Olouwole Eteka

EE 330 Lab 10

Design and Simulation of Digital Circuits using Hardware Description Languages

Introduction

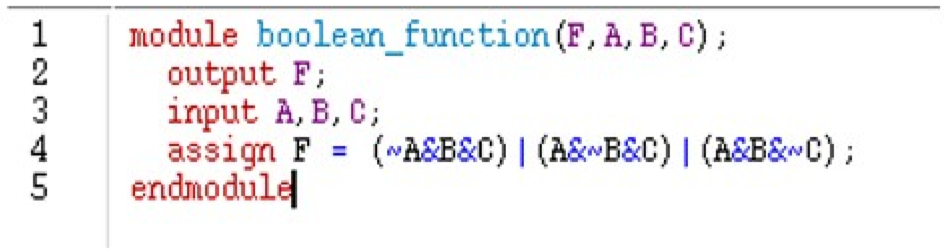
In this lab we will be learn how to implement Hardware Description Language. we will be using Verilog code to design a digital circuit. We will be using also ModelSim for testing out our test bench. We will also be converting codes to schematic and to layout using RTL compiler and Encounter and then make the layout out of it.

Part 1: Inverter

This part of the lab will touch some key point of the features from ModelSim we can use to manipulate a digital Boolean equation and creating test bench to test if it's working correctly. The code for my design is provided bellow.

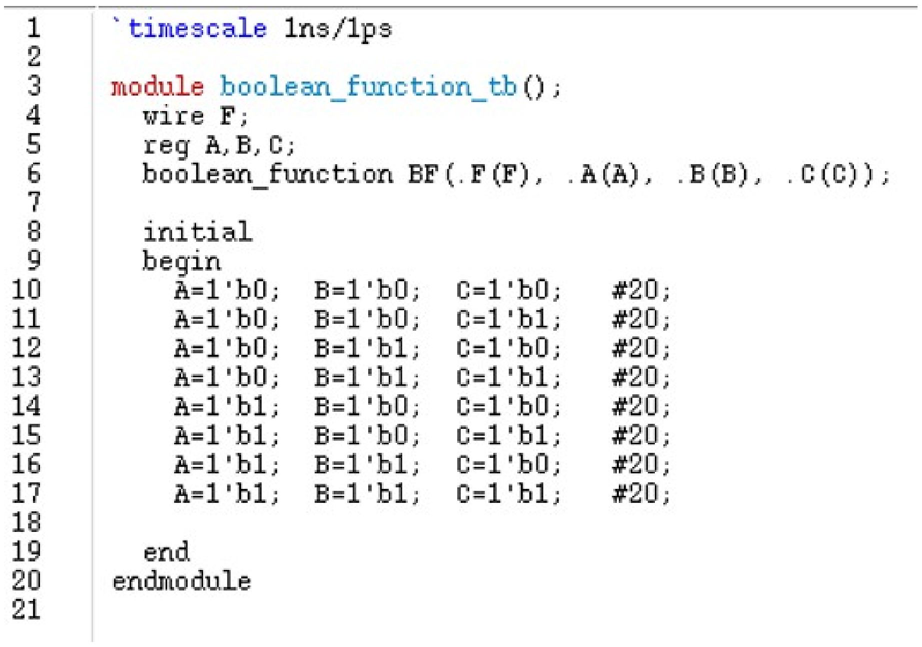
Code

The code for the Boolean function to implement.



```
module boolean_function(F,A,B,C);
  output F;
  input A,B,C;
  assign F = (~A&B&C)|(A&~B&C)|(A&B&~C);
endmodule
```

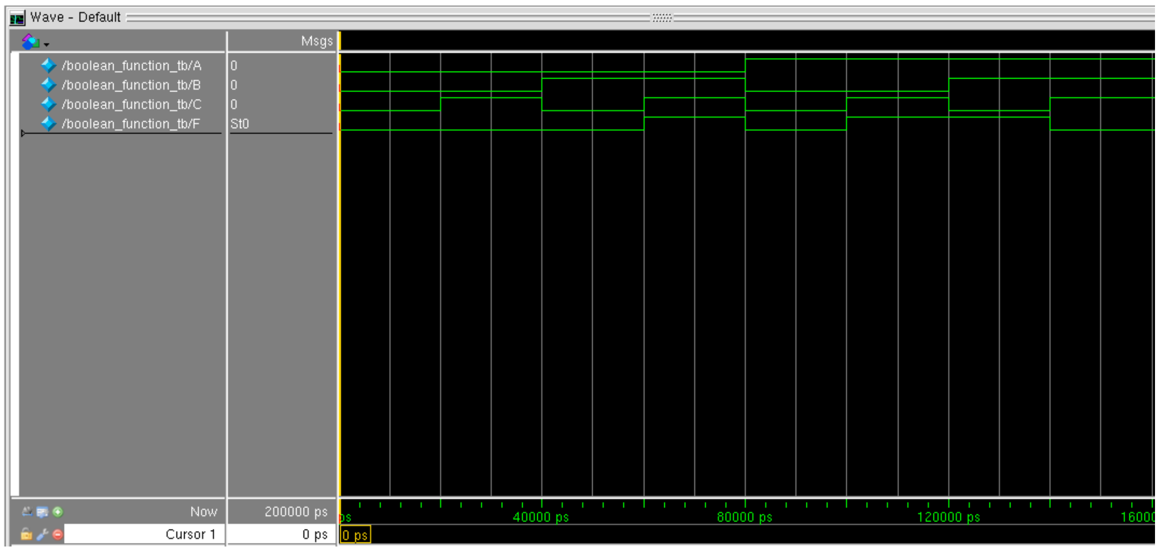
Followed by the test bench for it

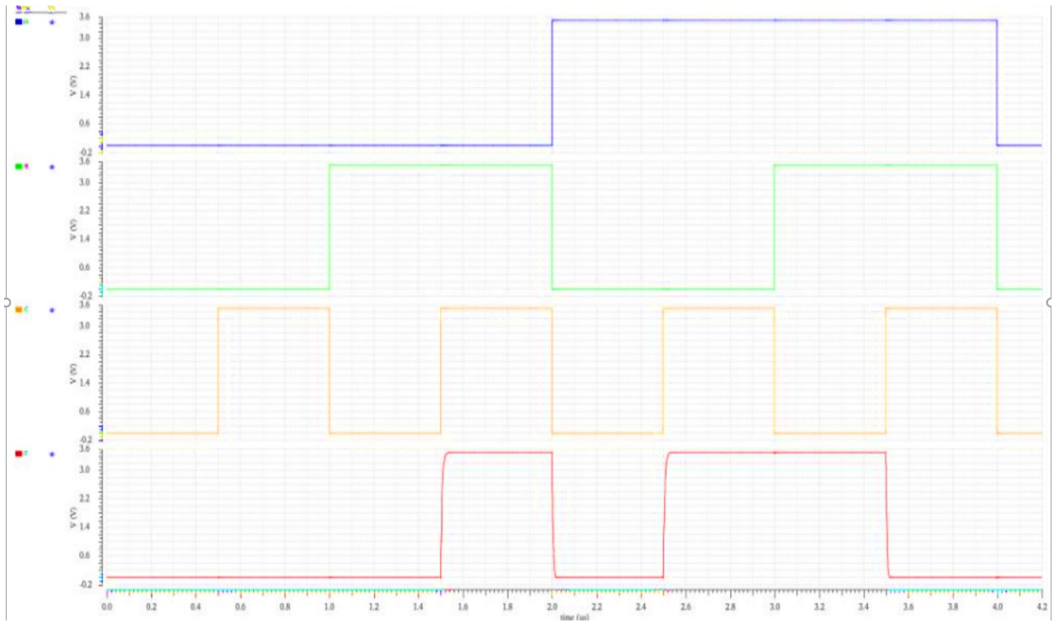


```
`timescale 1ns/1ps

module boolean_function_tb();
  wire F;
  reg A,B,C;
  boolean_function BF(.F(F), .A(A), .B(B), .C(C));

  initial
  begin
    A=1'b0;  B=1'b0;  C=1'b0;   #20;
    A=1'b0;  B=1'b0;  C=1'b1;   #20;
    A=1'b0;  B=1'b1;  C=1'b0;   #20;
    A=1'b0;  B=1'b1;  C=1'b1;   #20;
    A=1'b1;  B=1'b0;  C=1'b0;   #20;
    A=1'b1;  B=1'b0;  C=1'b1;   #20;
    A=1'b1;  B=1'b1;  C=1'b0;   #20;
    A=1'b1;  B=1'b1;  C=1'b1;   #20;

  end
endmodule
```

Stimulation

From this graph bellow we can conclude the code does it job and the graph is the correct as expected and matches the wave form from the same process in lab 4.

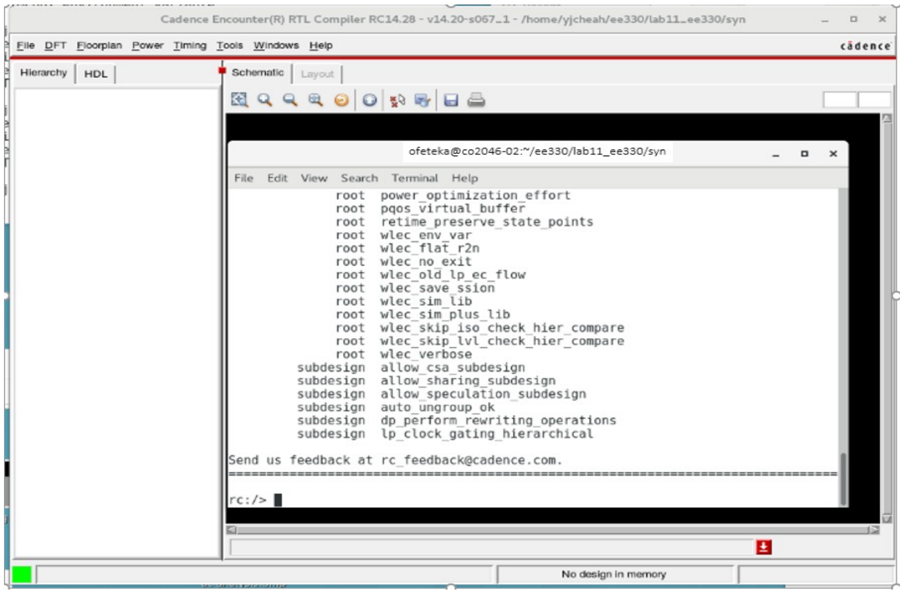




Part 2: Verilog Synthesis with RTL Compiler

In this part we will be using RTL compiler to synthesize our Verilog code. W can do so by following steps I listed below as specify by the lab.

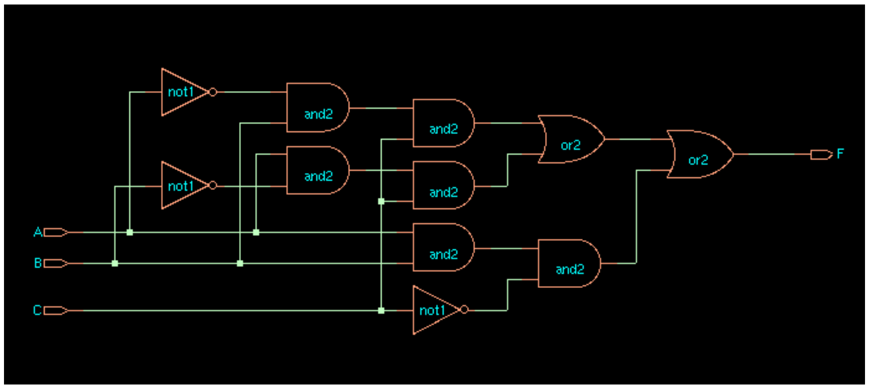
1. Setup standard cell library
2. Setup RTL compiler
3. Copy over the "boolean_function.v" Verilog file created from ModelSim and move into the directory "lab11_ee330/rtl/":
   `cp ~/ee330/boolean_function.v ~/ee330/lab_ee330/rtl`
4. Source the "ic" and "rc" software program, direct terminal to "lab11_ee330/syn/" and open
5. A GUI window should appear and terminal will change to the RC command line:


Cadence Encounter(R) RTL Compiler RC14.28 - v14.20-s067_1 - /home/yjcheah/ee330/lab11_ee330/syn
File DFT Floorplan Power Timing Tools Windows Help
cadence
Hierarchy HDL
Schematic Layout
ofeteka@co2046-02:~/ee330/lab11_ee330/syn
File Edit View Search Terminal Help
No design in memory

```
               root   power_optimization_effort
               root   pqos_virtual_buffer
               root   retime_preserve_state_points
               root   wlec_env_var
               root   wlec_flat_r2n
               root   wlec_no_exit
               root   wlec_old_lp_ec_flow
               root   wlec_save_ssion
               root   wlec_sim_lib
               root   wlec_sim_plus_lib
               root   wlec_skip_iso_check_hier_compare
               root   wlec_skip_lvl_check_hier_compare
               root   wlec_verbose
          subdesign   allow_csa_subdesign
          subdesign   allow_sharing_subdesign
          subdesign   allow_speculation_subdesign
          subdesign   auto_ungroup_ok
          subdesign   dp_perform_rewriting_operations
          subdesign   lp_clock_gating_hierarchical

Send us feedback at rc_feedback@cadence.com.
================================================================================

rc:/> 
```

6. In the RC command window, load the correct standard cell libraries and run the synthesis script (Ignore the warnings):
7. In the RC command window, load and elaborate “boolean_function.v”
8. RTL Compiler should have generated an unoptimized version of “boolean_function.v” and should show up in the GUI:

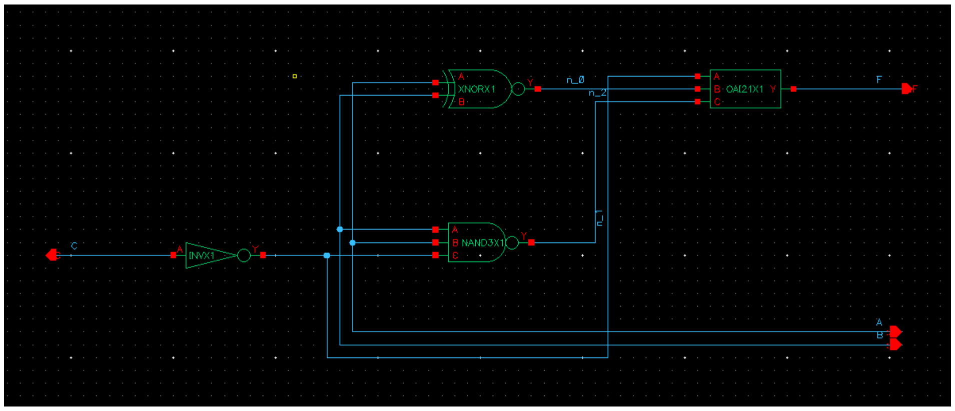
By doing the number 8 of this process I got the graph bollow :



9. Synthesize the design and export it:
   We ca n do so by using the code provided in the lab: “`synthesize -to_mapped -eff high -no_incr`
   `write -mapped > ${DESIGN}_synth.v`
   `write_sdc > ${DESIGN}.sdc`

”

10. Finally, part 10 can Import the synthesized design schematic to Cadence Virtuoso:
    After this step we should have an imported Schematic design to Cadence Virtuoso



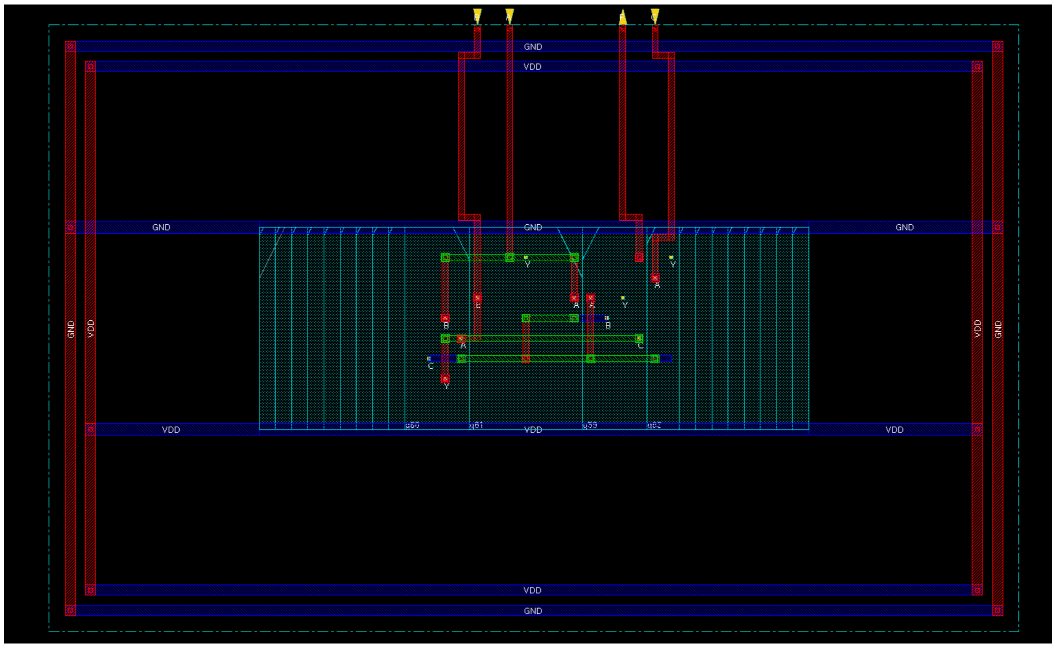
Part 3: Layout of Digital Circuits with Encounter

In this part of the lab, we will be creating a layout from the synthesized schematic from HDL.

To do so we need the follow the steps listed bellow

1. In “lab11_ee330” directory, create a folder named “encounter”.
2. In terminal, we can launch the encounter by the command: `encounter`
3. In the encounter, we go to File -> Import RTL and fill as specify by the lab
4. Go to File -> Import Design and fill in as below:
5. Now we should have an empty “die” appear in the Encounter GUI, then now we are going to add a Floorplan. Go to Floorplan -> Specify Floorplan:
   - i. Aspect ratio (height/width) = 1.0
   - ii. Core Utilization = 0.5
   - iii. Core Margins = 10 for all left, right, top and bottom
6. Then adding power planning. Go to Power -> Power Planning -> Add Rings
   - i. Set the power plan to be around the I/O boundary.
   - ii. Enter “GND VDD’ in the Net(s) field.
   - iii. Verify that the width and spacing comply with the process design rules and are a multiple of lambda.
7. Now, placement is needed. Go to Place -> Standard Cells -> Click OK
8. Adding routing.
   - i. Go to Route -> Special Route:
     Unselect Block pins, Pad pins, and Pad rings then click OK
   - ii. Go to Route -> NanoRoute -> Route -> Click OK
9. Adding filler cells. Go to Place -> Physical Cell -> Add Filler.
   - i. In Cell Name(s) field, click “Select” and select the only on filler cell in OSU018 library that named “FILL”.
   - ii. Click OK.

10. Now we can verify for errors:
    i. Verify -> Verify Geometry, Verify -> Verify Connectivity
    ii. The error reports can be seen from terminal
11. Save the design file by going to File -> Save Design (Save in encounter format).
12. GDS Export of Layout from Encounter. Go to File -> Save -> GDS/OASIS… Then fill it as specify by the lab
    After going to the process we imported the layout as showing bellow.

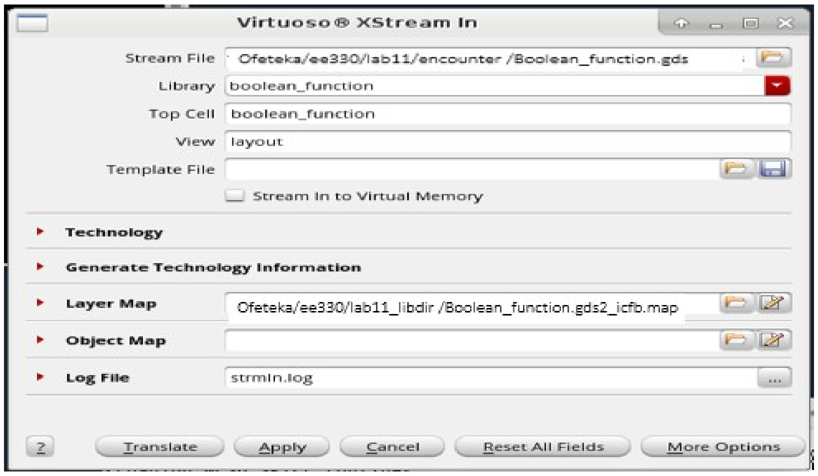


Part 4 Import the layout into Cadence

This part of the lab consists of importing the layout from Encounter into Cadence Virtuoso, then test them up by running DRC, LVS tests to compare between the schematic imported from RTL Compiler and layout imported from encounter, they should match.

We can do this by following the steps below.

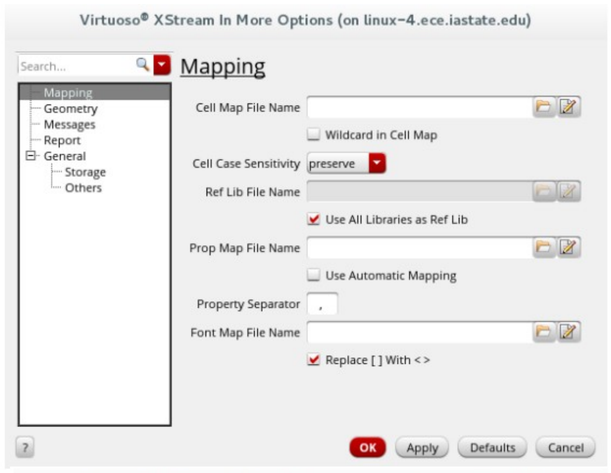
1. GDS import of layout into Virtuoso. On the CIW (Virtuoso window where errors and messages appear)
    i. Go to File -> Import -> Stream, enter the following:


Virtuoso® XStream In

Stream File: Ofeteka/ee330/lab11/encounter /Boolean_function.gds
Library: boolean_function
Top Cell: boolean_function
View: layout
Template File:
☐ Stream In to Virtual Memory

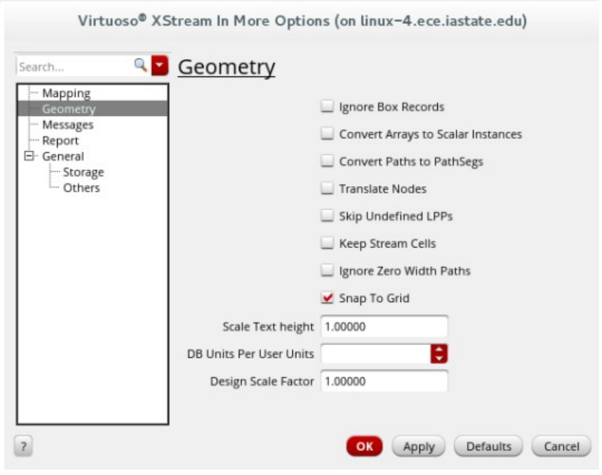
Technology
Generate Technology Information
Layer Map: Ofeteka/ee330/lab11_libdir /Boolean_function.gds2_icfb.map
Object Map:
Log File: strmIn.log

ii. In the same window, click More Options and check as follow:


Virtuoso® XStream In More Options (on linux-4.ece.iastate.edu)

Mapping

Cell Map File Name:
☐ Wildcard in Cell Map
Cell Case Sensitivity: preserve
Ref Lib File Name:
✓ Use All Libraries as Ref Lib
Prop Map File Name:
☐ Use Automatic Mapping
Property Separator: ,
Font Map File Name:
✓ Replace [ ] With < >

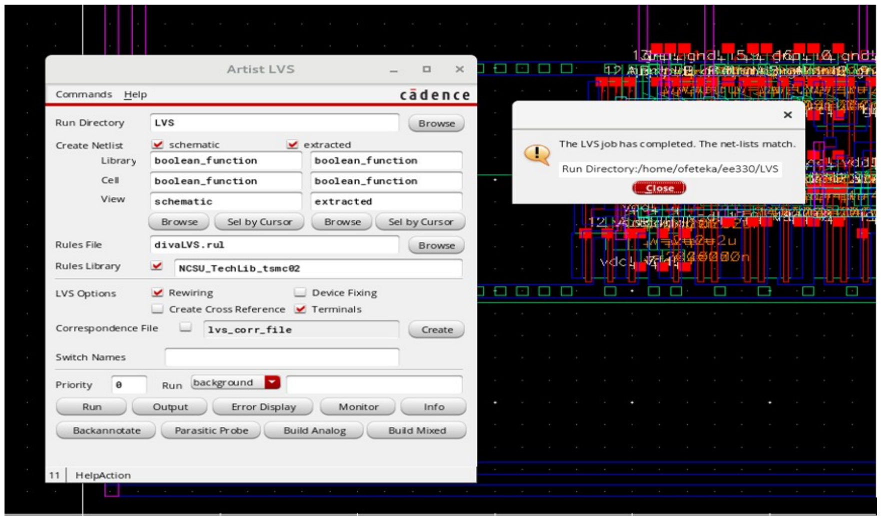

Virtuoso® XStream In More Options (on linux-4.ece.iastate.edu)

Geometry

☐ Ignore Box Records
☐ Convert Arrays to Scalar Instances
☐ Convert Paths to PathSegs
☐ Translate Nodes
☐ Skip Undefined LPPs
☐ Keep Stream Cells
☐ Ignore Zero Width Paths
✓ Snap To Grid
Scale Text height: 1.00000
DB Units Per User Units:
Design Scale Factor: 1.00000

iii. Click OK and then in the pop-up, Click Translate (We should receive 0 errors with little or no warnings).

2. Now we have the layout successfully imported to Cadence Virtuoso, but since the DRC will not pass due to “dubious Data” errors that happen because Encounter create the pin names as metal layer text instead of just text layer. Therefore, we should follow the following steps:
   i. Add Metal 2 pins at their respective places
      (a) Input: “A”, “B”, “c”
      (b) Output: “F”
      (c) Input/output: “vdd!”, “gnd!” (Note that we didn’t use VDD and GND because in the schematic that we imported from RTL Compiler, it uses the names “vdd!” and “gnd!” for VDD and GND and in order for LVS to works, they have to use the same pin names)
   ii. Delete the metal layer texts “A”, “B”, “C”, “F”, “VDD”, and “GND”
3. Run LVS test and make sure it matches:



Run Directory: LVS

Create Netlist	✓ schematic	✓ extracted
Library	boolean_function	boolean_function
Cell	boolean_function	boolean_function
View	schematic	extracted

Rules File: divaLVS.rul

Rules Library: ✓ NCSU_TechLib_tsmc02

LVS Options: ✓ Rewiring ☐ Device Fixing ☐ Create Cross Reference ✓ Terminals

Correspondence File: ☐ lvs_corr_file

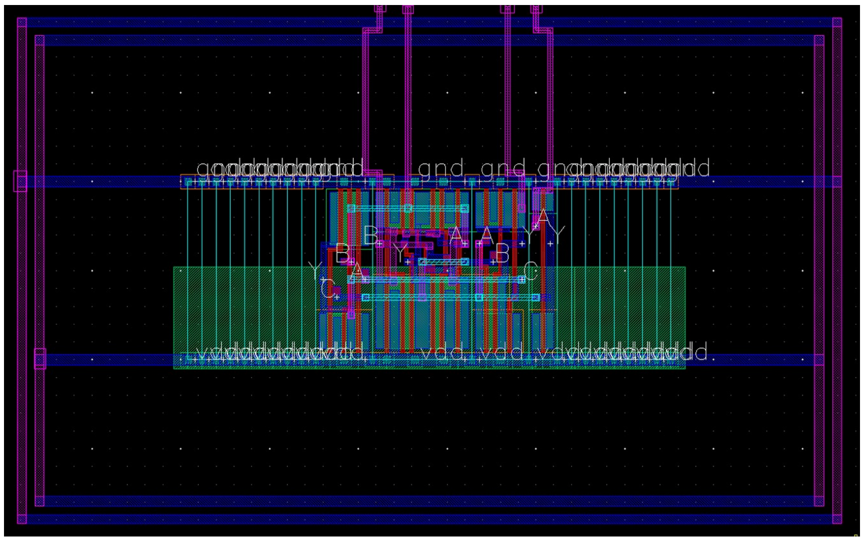
Switch Names:

Priority: 0 Run: background

The LVS job has completed. The net-lists match.

Run Directory:/home/ofeteka/ee330/LVS

The finial stage is looking like the graph below



Conclusion

This lab was useful to learn the process of changing Hardware Description Languages into Layout Design by first creating a HDL Verilog code. I learned how to do a proper testing using a test bench also some tricks to perform a better layout. Importing both schematic and layout designs into Cadence Virtuoso was a good way of comparing make sure the layout is working properly by running DRC and LVS tests.