) the box “Use all libraries as ref. Lib” in the More Options -> Mapping area.
5. Steam in the file again, this time with the “Use all libraries as ref. Lib” checked. This should give 0 errors and 7 warnings
6. Check the Layout and see if it worked.

Step 6: LVS & DRC Checks

DRC and LVS error corrections

At this point, the synthesized schematic and layout from Innovus have been imported into Virtuoso. **Run a DRC and LVS check.** Both will fail, but it is critical that you run the checks before continuing.

The DRC will not pass due to “dubiousData” errors, the Innovus stream is creating metal layer text that should have been text layer. These can be deleted, but before doing so you should add pins. Innovus is adding the name of what the pin should be where the pin should be placed, but does not actually create the pins. We advise creating the pins with the label name then deleting the label. VDD here is the global VDD which is named “vdd!” and ground is the global ground which is named “gnd!”



Submit a valid DRC and LVS for this 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 RTL Compiler 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.