EE 330 Laboratory 5
From Boolean Equation to Silicon

Fall 2022

Contents

Background Information: 2

Checkpoints 3

Part 1: 3 - Input NAND or NOR Gate 3

Part 2: Exchanging gates 3

Part 3: Boolean Function 4

Design Constraints 5

Part 4: Debugging 5

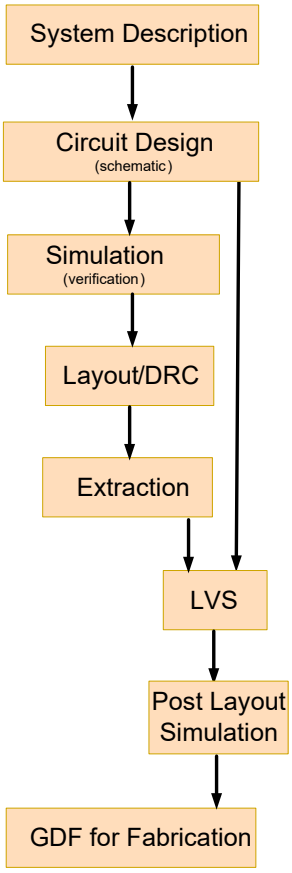
Appendix 7

Generating From Source 7

Background Information:

The objective of this experiment is to experience the design flow of a digital system from the system description through layout where more than one designer is a key contributor to the design. This will be approached by using a simple Boolean system and two designers.

In this experiment, one designer will create a 3-input NAND gate and a second designer will create a 3-input NOR gate. Both should create an inverter. By sharing the gate you create with a partner, you will have all three gates. As part of the prelab for this experiment you will select a Boolean function that can be implemented (without Boolean simplification) with 3-input NAND gates, 3-input NOR gates, and Inverters. You will then implement the Boolean function you selected in silicon (the layout) given area and pin constraints. You can re-use an inverter designed previously or design an inverter for use in this experiment.



Of particular importance when sharing these gates is that they are compatible with the overall floorplan you have for your layout. Horizontal rails for power (VDD) and ground are widely used in the layout of digital circuits with the logic blocks themselves placed between these rails. At a minimum, the floorplan of both designers should have the same distance between the VDD and ground rails.

A simplified Custom IC design flow for an analog circuit and small digital circuits is shown in the flow chart. From the flow chart, it can be observed that we have now studied most of the basic skills needed to do a complete design of a simple circuit. In this experiment, we will take the system description of a system in the form of a simple Boolean expression and convert it to a layout for fabrication through MOSIS. Before you start the design, consider your layout technique, specifically the floorplan. This will allow you to save considerable time in later parts of the experiment.

When considering the floorplan you will be using to implement the Boolean function in Part 2.3, you will find it useful to **layout your gates so that it is easy to interconnect them** in the layout of the Boolean function. For example, in addition to having a **VDD rail (BUS) on the top** of your layout and a **VSS (Ground) rail on the bottom**, it might be convenient to place all **inputs on the left** and the **output to the right** in your cells. With this approach, layout of the Boolean function could involve running signal paths horizontally in the layout. If the layout becomes to long horizontally, pairs of VDD and VSS rails (still equally spaced) can be stacked vertically.

By having you and your partner agree on the floorplan, most importantly the spacing between VDD and VSS busses, the interconnection of the NAND gates, NOR gates, and inverters will be simplified in Part 3 of this experiment.

Checkpoints

The checkpoints for this lab are as follows:

1. NAND/NOR DRC and LVS from Part 1
2. Exchanged NAND/NOR gate
3. Boolean Function Gate-Level Schematic
4. Boolean Function schematic TB
5. Boolean Function DRC and LVS
6. Debugged Schematic, ADE, and working DRC and LVS

Remember, all checkpoints must be shown to a lab TA before the report is submitted. You should include these checkpoints in your lab report.

Part 1: 3 - Input NAND or NOR Gate

As discussed in the prelab, create an Inverter along with a NAND gate or a NOR gate as agreed upon between you and your partner. **Use pcells** (the instances from Lab 3) for the transistors when creating the layout. In a previous experiment you created a schematic and test bench, so following the same approach, create a **layout** of the gates you designed. Be sure your layout passes both DRC and LVS.

When creating the gates for this experiment, use minimum allowable widths and lengths for the NMOS and PMOS devices. Though not of emphasis in this experiment, sizing of devices that are larger than minimum are often preferred in many applications. It will be easiest to achieve such sizing requirements if pcells with multiplicity and fingers are used.

Part 2: Exchanging gates

Exchange gates using the following steps. Though these steps are described for exchanging the cell named **"inverter"**, they can be used for exchanging other cells as well. You will need to

```
# Navigate to your lablib folder:
cd ~/ee330/lablib
# Modify the permissions of the files you want to transfer
chmod 777 -R ~/ee330/lablib
# Create an archive of the cell
tar cvf ~/inverter.tar inverter
```

"pack" the **inverter** cell in the **lablib** library in a single file. Close Virtuoso, open a new terminal window, and from command line run the following:

The file **inverter.tar** in the home directory now contains all the views required to use the **inverter** cell along with read, write and execute (rwx) access for others to use **inverter** and all subdirectories.

To send this file to another person, simply send it as an attachment to an email message, or a shared object in Google Drive. The recipient should then download it into their home directory. The following steps describe how the recipient can then use of **inverter** in their design. To keep the imported designs separate, create a new library called **exchange** (or some other name) and attach it to the AMI06 process. Then run the following:

```
# Move the inverter.tar from home to exchange:
mv ~/inverter.tar ~/ee330/exchange
# Move to the exchange directory
cd ~/ee330/exchange
# Extract the design
tar xvf inverter.tar
```

The inverter cell will now appear in the **exchange** library. Instantiate (create an instance of **inverter** in your schematic using the "i" hot key) the symbol and the **layout** views from this cell to use **inverter** in your own design.

Part 3: Boolean Function

Implement the Boolean function from the prelab by going through the complete design flow (schematic, symbol, test bench, layout, LVS). In this implementation, adhere to the constraints given below. Assign an appropriate name for the cell you create that implements the Boolean function. In this implementation, use the Inverter you created in the previous lab, your NAND or NOR gate from Part 1, and your partners NAND or NOR gate from their Part 1.

By re-using cells that were created, considerable time and effort should be saved since you should not need to place any individual transistors when implementing the Boolean function. The test bench should verify proper output values for all 8 input combinations of A, B, C.

Do not **simplify the Boolean expression** you have chosen. If you follow the floorplanning suggestion of placing VDD on the top, VSS on the bottom, inputs on the left, and outputs on the right, the layout of your logic circuit comprising the Inverter, NAND, and NOR gates, your layout may have gate placements that look something like that shown in Fig. 1 (depicted for 2-input rather than 3-input gates). Of course, the actual layout will be with transistors, not symbols for the gates.

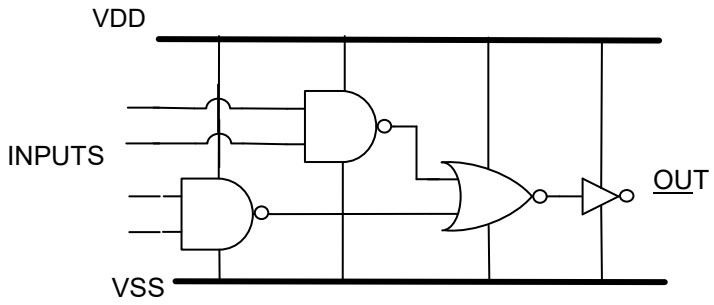


Figure 1: Cell input, output, signal flow, and power directions

Design Constraints

- You may only use the NAND, NOR and inverters you previously created.
- The Boolean function you selected should not be simplified
- The completed cell should fit inside a square of dimensions 40µm×40µm. (Use a ruler)
- The inputs (A, B, C,) for the complete cell must be available at the left boundary of the cell and enter the cell with M1.
- The output Y of the completed cell must be available at the right boundary of the cell and emerge with M1.
- Use a 3µ wide M1 bus at the top of the complete cell for VDD and a 3µ wide M1 bus at the bottom of the complete cell for VSS. Use these rails to supply power to your gates. You may use different widths for the power busses internal to your cell or internal to individual gates.
- You may not use M3 or higher level metals anywhere.

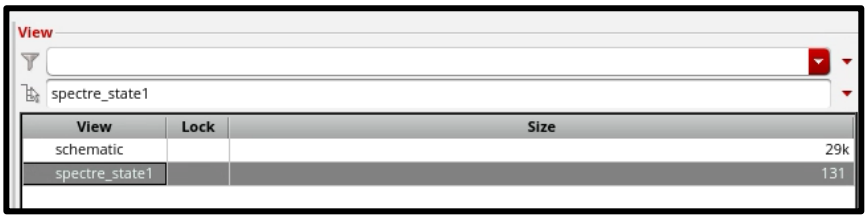
Part 4: Debugging

Cadence Virtuoso has a number of powerful debugging tools available, whether that be detailed output logs or ways of obtaining information from different parts of a circuit. In the final portion of this lab, you will get practice with these tools and with debugging in Virtuoso. To begin, download the "DebugMe" and "DebugMe_TB" compressed files from the course webpage and add them to the "exchange" library that you created earlier in this lab. This will create two new cell views, aptly named "DebugMe" and "DebugMe_TB" which together contain the schematic, symbol, layout, testbench, and ADE state for the following Boolean function:

$$F = \bar{A} + B$$

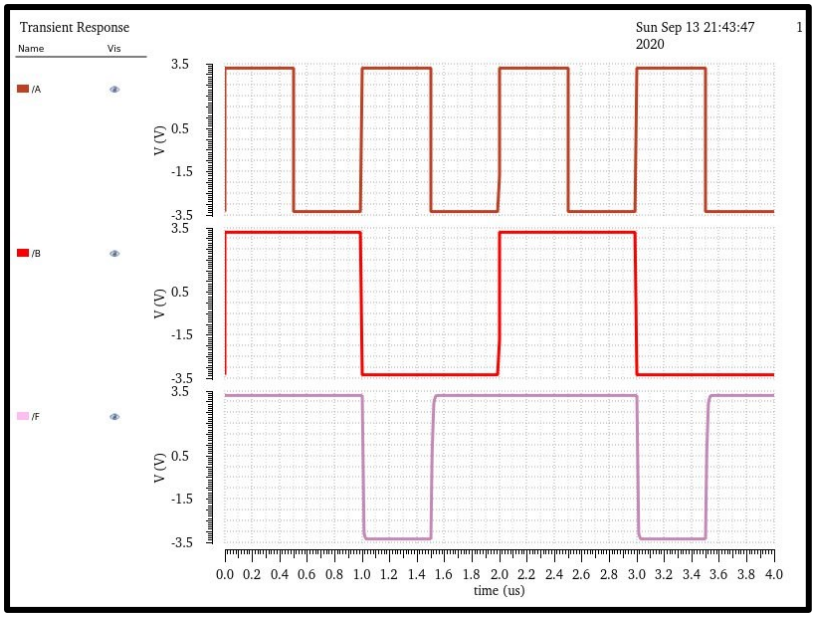
A number of errors have been purposefully made in the provided files. **The only files which do not have errors in them are the Boolean function schematic (the "schematic" view located in the "DebugMe" cell) and the Boolean function symbol (the "symbol" view located in the "DebugMe" cell). Do not change these cell views**. Your task in the final part of this lab is to use the debugging tools available to you to correct the existing errors. For your checkpoint, you

must show a testbench which proves the circuit works correctly as well as valid DRC and LVS results. You may not create a new testbench or ADE instance; use the provided items only (that is, do not create any new views). To open the pre-made ADE instance, click on the "spectre_state1", pictured below.



In your lab report, in addition to showing your checkpoints, you must show screenshots of your layout, of your ADE setup screen, and of the testbench circuit. You must also state what changes you made in each.

TAs will provide limited help on this portion of the lab, although if you believe something may have gone seriously wrong or you are really stuck, you may ask and see what help will be provided. To aid you in your debugging efforts, the testbench simulation results that you should expect to receive are provided below. To receive full credit, your results should match the provided plot.

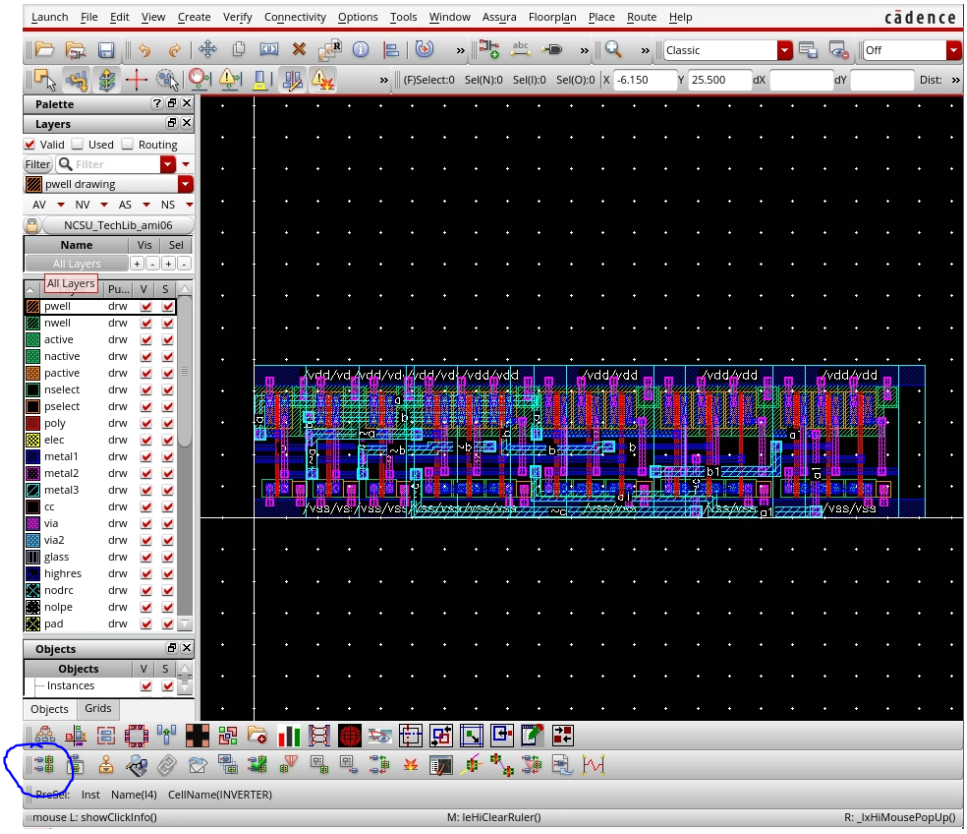


Appendix

Generating From Source

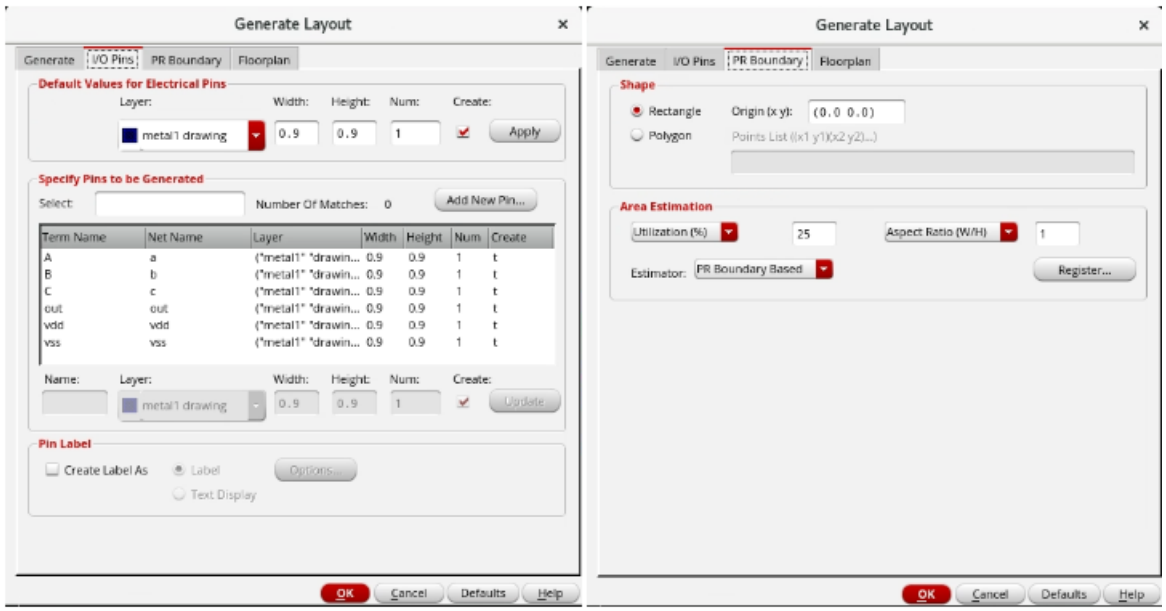
Sometimes, when doing layout, it is more convenient to generate components rather than instantiating them. Luckily, Cadence is powerful software, and allows us to do this. However, use of these tools may not be intuitive or user friendly. So, this guide in the appendix is meant to explain how to generate components in a layout and reduce the amount of work we do in cell placement and routing.

To generate our layout, first we will need a fully functional schematic; you'll probably want to have something that you know works, has been verified by a testbench, and only contains components that have layouts or standard cells. Now, launch Layout XL within Cadence like we would for any other layout. From here, navigate to the bottom left of the window, and press the "Generate All From Source" button, as seen below. This feature is also available under the "Connectivity" dropdown menu, under the "Generate" tab, as the "All From Source" button.

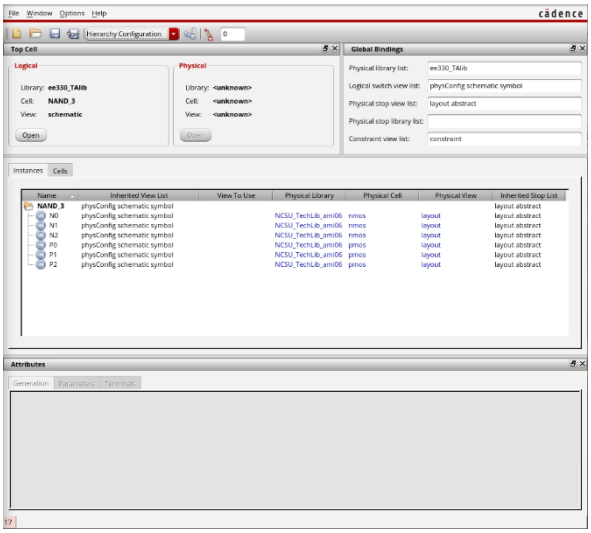


Pressing the above button should open a menu. If you already have objects placed in your layout, it will ask if you want to proceed. If you don't care about what you've already placed, go ahead and hit "yes". If not, make a backup copy before you proceed.

There are plenty of tools available to us in the generation menu. There are two tabs that we will want to use every time we generate a layout: the I/O Pins tab and the PR Boundary tab. The I/O pins tab will allow you to specify the size, quantity, and layer that each pin will be placed on. The default configuration is usually good enough, and pins can be edited after they are generated. However, if you already have a plan in mind, this is where you can change pins before they are made. The PR Boundary pin will allow you to specify a boundary region for your layout; this is useful if you have a certain space constraint you must meet with your layout. You can specify shapes as well as a general shape and area utilization percentage.



Once your generation configuration is completed, press the ”OK” button to generate your layout. This will place pins and standard cells in your layout as well as creating a physConfig cell view. If you only see your pins in your layout and not cells, the physConfig cell view is what you will want to change. This cell view contains all the information that the Generate from Source tool uses to place cell layout information within your layout. For example, if you wanted to generate a PMOS transistor, you would select the library that contains the process you want to use for Physical Library, and then specify PMOS under Physical Cell. This also works if you’ve already created a layout for a cell; select your lab library under Physical Library, and pick the cell you’re interested in under Physical Cell. Once you’ve successfully selected everything, save the file and rerun the Generate from Source tool. You should see the cells you are trying to if they did not generate previously.



Assuming you've done everything correctly, you should see every cell and pin you need for your layout. In addition, every terminal within your layout should now be labeled as to what net it corresponds to within your layout. The Generate from Source tool will also pull in the netlist from the corresponding schematic, giving the connections that will need to be made within the layout. This tool will prove valuable when it comes time to run the LVS tool; if you already know what connections need to be made, it is less likely that you will miss one and fail LVS check. Now that you have all of your cells and pins generated, you can place and route to your heart's content.