MODULE

4

SAN JOSE STATE UNIVERSITY

Electrical Engineering Department

# Using IIT's Standard Cell Library to Synthesize Top Down designs authored with Verilog

IC DESIGN GROUP SAN JOSE STATE UNIVERSITY

# A supplemental guide for using CDS tools for IC design

David W. Parent
Assistant Professor
Electrical Engineering, SJSU
One Washington Square
San Jose, CA 95192-0084
Phone 408.924.3963 • Fax 408.924.2925

# Table of Contents

**LIST OF FIGURES:**




## Acknowledgements:

This tutorial is based on the NCSU design kit. For more information, see http://www.ece.ncsu.edu/cadence/CDK.html. This tutorial also follows the design flow used by WPI at http://vlsi.wpi.edu/cadence/.

This would not be possible without the excellent library and documentation developed by James Stine and his Graduate Student *Johannes Grad* at the Illinois institute of Technology (http://www.ece.iit.edu/~cad/cadence/seultra/ ).

Chapter

1

# Standard Cell Design:

## Background:

The standard cell design methodology is a top down approach to digital design. It is also called the ASIC approach and is used in EE287 . Any time you start with a HDL description of the logic and work your way down you are following this kind of approach. The advantages of this approach are that you can describe logic in words without committing to a specific circuit. for instance A<=B+C, would define and adder, but you have not committed to a ripple carry of carry look ahead architecture. You can optimize your adding at a higher level without have to do a \n adder circuit by hand. This approach along with the use of standard cell library which has an estimate of the delay and power for various circuits that are already laid out, can be used to automatically generate layouts from a Verilog or vhdl net-list.

The design flow for this approach is:

1. Describe the functionality of your circuit in an HDL (In this case Verilog)
    a. Keep re-writing code and simulating until the logic meets specification (No timing information is created at this time.)
2. Synthesize your HDL description of your circuit to a library that has timing (delay) information.
    a. Optimize the performance of your design until you meet your logical, timing, and power specification.
3. Place and route your design with area and topology information from a physical library.
4. Create a physical description of your circuit in “GDSII” format
5. Import this physical description into a DFII database (The view your are familiar with if you have done the bottom up cadence tutorial. For example: the layout view.)
6. Run a DRC check on the layout. (draw a do not DRClayer over the pins)
7. Create a Spice netlist from the layout view by extracting the layout.

8. Create a symbol for the design
9. Create a test bench for this design, use the symbol to set up the spice test bench
10. Verify the logic, timing, and power specification for the corners (If you have them, if not use the nominal values)
11. Send the design out
12. Verify the circuit when it comes back from manufacturing.

It is very important to remember that each step in the design flow you are estimating the performance of your circuit. The accuracy of the simulation increases as you go down the design flow, but so does the simulation time. Furthermore, it takes more time to fix an error the farther you are along the design flow. As a rule of thumb it is better to give your design a 30% margin of error at the beginning of the design, so you do not have implement costly fixes at the end of the design flow or god forbid after the circuit has been manufactured. Currently the design cycles in industry are so short that you do not have the luxury of fixing a bug in a later release.

For more on various design flows see chapter 1 of the new Kang and Leblebici book.

## Getting Started:

This tutorial will take you through the basics of taking a design through synthesis and place and route. At this time, it will not cover how to actually optimize a circuit in the higher level tools. It assumes that you know how to optimize a Verilog description of a circuit from a Top-Down design class.

Choose which technolgy you want to use. The available technologies are:

- AMI06
- TSMC025
- TSMC018

Copy the start up files into your account, depending on which technology you selected (Figure 1, Figure 2 and Figure 3.). In this case, we will be doing a 10 bit lfsr. No matter what type of circuit you are doing you need these files.

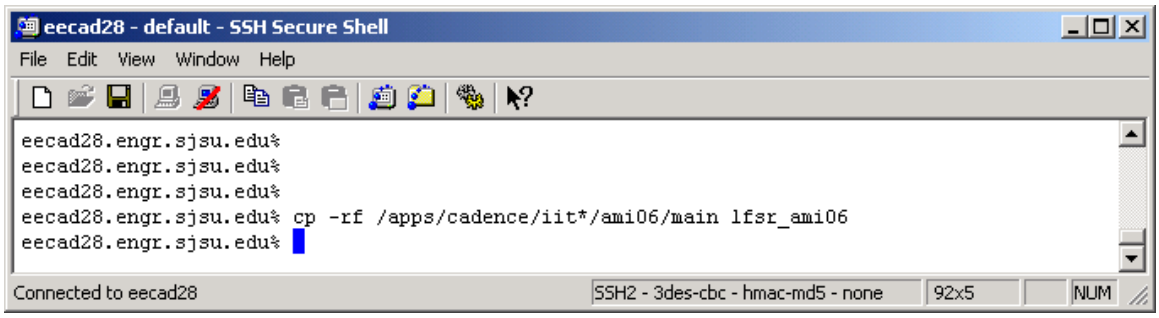


Figure 1: Files to copy if you are using the AMI06 process.

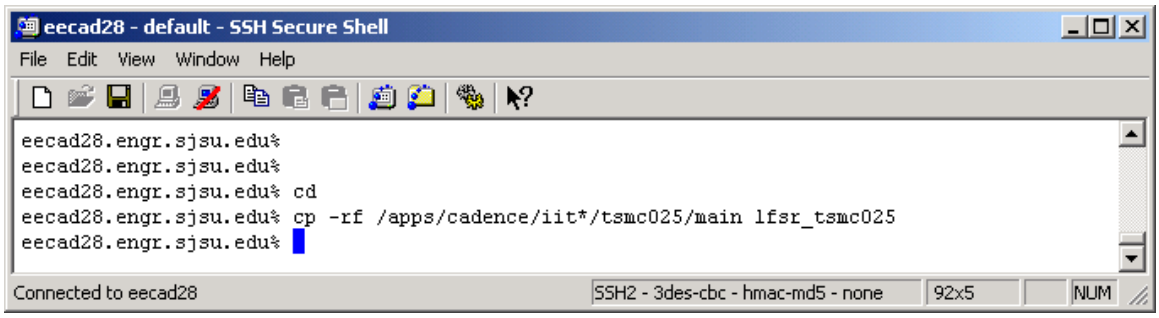


Figure 2: Files to copy if you are using the TSMC025 process

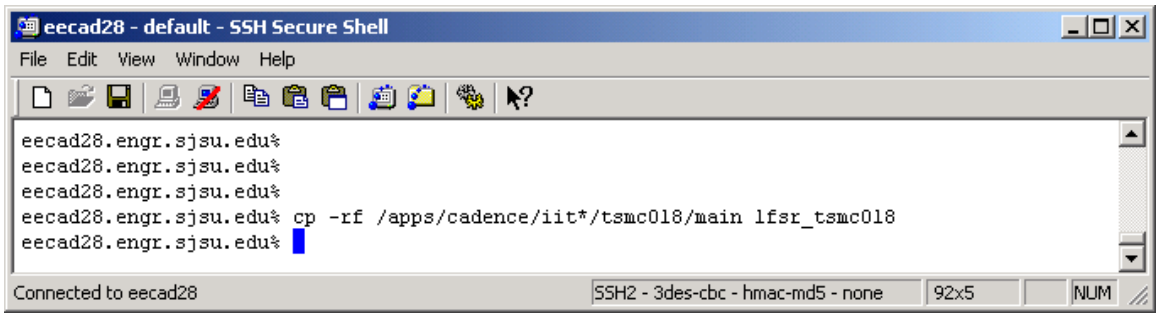


Figure 3: Files to copy if you are using the TSMC018 process

We will be following the AMI06 process for the rest of the tutorial.

Now cd into the directory you just created and copy the actual verilog file into it (Figure 4).

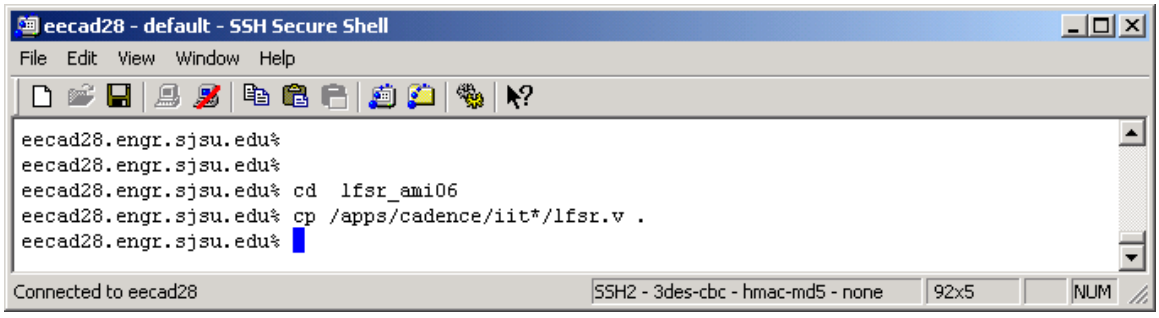


Figure 4: Copying the verilog file into the proper directory.

Use a text editor to make sure the code matches the code in Figure 5.

## Getting ready for Synthesis:

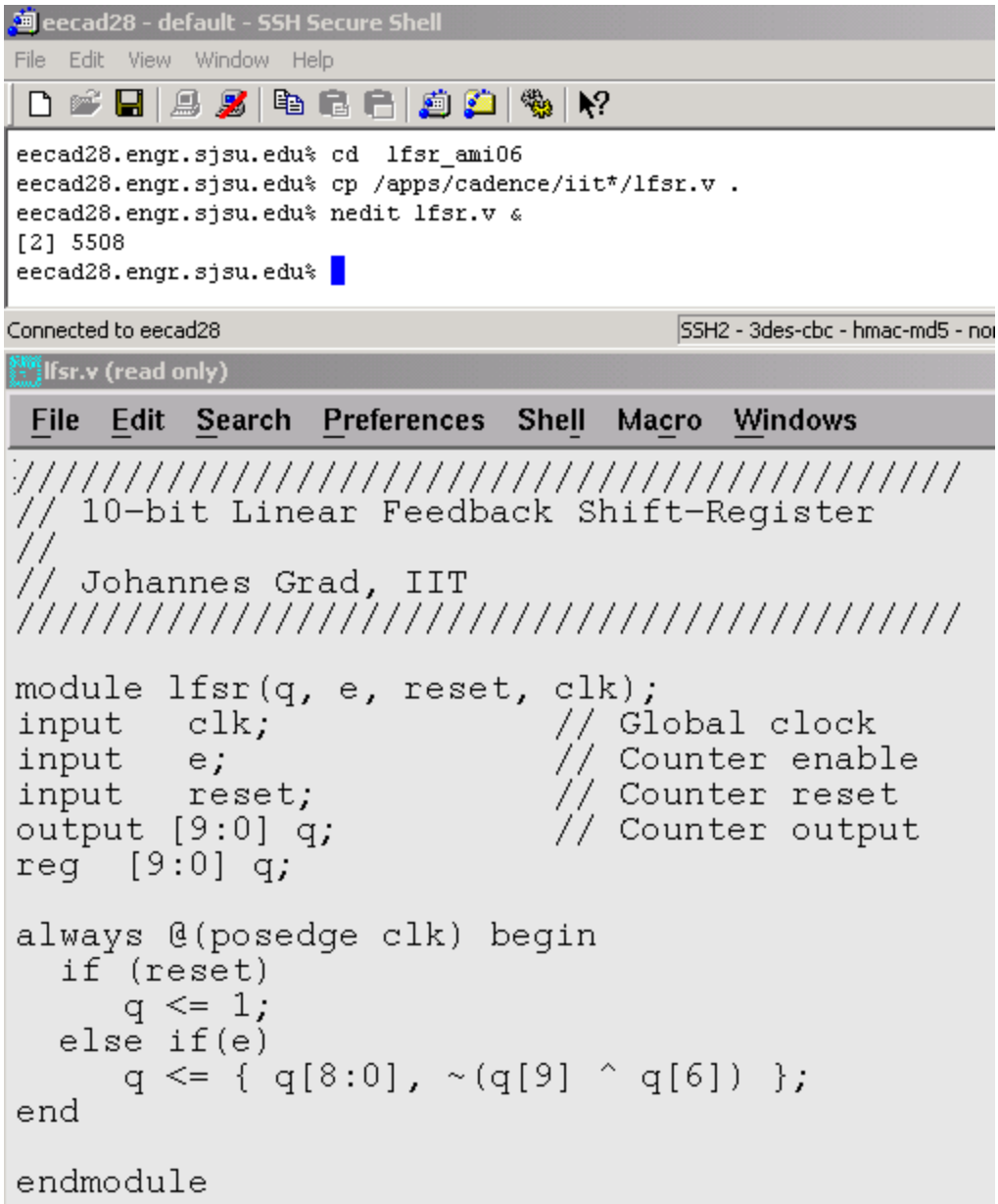


Figure 5: Code for lfsr.

Now we have to make sure you are running the correct version of Synopsis tools.

If your account was created before the 19'th of September 2003, copy a new .cshrc file into your home directory, and then log out and log back in(Figure 6).

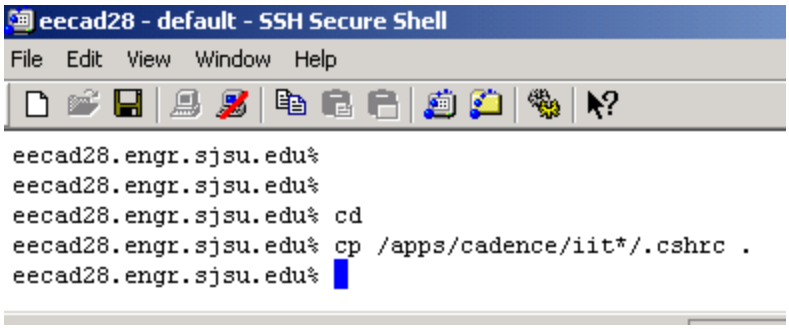


Figure 6: Getting the correct .cshrc file (Remember to log out and log back in.)

Change back into you lfsr_ami06 directory.

Before we synthesyze the lfsr we have to modify the script file that the Synopsis tool uses to compile your design into a net list.

Open the text editor and open the file compile.scr

```
eecad28.engr.sjsu.edu% pwd
/home/dparent/lfsr_ami06
eecad28.engr.sjsu.edu% nedit compile.scr &
[2] 5522
eecad28.engr.sjsu.edu%
```

Figure 7: opening compile.scr.

The script file is slef explanatory. You have have to search and replace TOP_LEVEL_NAME and FILE_NAME with lfsr.

```
/*****************************************************
/* Compile Script for Synopsys                       *
/* Johannes Grad, IIT                                *
/*                                                   *
/*                                                   *
/* Change "TOP_LEVEL_NAME" to your design name       *
/* Change "FILE_NAME" to your verilog sourc          *
/*           code file. If you have more than        *
/*           one, you need one line for each         *
/*           Do not include the testbench            *
/*****************************************************

link_library=target_library={iit06_stdcells_pads.db

define_design_lib WORK -path .

read -f verilog FILE_NAME.v

set_flatten true
verilogout_show_unconnected_pins = "true";
max_area 0.0
current_design TOP_LEVEL_NAME

compile -ungroup_all -map_effort medium

set_max_fanout 8.0 TOP_LEVEL_NAME

compile -incremental_mapping -map_effort medium

check_design

/* If you want pads, uncomment these 2 lines and */
/* list all your pins/busses that you want have as
/* set_port_is_pad {clk,load,Cin,Z,A,B} */
/* insert_pads */

write -f verilog -output TOP_LEVEL_NAME.vh
write -hier -output TOP_LEVEL_NAME.db
```

Figure 8: The compile.scr file before editing.

```
/************************************************/
/* Compile Script for Synopsys                  */
/* Johannes Grad, IIT                           */
/*                                              */
/*                                              */
/* Change "lfsr" to your design name    */
/* Change "lfsr" to your verilog sourc       */
/*           code file. If you have more than   */
/*           one, you need one line for each    */
/*           Do not include the testbench       */
/************************************************/

link_library=target_library={iit06_stdcells_pads.db}

define_design_lib WORK -path .

read -f verilog lfsr.v

set_flatten true
verilogout_show_unconnected_pins = "true";
max_area 0.0
current_design lfsr

compile -ungroup_all -map_effort medium

set_max_fanout 8.0 lfsr

compile -incremental_mapping -map_effort medium

check_design

/* If you want pads, uncomment these 2 lines and */
/* list all your pins/busses that you want have as pads '
/* set_port_is_pad {clk,load,Cin,Z,A,B} */
/* insert_pads */

write -f verilog -output lfsr.vh
write -hier -output lfsr.db
```

Figure 9: The compile.scr file after editing.

Now we actually run the synthesizer from the command line in Synopsys(Figure 10).

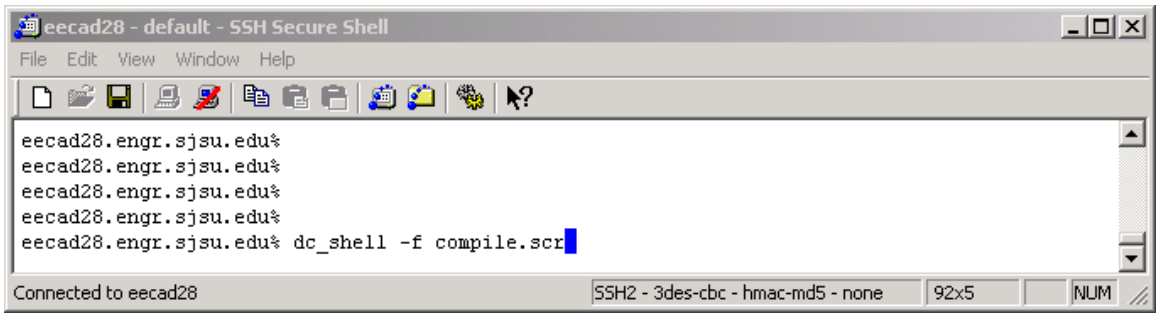


Figure 10: Running Synthesizer.

You should see output like Figure 11.

**At this point you should check the timing of the circuit and eliminate any long paths that make the circuit fail specification. You should be within 30% of your spec at this time to allow for routing induced delays.**

```
  Beginning Delay Optimization Phase
  ----------------------------------

   ELAPSED            WORST NEG TOTAL NEG  DESIGN
    TIME      AREA      SLACK     SLACK   RULE COST
  --------- --------- --------- --------- ---------


  Beginning Phase 1 Design Rule Fixing  (max_fanout)
  ----------------------------------

   ELAPSED            WORST NEG TOTAL NEG  DESIGN
    TIME      AREA      SLACK     SLACK   RULE COST
  --------- --------- --------- --------- ---------
    0:00:01   15192.0      0.00       0.0       2.0
    0:00:01   15408.0      0.00       0.0       1.0
    0:00:01   15624.0      0.00       0.0       0.0


  Beginning Area-Recovery Phase  (max_area 0)
  -----------------------------

   ELAPSED            WORST NEG TOTAL NEG  DESIGN
    TIME      AREA      SLACK     SLACK   RULE COST
  --------- --------- --------- --------- ---------

  Optimization Complete
  ---------------------
  Transferring design 'lfsr' to database 'lfsr.db'
Current design is 'lfsr'.
1

check_design
1

/* If you want pads, uncomment these 2 lines and */
/* list all your pins/busses that you want have as p
/* set_port_is_pad {clk,load,Cin,Z,A,B} */
/* insert_pads */

write -f verilog -output lfsr.vh
Information: Please make sure that you have run the
1
write -hier -output lfsr.db
Writing to file /home/dparent/lfsr_ami06/lfsr.db
1

report_timing > timing.rep
1
report_cell   > cell.rep
1
report_power  > power.rep
1
quit
1
dc_shell>
Thank you...
eecad28.engr.sjsu.edu%
```

Figure 11: Output from synthesis.

## Setting up your DFII Data Base:

Ultimately we are going to load in the gds2 file from the place and route tool into icfb. WE must create a library with icfb before we run the seultra tool script. The reason is that the se_shell script creates a cds.lib file that will not allow you to view the technology file information.

To create this directory: We have to start icfb and create a directory with the proper techfile and same name of the verilog file (**Error! Reference source not found.**).

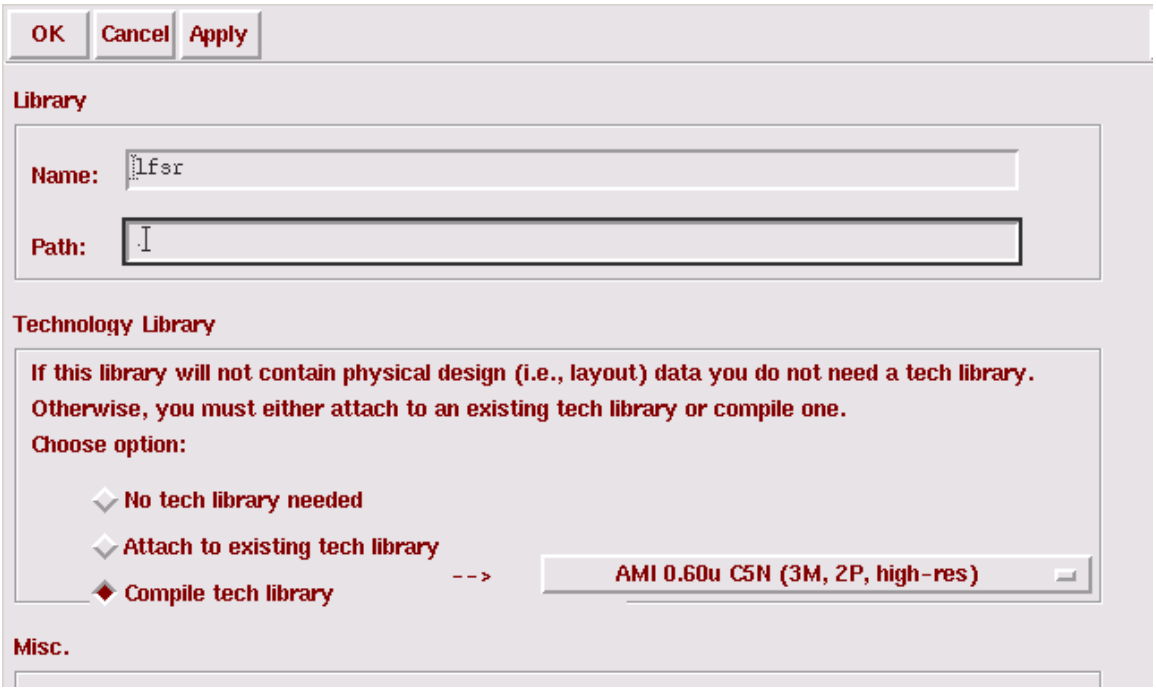


Figure 12: Creating a DFII library.

If it is done correctly, you should see the library manager look like Figure 13.

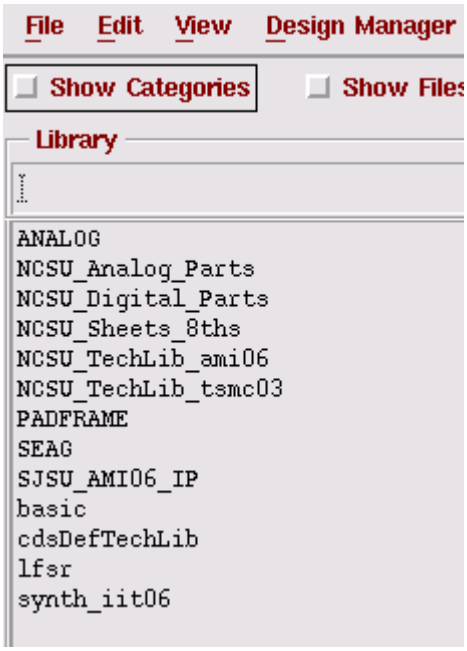


Figure 13: Library Manager after DFII data base is created.

## Place and Route:

Now we need to run the place and route tool from Cadence called seultra. Before we do this, we have to edit the script file seultra.scr according to Figure 14.

```
#
##############################################################################

# read in LEF file
INPUT LEF FILENAME "iit06_stdcells_pads.lef" REPORTFILE "importlef.rpt" ;

# import verilog netlist
SET VAR INPUT.VERILOG.POWER.NET "vdd";
SET VAR INPUT.VERILOG.GROUND.NET "gnd";
SET VAR INPUT.VERILOG.LOGIC.1.NET "vdd";
SET VAR INPUT.VERILOG.LOGIC.0.NET "gnd";
SET VAR INPUT.VERILOG.SPECIAL.NETS "vdd gnd";
INPUT VERILOG FILE "iit06_stdcells_pads.v.se" LIB "synth_iit06" ;

#
# Read here the result of your compile script
# change FILE_NAME to your synthesis output file
# and change TOP_LEVEL_NAME to the name of your toplevel module
# (most likely both will be called the same)
#
INPUT VERILOG FILE "lfsr.vh" LIB "synth_iit06" REFLIB "synth_iit06" DESIGN "synth_iit06.lfsr:hdl"
```

Figure 14: Editing the seultra.scr file.

It is simple; just follow the instructions of the script file.

Now we have to run a set up file that tells the shell where the binaries are for the Cadence Place and rout tool Figure 16).

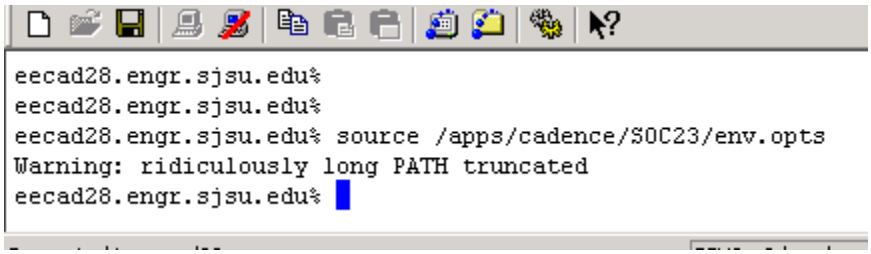


Figure 15: Sourcing the evn.opts file.

Now run the place and route tool from the command line (Figure 16). This will take a long time even for this simple design.

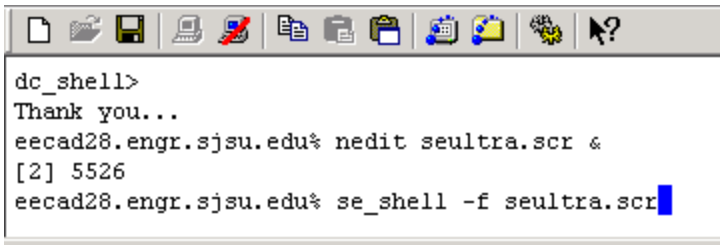


Figure 16: Running the place and rout tool.

When it is finished, your terminal should look like Figure 17

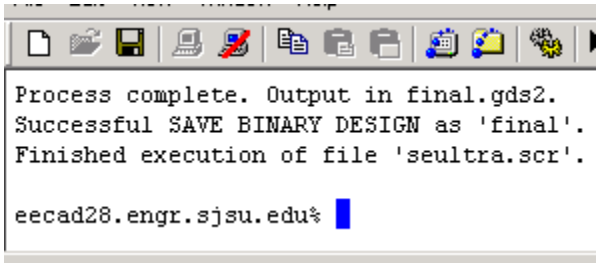


Figure 17: Place and route finished.

## Converting from SEULTRA fomrat to DFII (icfb)

The format of the project is not in the correct DFII database format to convert it we run a special script developed by IIT. To run the script type in the command iitcells_se2icfb, as in Figure 18.

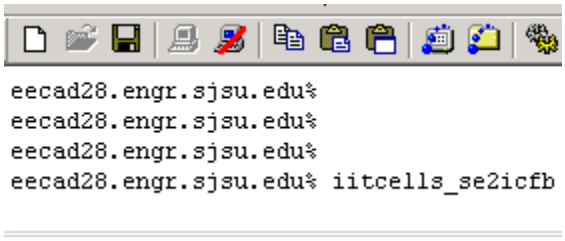


Figure 18: Conversion Script.

You should get a listing as in Figure 19.

```
Checking if final.gds2 exists......OK
Determining top-level name.........OK (lfsr)
Creating temporary cds.lib.........OK
Determining Technology.............OK (AMI 0.5um)
Removing old library...............OK (lfsr)
Creating new DFII library..........OK (lfsr)
Creating PIPO script file..........OK
Running PIPO (GDS Stream-In).......
        ************************************************************
        *                                                          *
        *      CADENCE Design Systems, Inc.                        *
        *      Virtuoso(R) Physical Data Translator 4.4.6          *
        *      EXEC TIME : 22-Sep-2003  20:10:36                   *
        *                                                          *
        ************************************************************

*** There were 0 error and 1 warning messages ***

Statistic and more information, please check /home/dparent/lfsr_ami06/PIPO.LOG fi
le.

NORMAL EXIT ...

* Warning * Years in non-standard 4 digit format found in the stream file. Such y
ears have been converted to standard format. The tool that wrote the stream file
should write the years in standard format, which is the number of years since 190
0. For more information, please see the documentation/KPNS.


Cleaning up........................OK
Good by.
eecad28.engr.sjsu.edu%
```

Figure 19: Output of Conversion Process.

You can now see the layout view of the lfsr in icfb's layout tool (Figure 20).

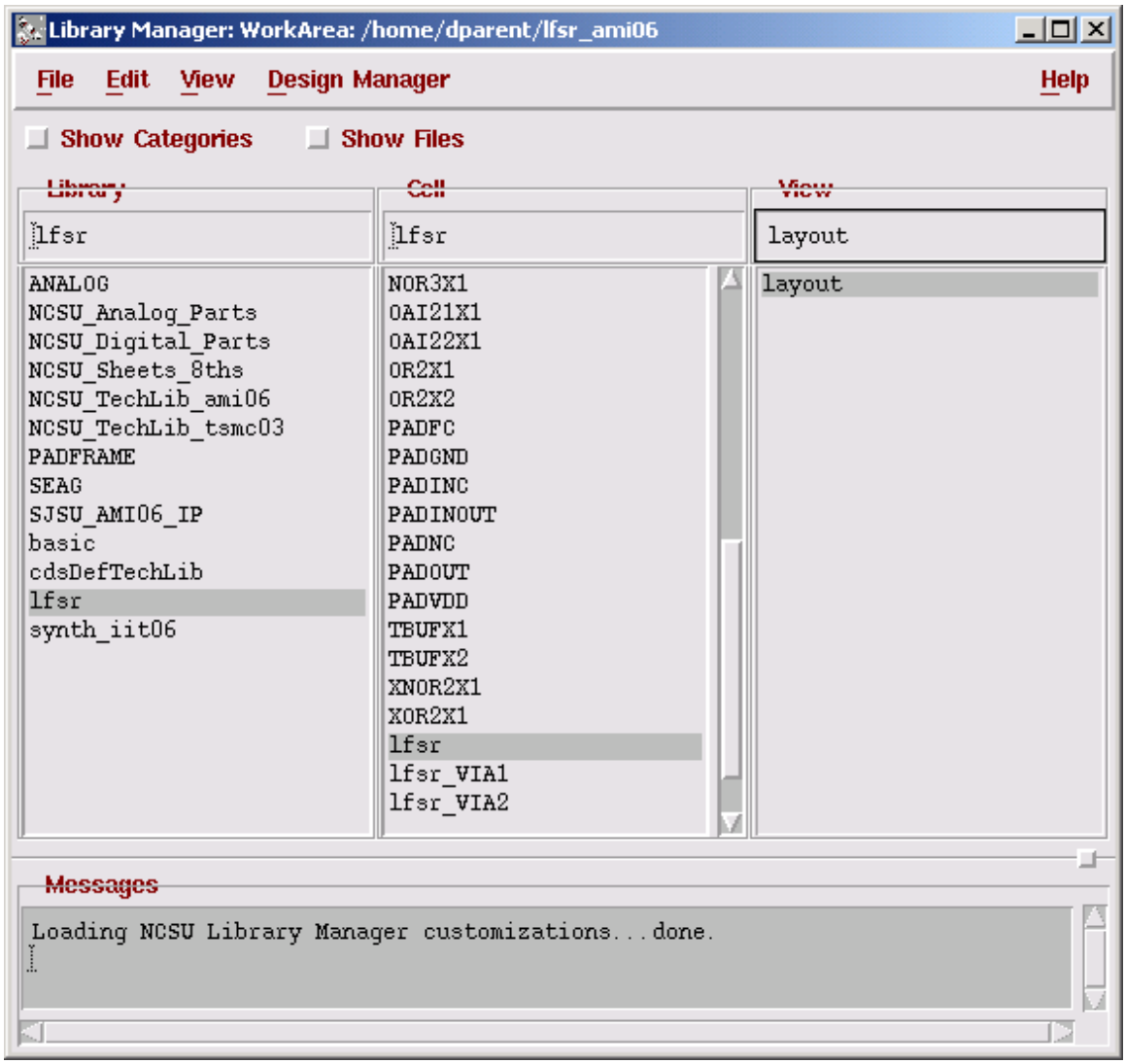


Figure 20: lfsr converted.

We can see the final layout in Figure 21.

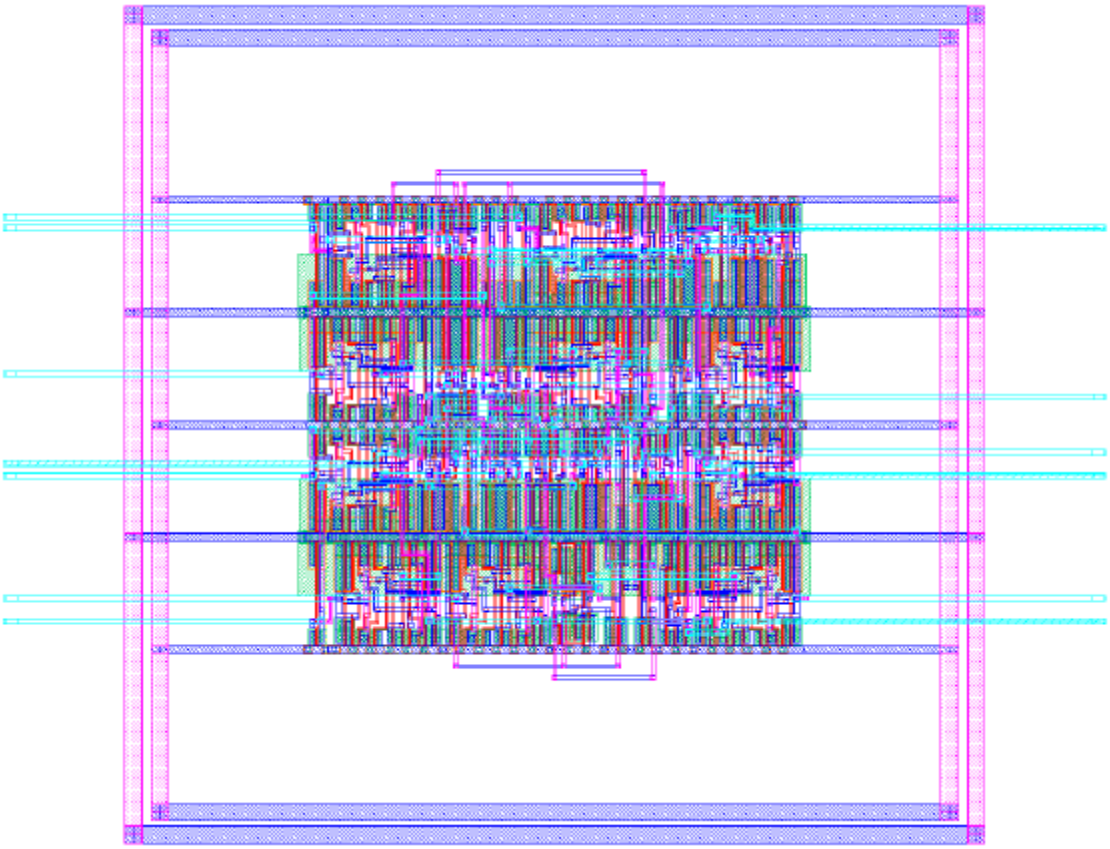

Figure 21: Final Layout of lfsr.

## Checking the timing of the circuit using Spice:

I am not quite sure why but the extraction from the library you create your design in (in this case lfsr) does not work. You have to create another library like lfsr_test and copy the layout into it, and then run extract. You can do this from the icfb session you started before you sourced the env.opts file to het the paths for the place and route tool to work. If you closed that icfb session, then start another terminal and then start icfb again.

Create a library according to Figure 22.

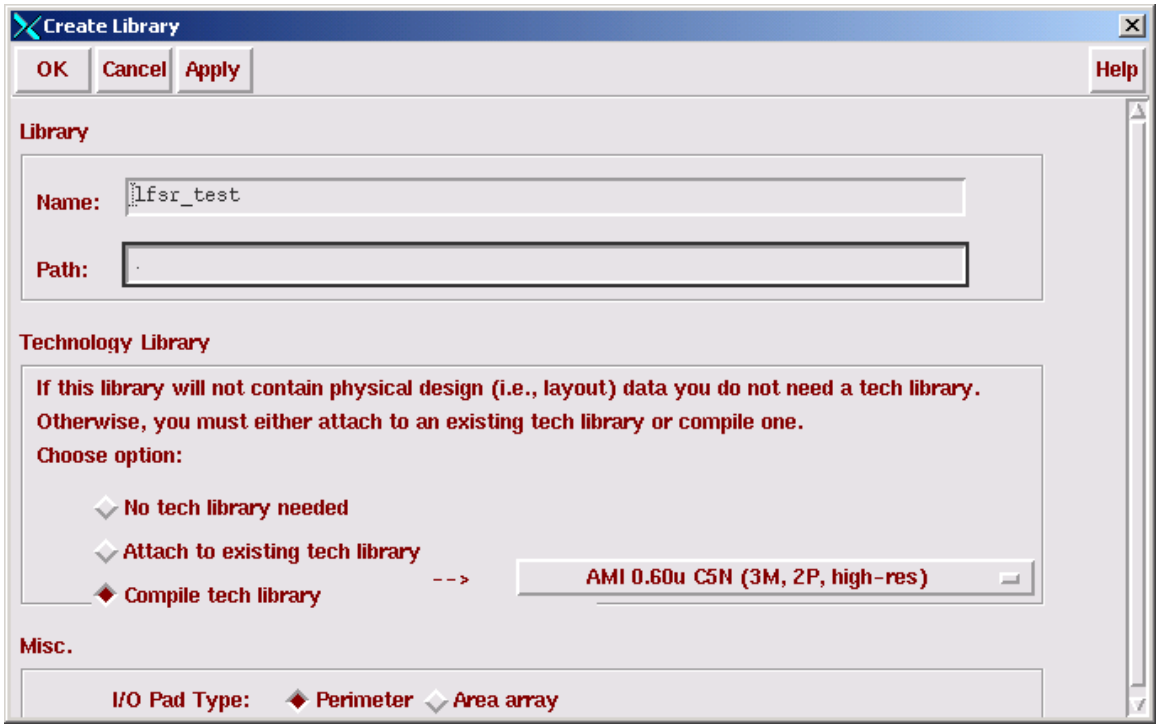


Figure 22: Creating another library to run extraction on the layout of the lfsr.

Copy the lfsr, layout view into the library lfsr_test according to Figure 23. (If you get any warnings, just click on fix errors and then proceed.)

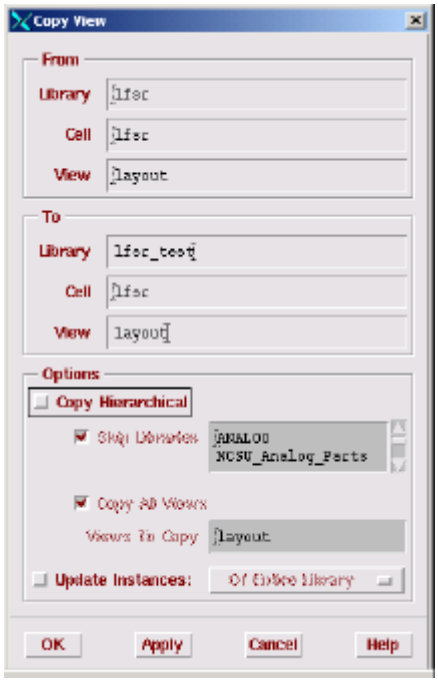


Figure 23: Copying the lfsr layout view.

Now open up the lfsr layout view from the lfsr_test library (Figure 24).

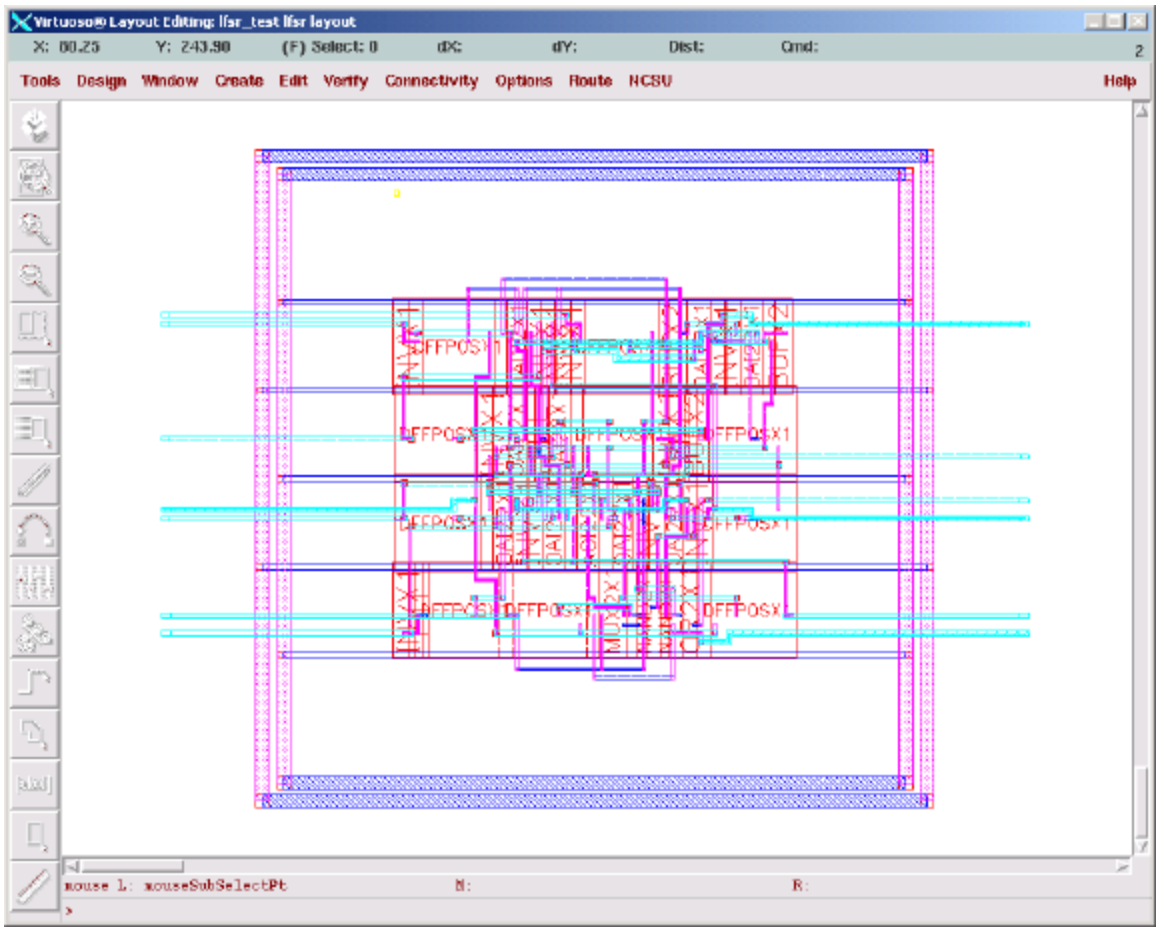


Figure 24: Layout view of lfsr in lfsr_test library.

Go to Verify, DRC check and clcik on ok in the pop-up like Figure 25.

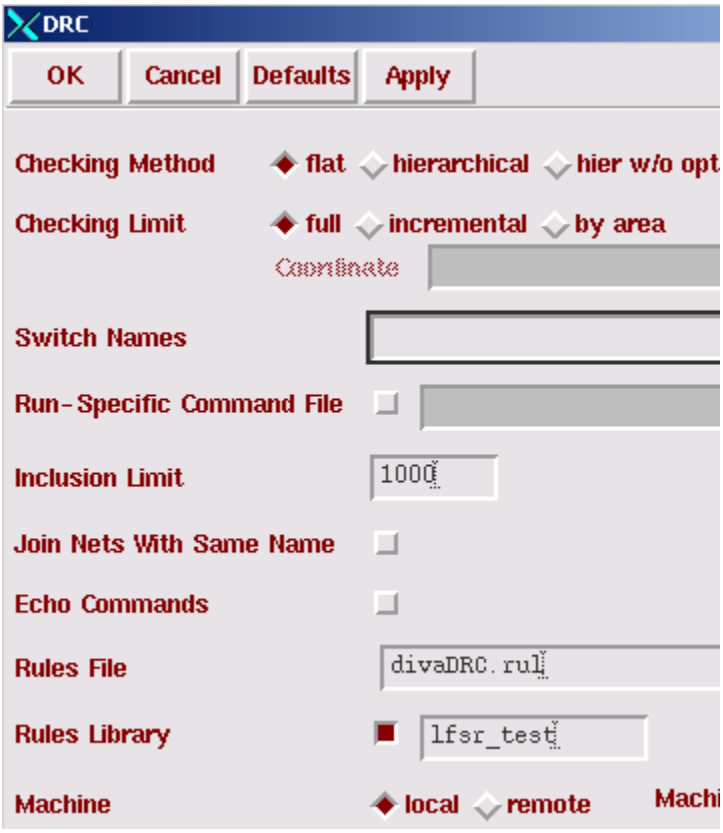


Figure 25:DRC Check.

If you get some errors about malformed metals as in Figure 26, do not worry about it.

```
********  Summary of rule violation for cell
# errors  Violated Rules
      2   Improperly formed shape - metal1
     13   Improperly formed shape - metal3
     15   Total errors found
```

Figure 26: Acceptable errors from DRC.

The pins are have to be created from labels that have been automatically generated.

To create these pins first delete all the DRC error markers by going to Verify...Markers... Delete All.

Go to Create... Pins from labels in the Virtuoso editor. You should see a pop-up like Figure 27.

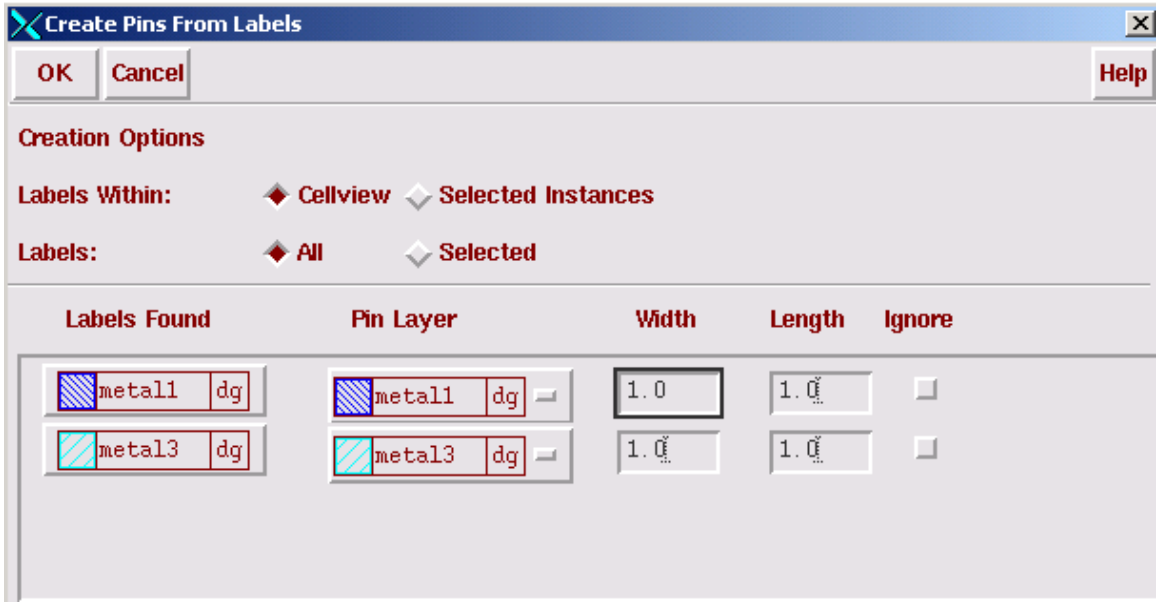


Figure 27: Creating Pins

Just click ok.

Now go to Verify..Extract to extract the circuit for simulation (Figure 28).

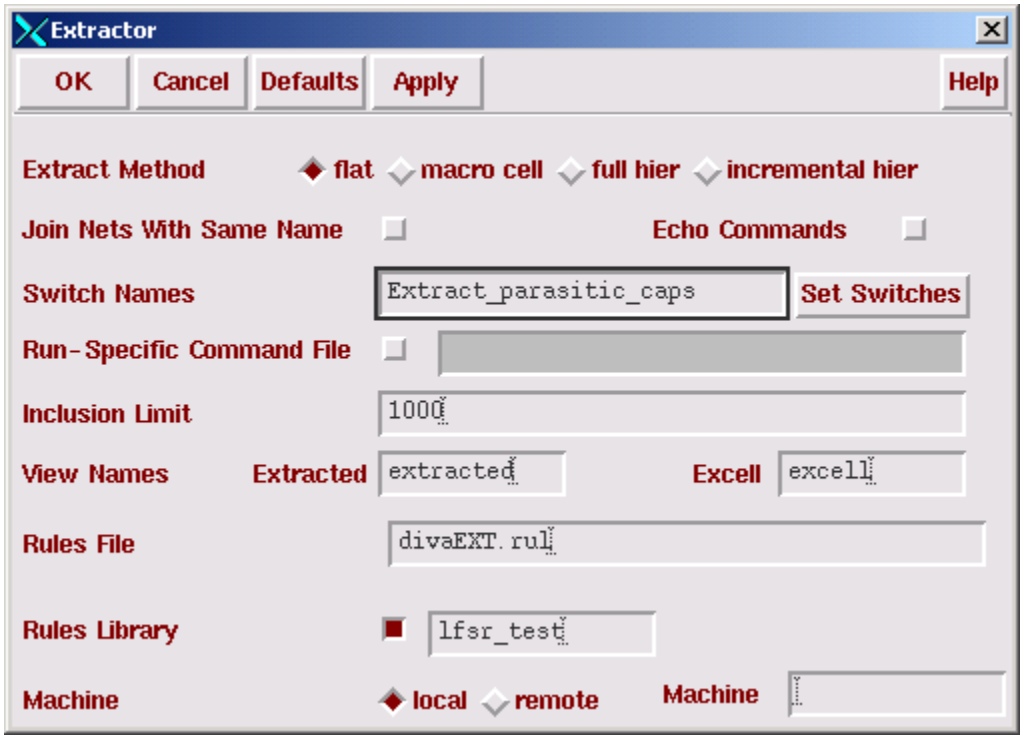


Figure 28: Extracting with parasitic Capacitances.

Note: The NCSU kit will not do high speed clock trees very accurately. Be Carefull!

The extraction report should contain no errors.

## Running a Spice Simulation:

Open up the extracted view and start the analog environment (Figure 29).

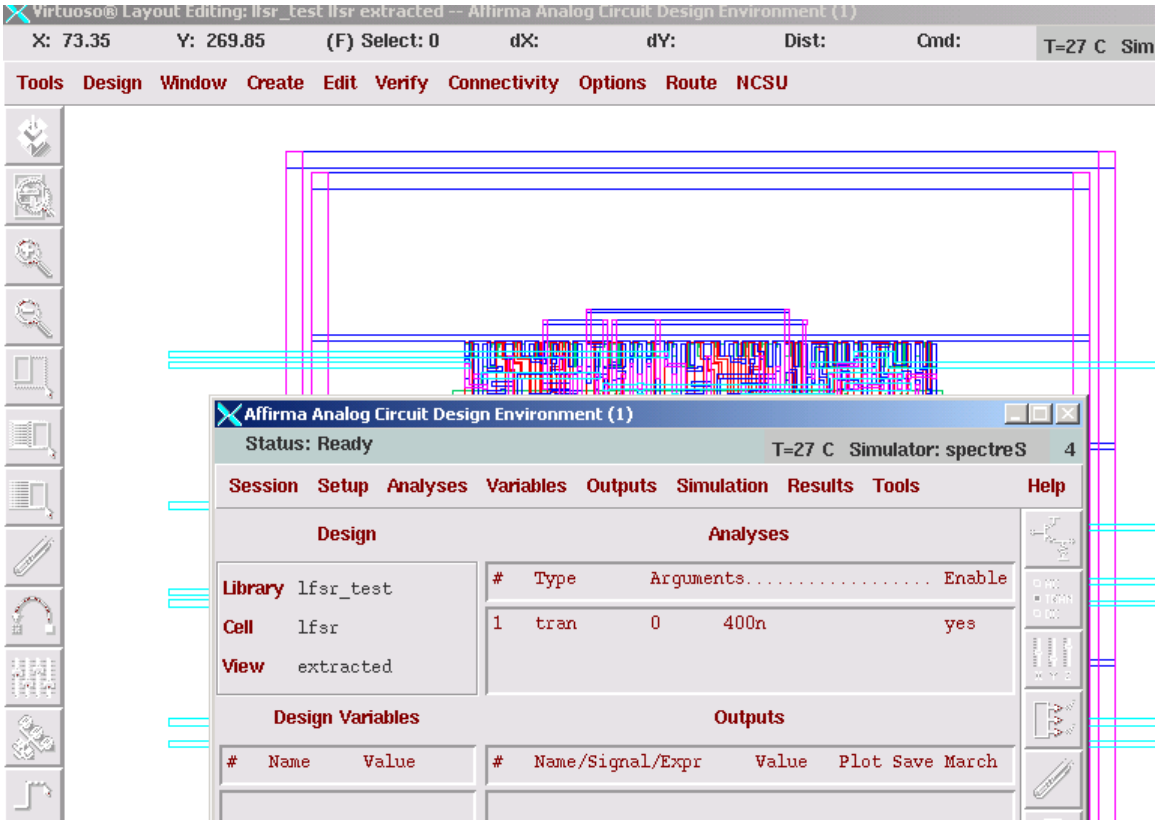


Figure 29: Starting Affirma.

Make sure extracted is in the view list under Setup... Environment.

Make sure the transient analysis is set to 400ns.

To set up the test vectors in the Affirma window go to Setup... Stimulus.. Edit Analog.

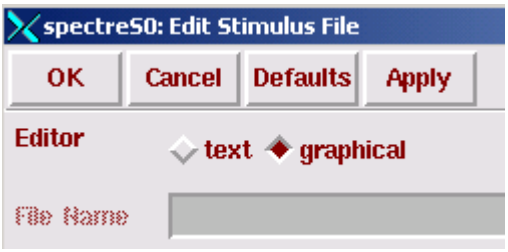


Figure 30: Seting up the stimulus file.

Click graphical in the pop-up (Figure 30). and Figure 31should appear.

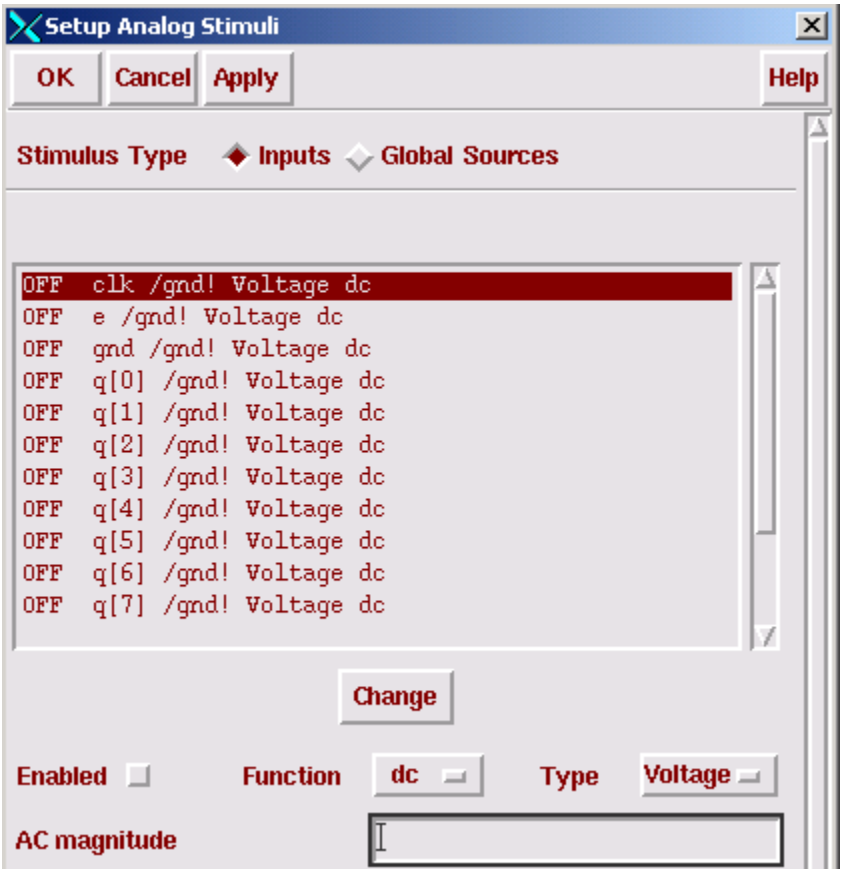


Figure 31: Setting Stimulus File.

Now wee need to change the inputs to DC sources such as VDD and Gound and inputs to Vpulses. If it is an output leave it alone (q[0]...q[9]

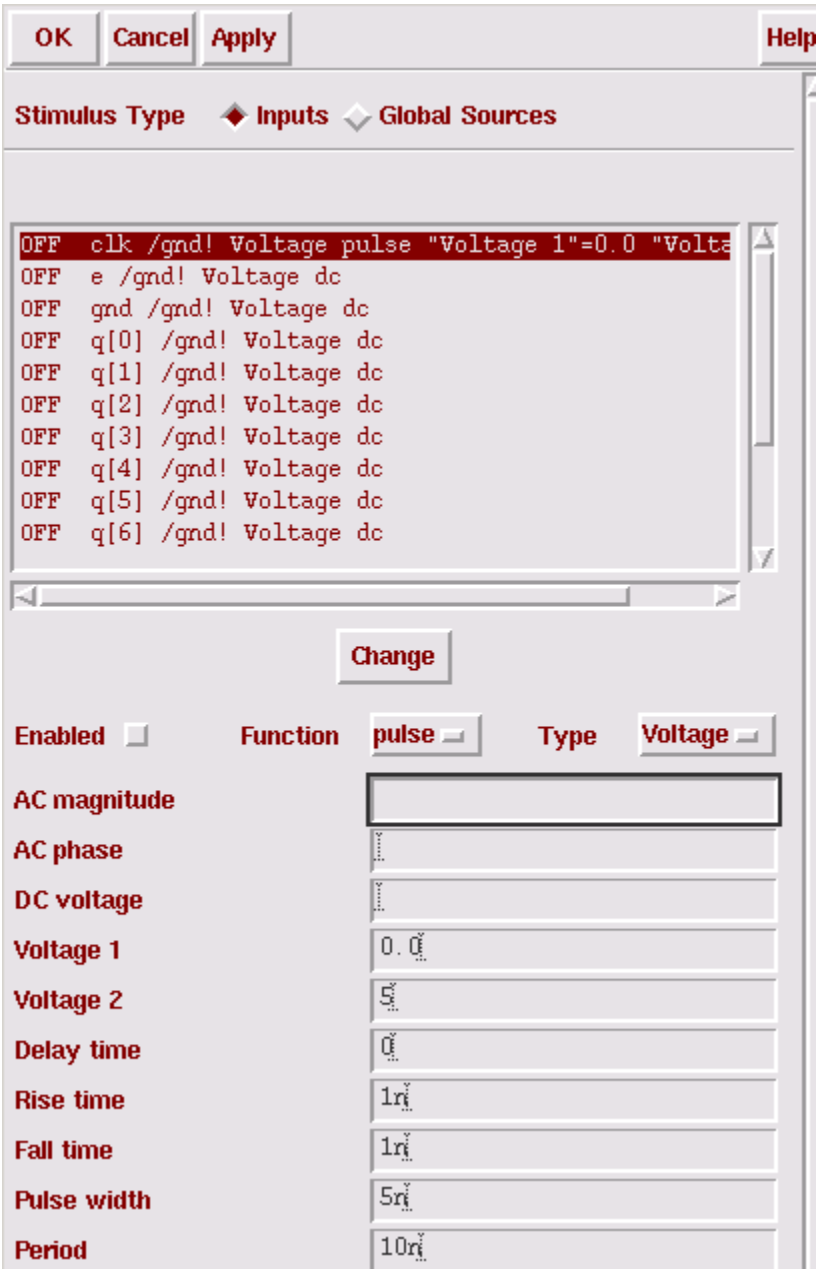


Figure 32: Setting the Clock.

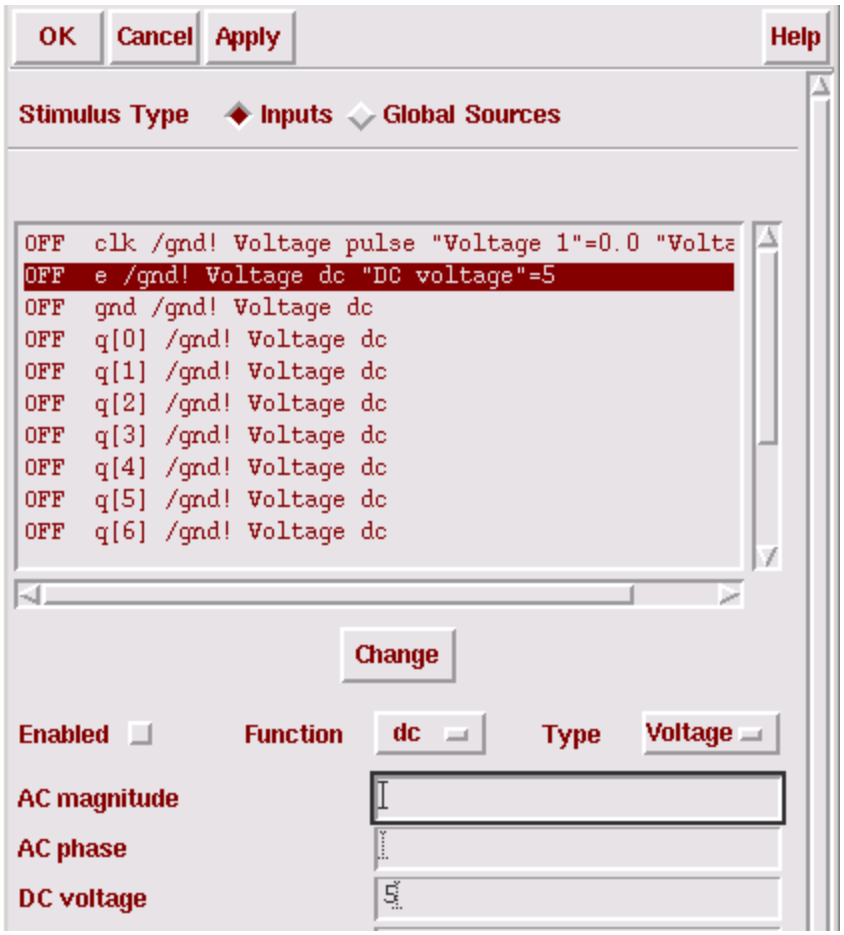


Figure 33: Setting Enable to logic 1 (5V).

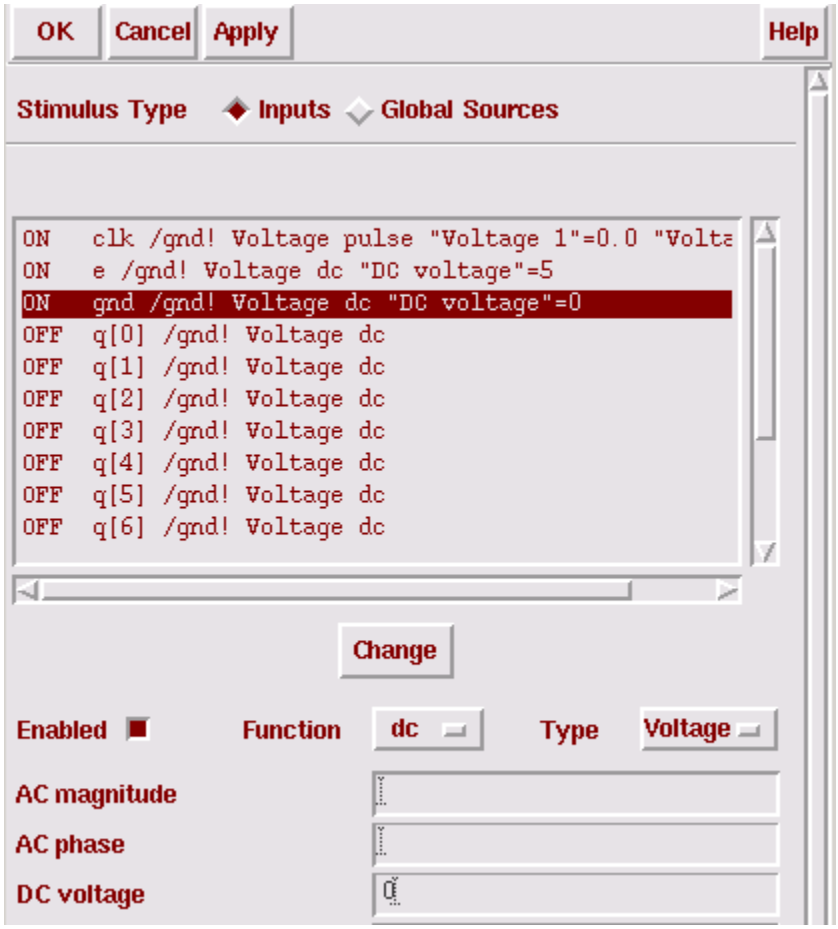


Figure 34: Setting gnd to zero volts.

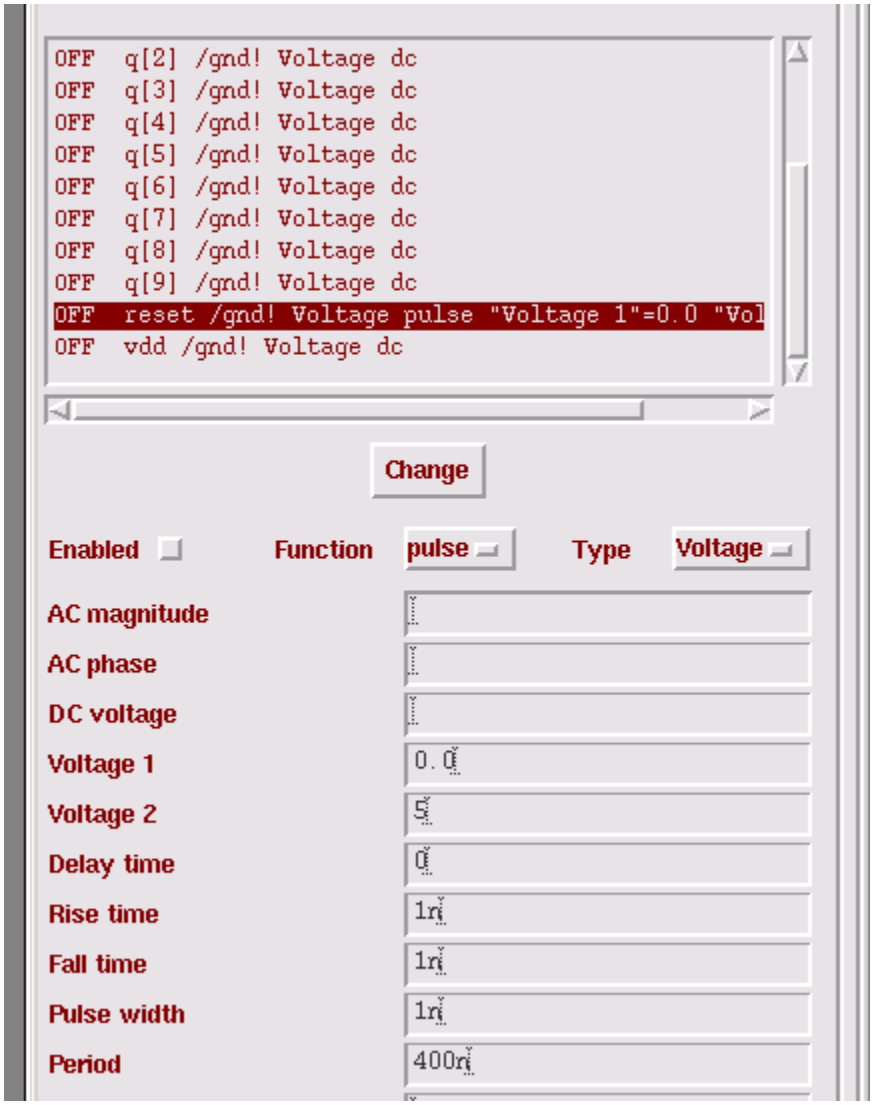


Figure 35: Changing reset.

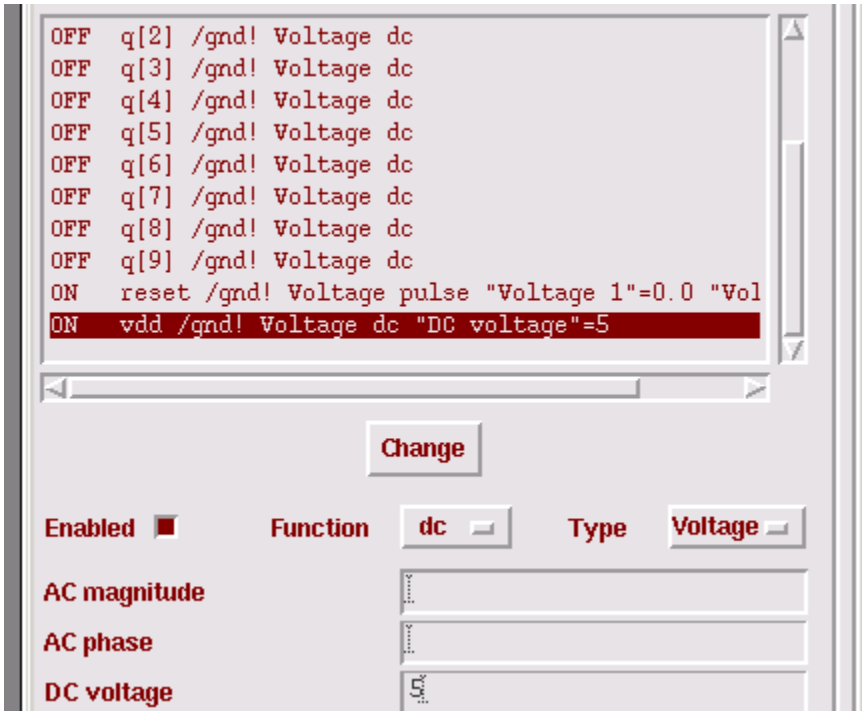


Figure 36: Setting VDD to 5 Volts.

**Make sure vdd, reset, e, clock, and gnd are all enabled!**

Click on apply, but do not close the window.

In the Affirma window Go to Outputs... To be Plotted... Select from schematic and click on all the metal3 lines until the high light (Figure 37)

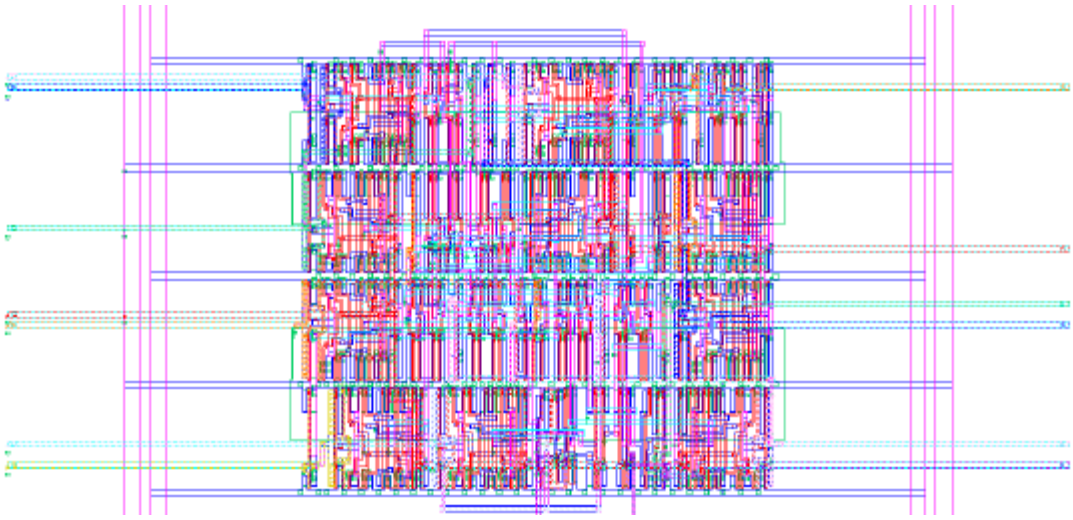

Figure 37: Selecting Outputs to be Plotted.

Look in the Afirma window to make sure all the inputs and outputs are listed.

Run the simulation.

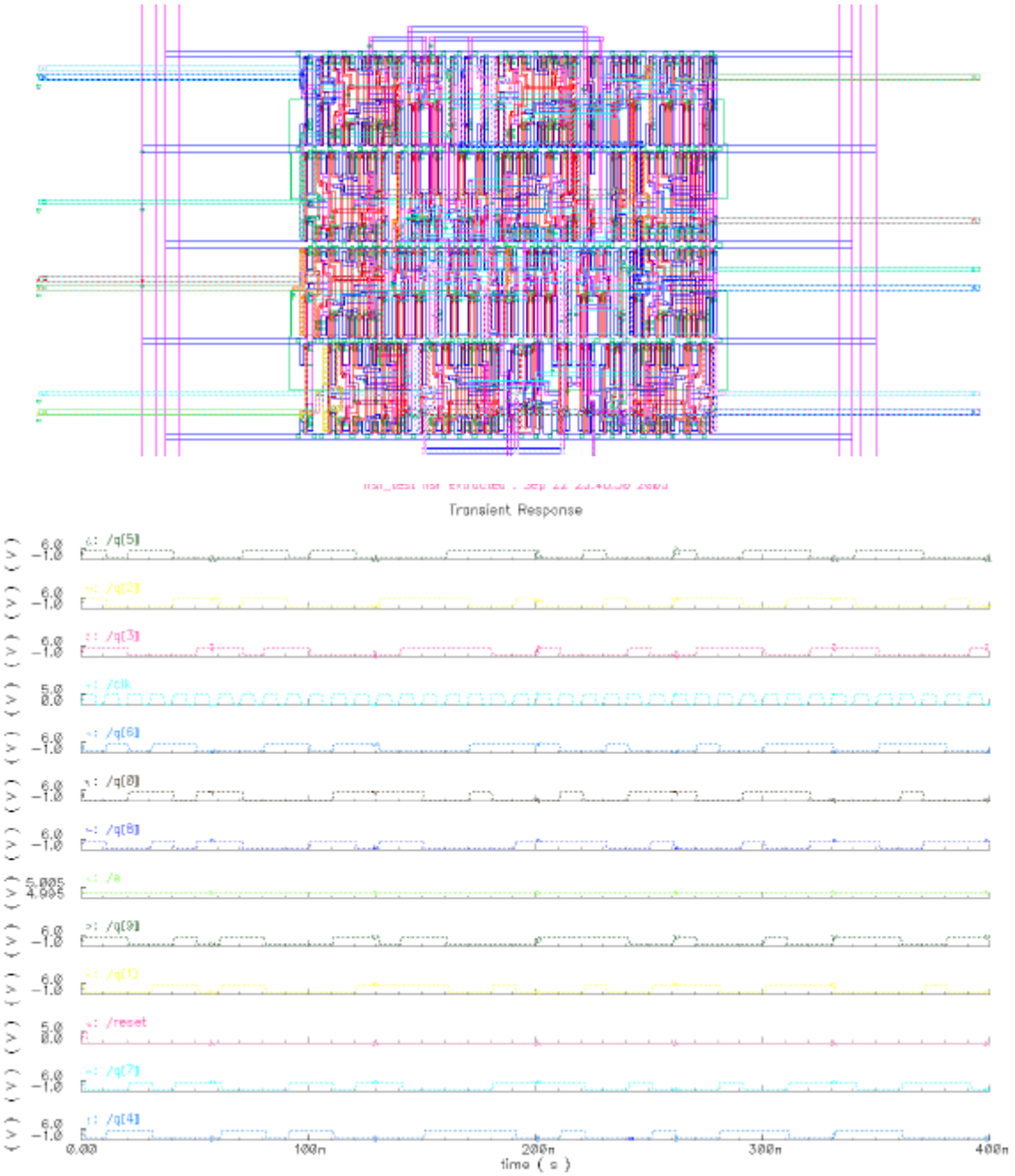


Figure 38: Final Simulation.

You should get something like Figure 38.

**This is the very basics of this flow. I have not shown you how to set any synthesis or place and route parameters. You use it at your own risk.**