Synopsys Custom Designer Tutorial for a chip integration using the University of Utah Standard Cell Libraries In ON Semiconductor 0.5u C5 CMOS

Version 12.0

Overview

This tutorial will take you through the following steps:

1) Create a new library and add standard cells and copy a pad frame. The copy of the pad frame will serve as the final chip view.

2) import your GDS file and verilog netlist from IC Compiler – referred to as the "core" - into the new library.

3) fix pin text on the core

4) instantiate the core into the top-level chip in both layout and schematic

5) wire the core and pad frame together in layout and schematic

6) run DRC, extraction and LVS on the chip

7) generate a chip level GDS to be sent to MOSIS for fabrication

8) reimport the chip GDS into a new library and re-run checks.

9) Submit the GDS to MOSIS.

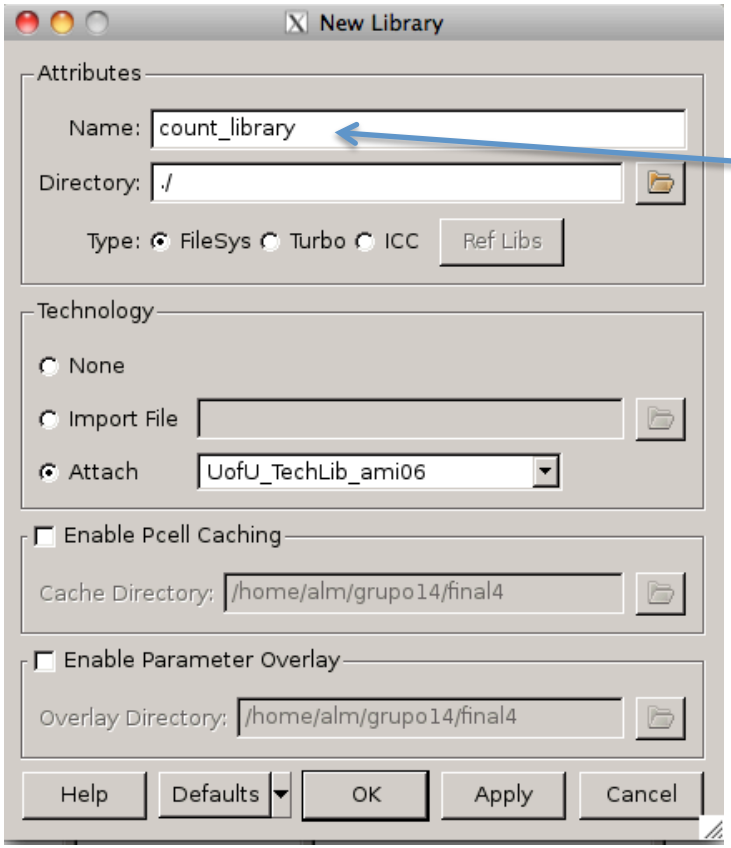
Start the tool

Create a working directory in unix – I made one called "lab7c" using "mkdir lab7c"

Add cds.lib, lib.def and color map files into the working directory. These files are on the website.

Invoke the tool with "cdesigner &". You may need to set up your environment variables first. Like "synopsys.env" - depends on your account.

Open the library manager and create a new library. I would name it something like:



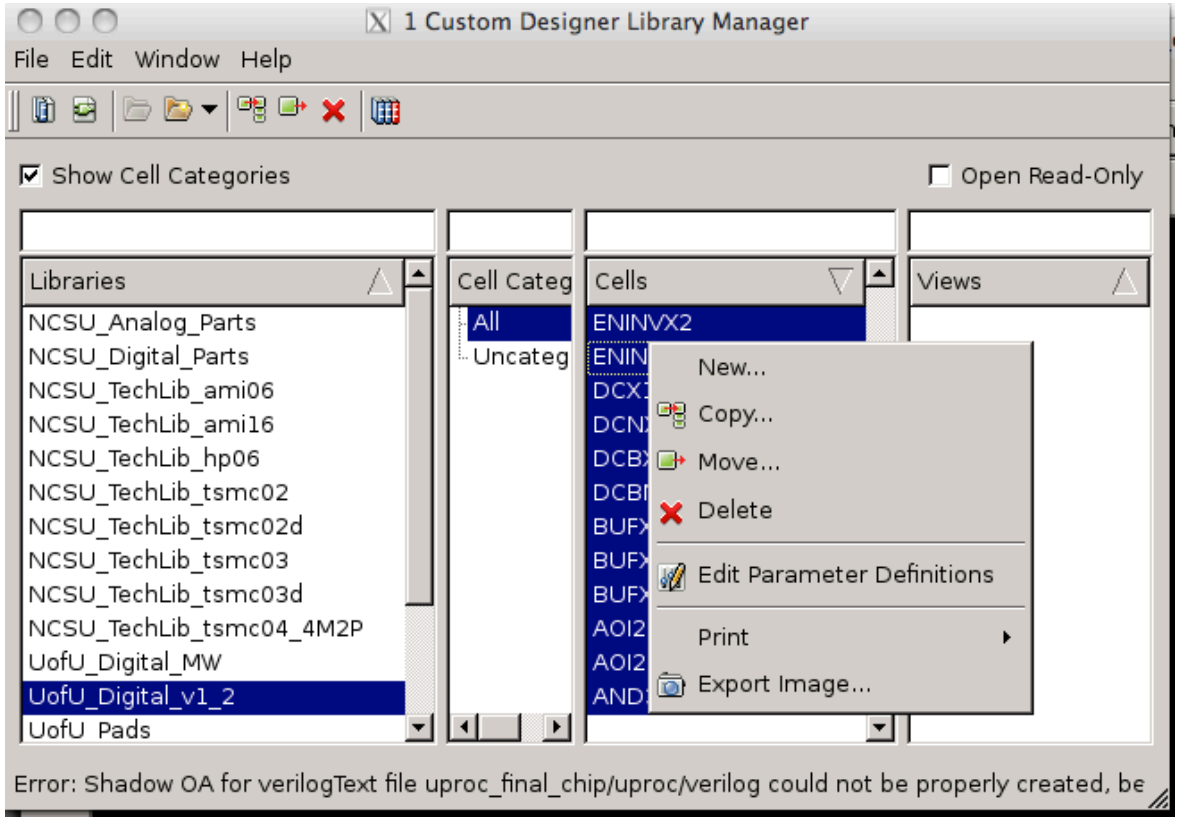
<project_name>_library

In this example I made a simple counter so the project is called count_library. I try to use this throughout this tutorial. You need to replace this name with your project name throughout.

Fill your library with standard cells

Select and copy all cells from the UofU_Digital_v1_2 library into your new library using Library Manager. Right click on the cells to copy.

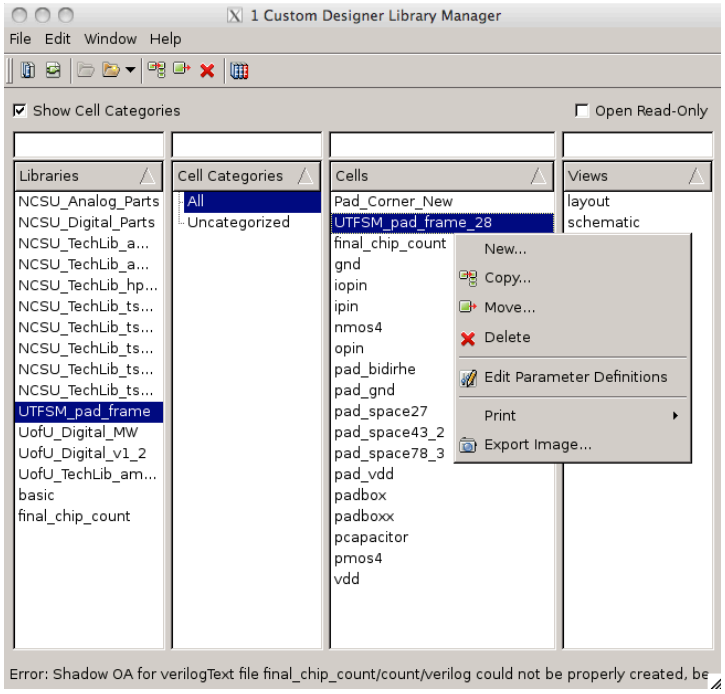
When copying libraries you may see a dialog box about replacing cells. These are all the same so just re-copy and replace any redundant cells.

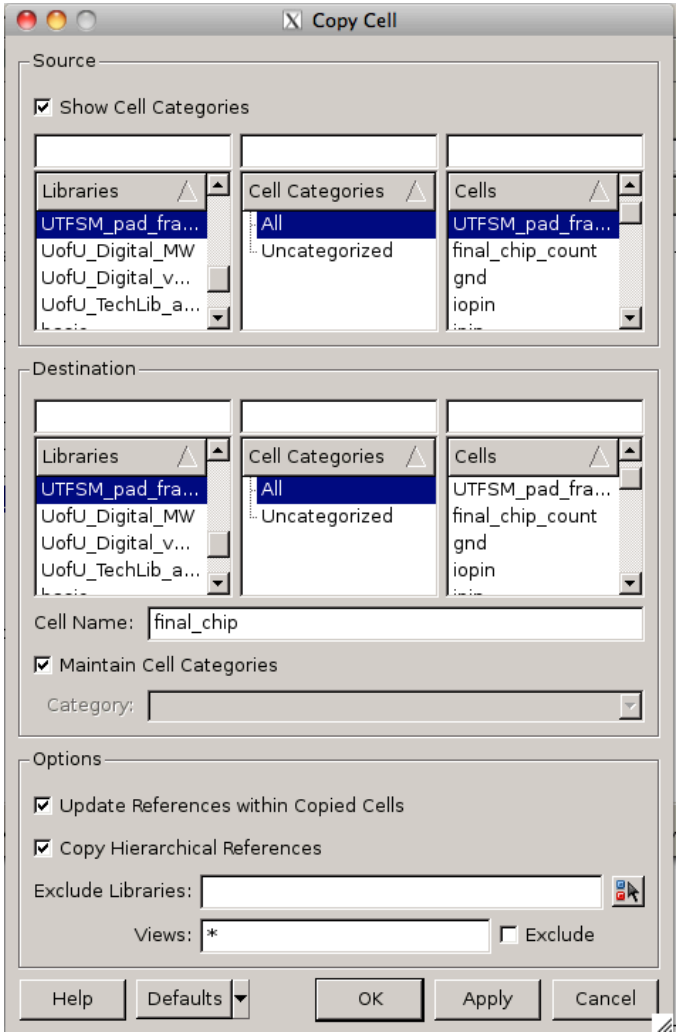


Fill your library with a pad frame that will serve as your chip base

Also copy the UTFSM_pad_frame cell into your library but rename it “final_chip”.

You will edit this cell by introducing your core into the cell and connecting the core in both layout and schematic to create the final design to be fabricated.





Import core into library as GDS – step 1

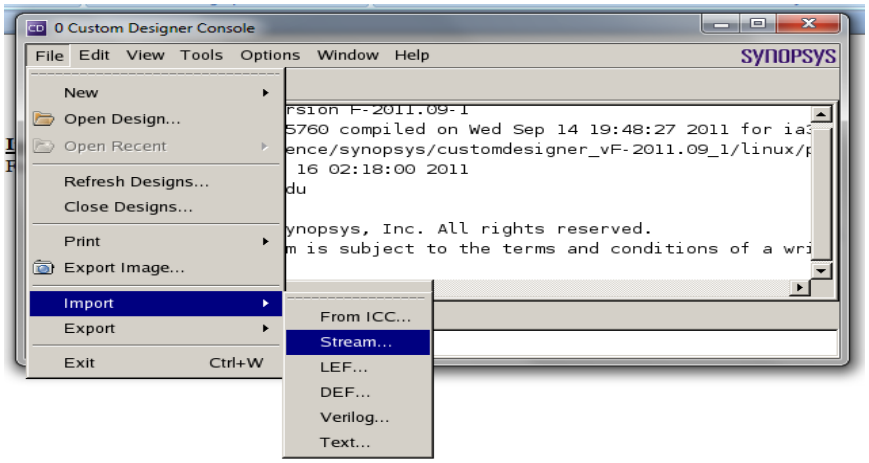
From Custom Designer Console we can import GDS.

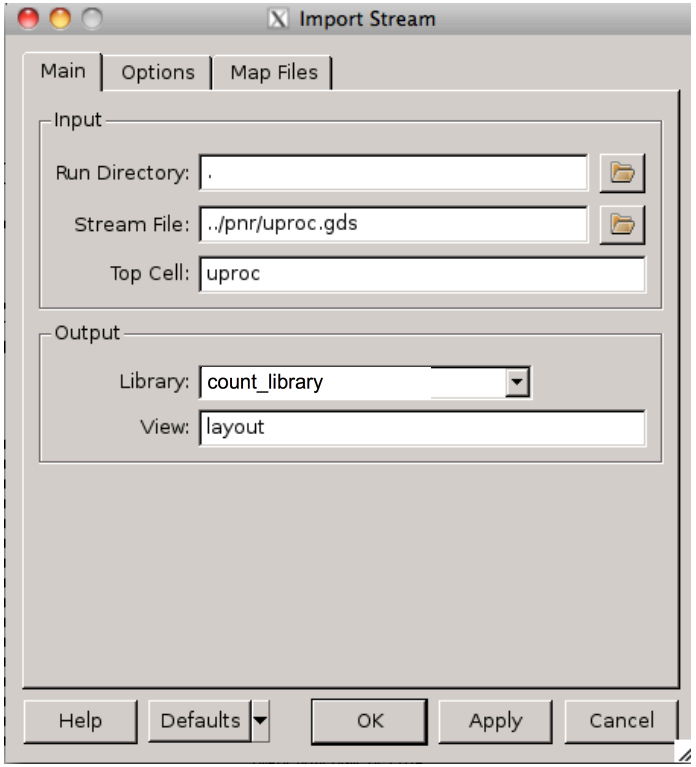
From the Console, choose File > Import > Stream.

Under the Main tab,

specify required input details: Run Directory; Input GDS Stream file and top cell.

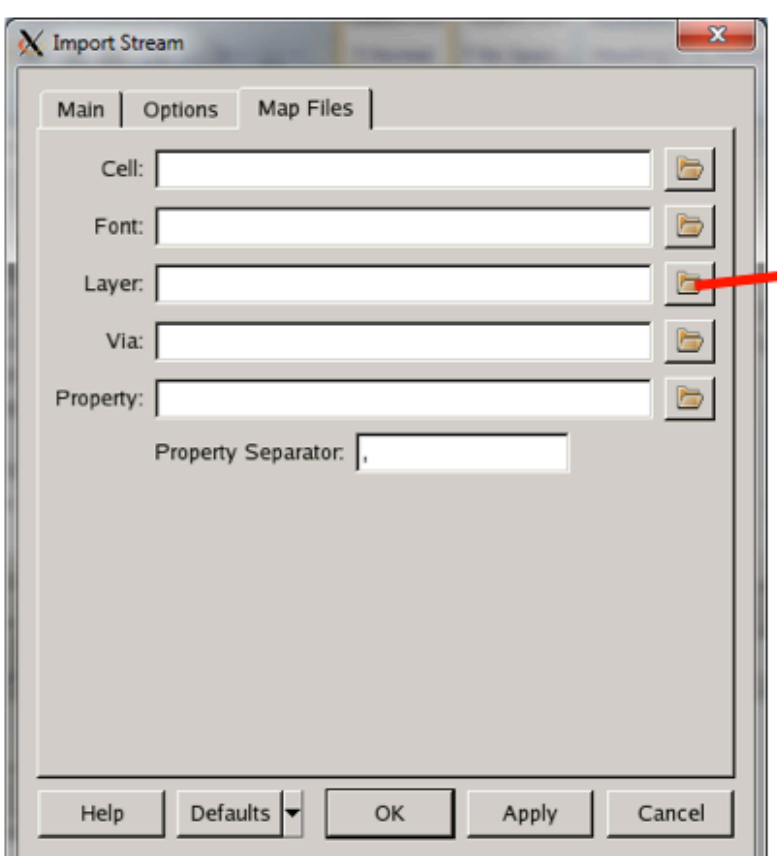
Output details: Output Library, view: Layout





Import core into library as GDS – step 2

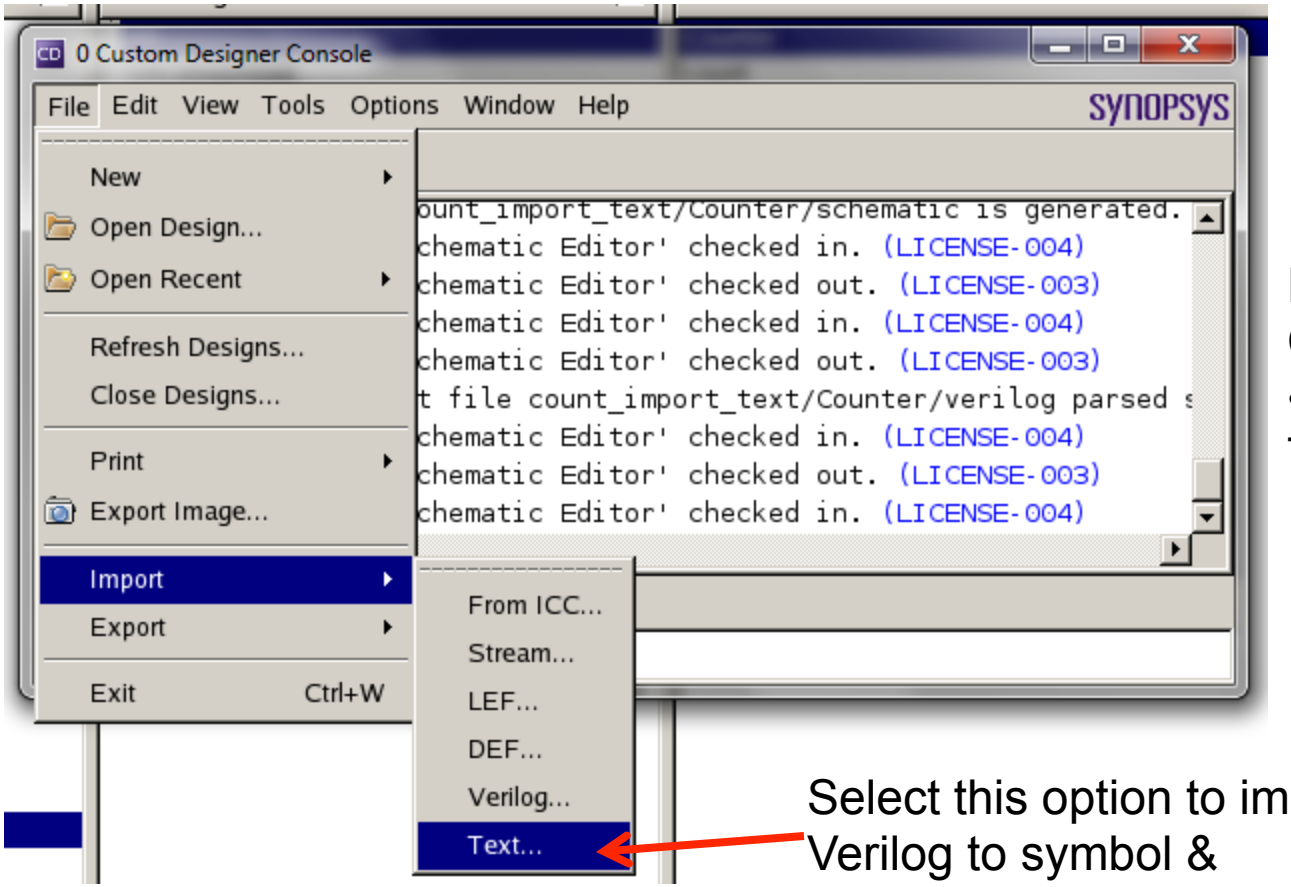
Select the correct layer map. (UofU_TechLib_ami06_version2.layermap) included in the chip tutorial zip files and in the website on lab 7C.



Browse the layer map in Map files tab – this was provided with the tutorial

UofU_TechLib_ami06_version2.layermap

Import Verilog to create Symbol and Netlist view for the Core in Custom Designer

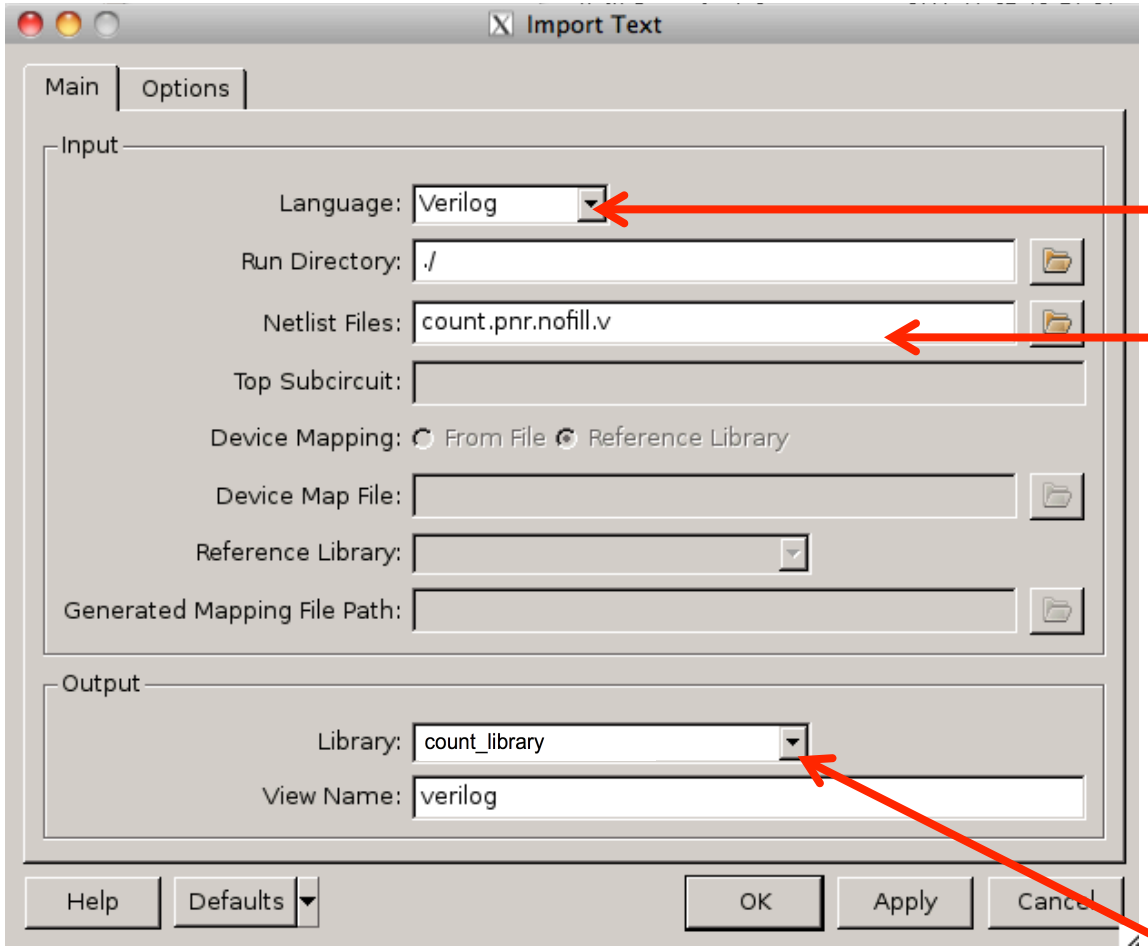


From Custom Designer Console

•Go to File → Import → Text

Select this option to import Verilog to symbol & schematic

Import Verilog for core: Choose Input Netlist File and Target Library



Select the language as Verilog

Browse for your post-place-and-route netlist. My ICC script creates a verilog netlist with no fill cells. Fill is only for implementation and has no logic function or connection

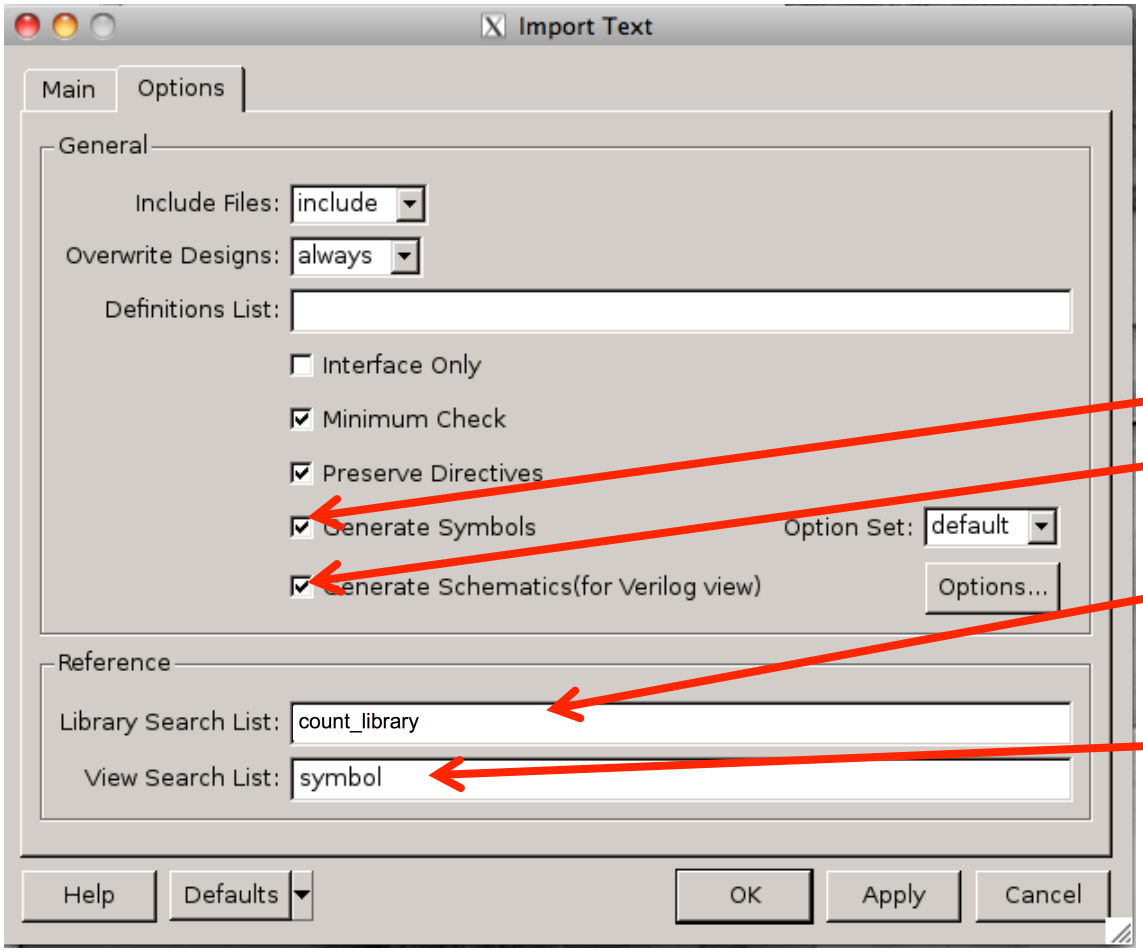
Use the following unix/linux command to remove fill:

```
grep –v FILL netlist.v > netlist.nofill.v
```

Choose the library that you created previously

Import Verilog for core: Choose Input Netlist File to create a symbol /schematic for your core

This step can a minute to perform if you have many gates in your core.



Creates a symbol for your chip level schematic and a schematic for LVS.

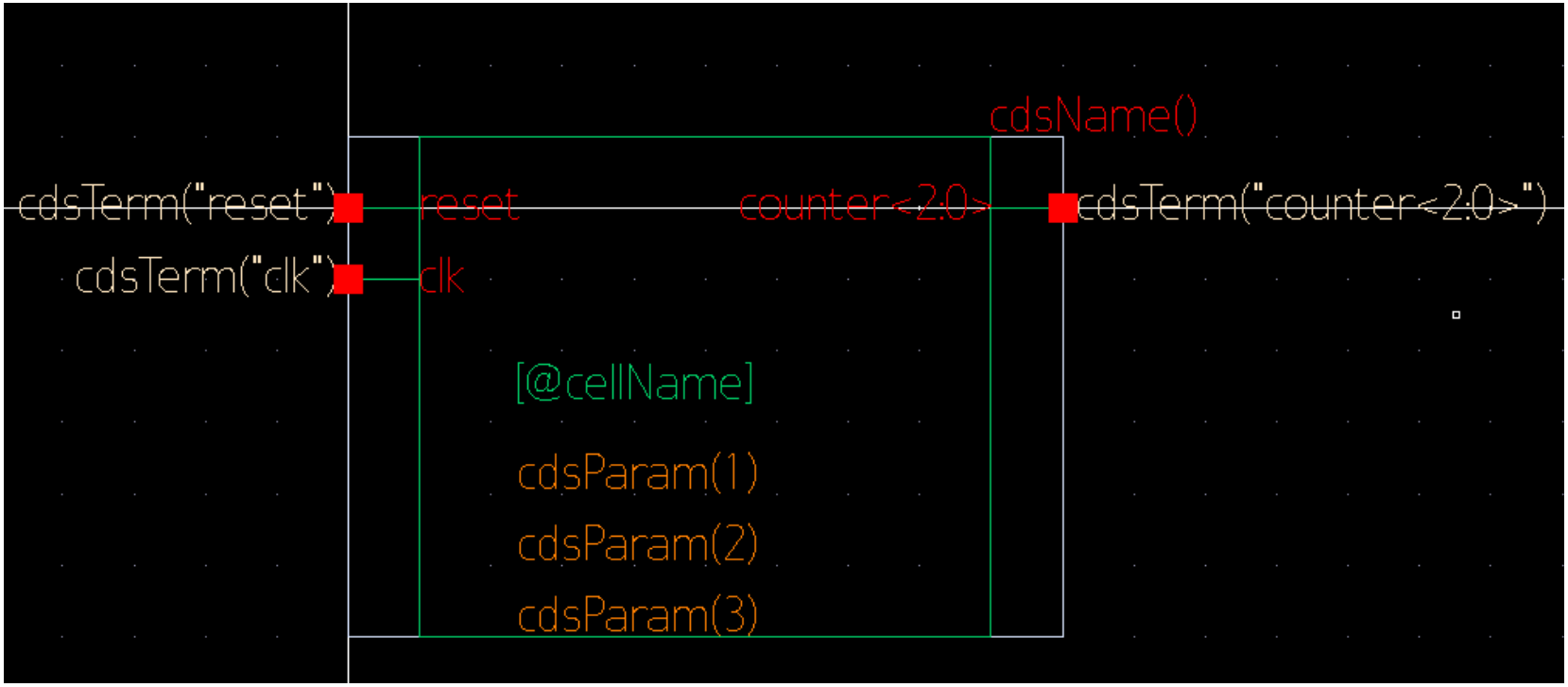
This is your working library after you copied in all the standard cells.

This can be either schematic or symbol.
I prefer symbol.

Import Verilog: View of created symbol with port names

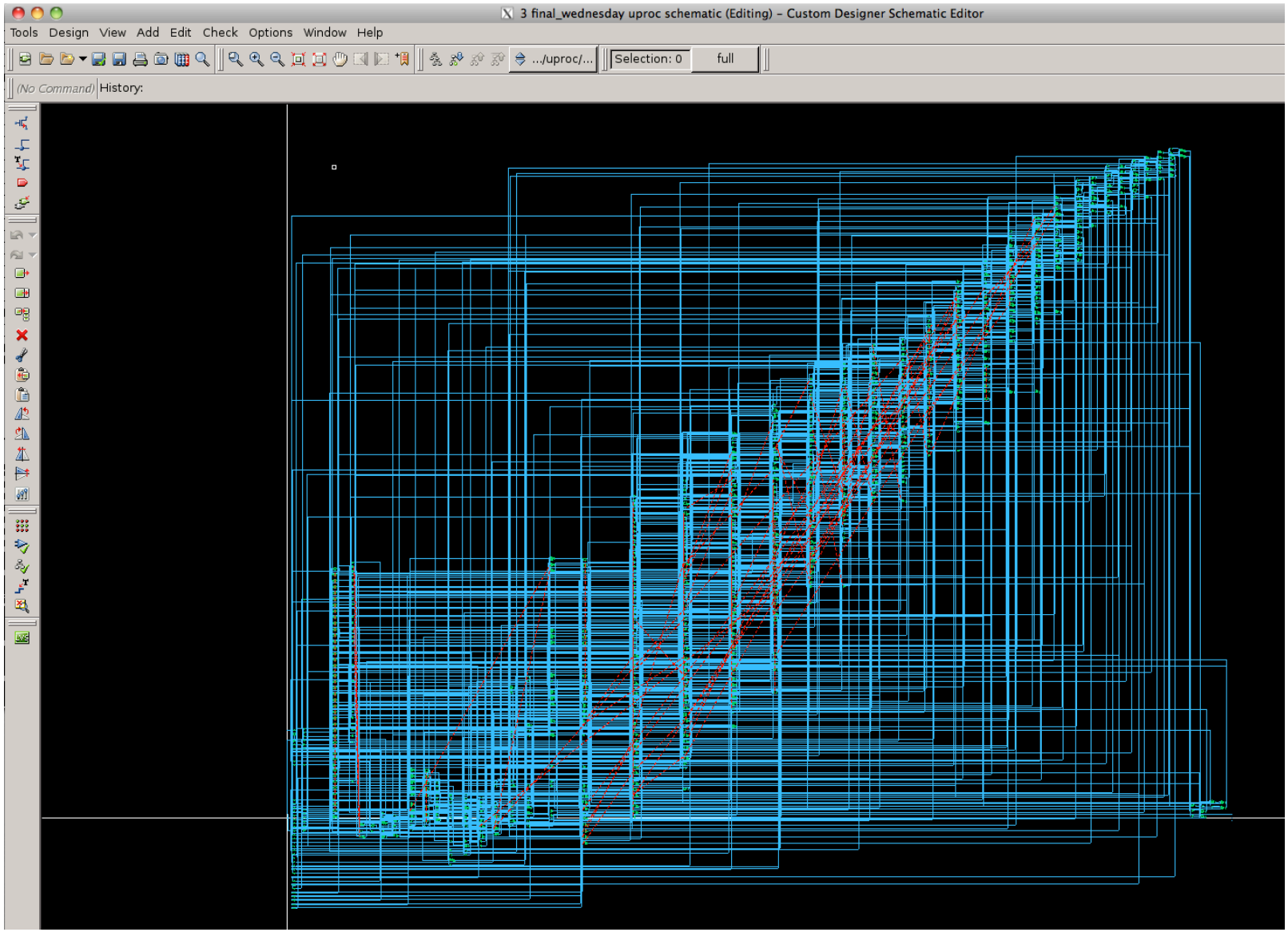
Go to your core cell view and double click the symbol to see something like this.

This will be used in your chip level schematic to represent and to connect to your core.



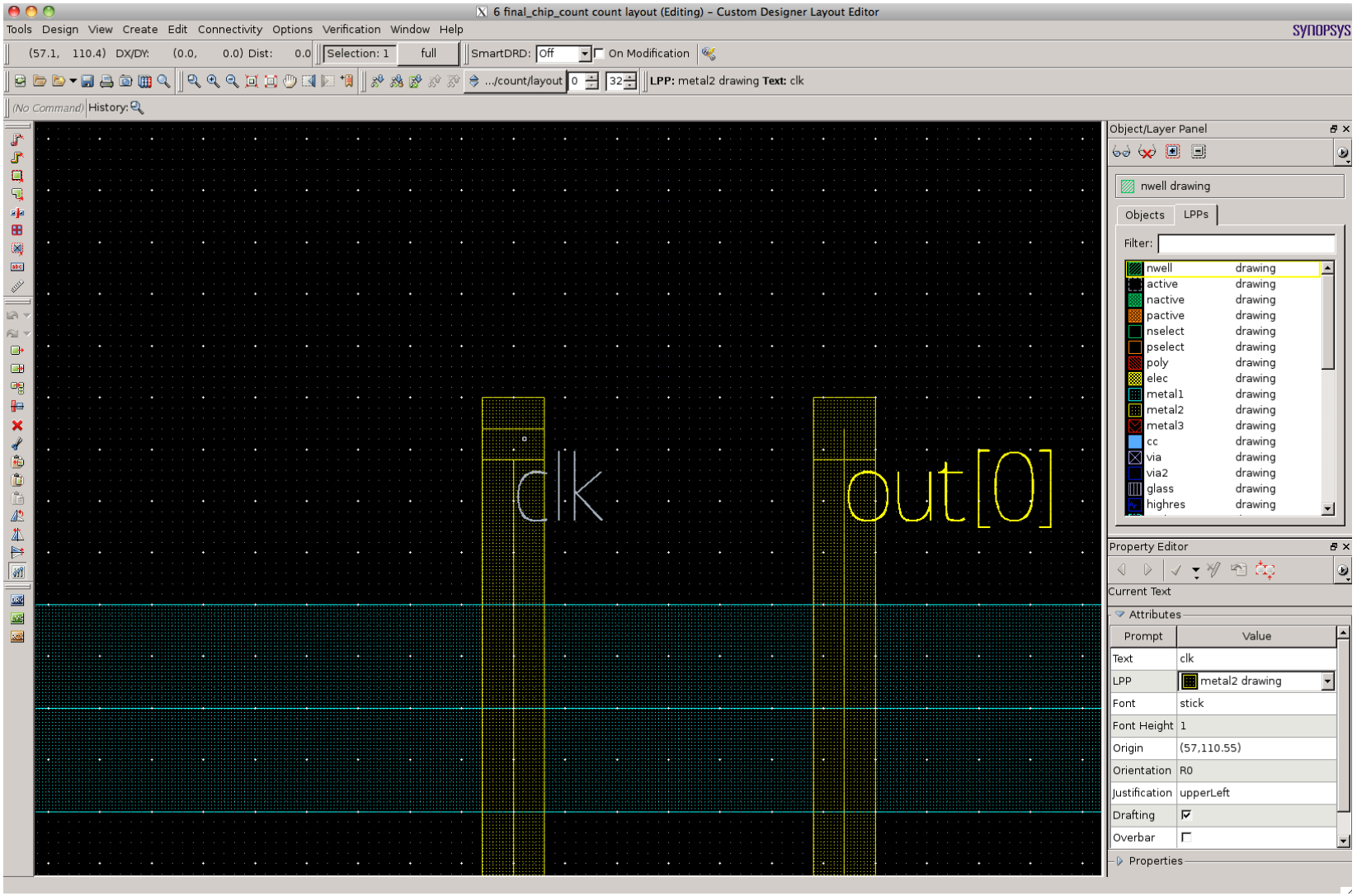
Import Verilog: View of created schematic necessary for LVS checking at the end

Don't worry about red fly lines in the schematic. I'm investigating this but have found no problems



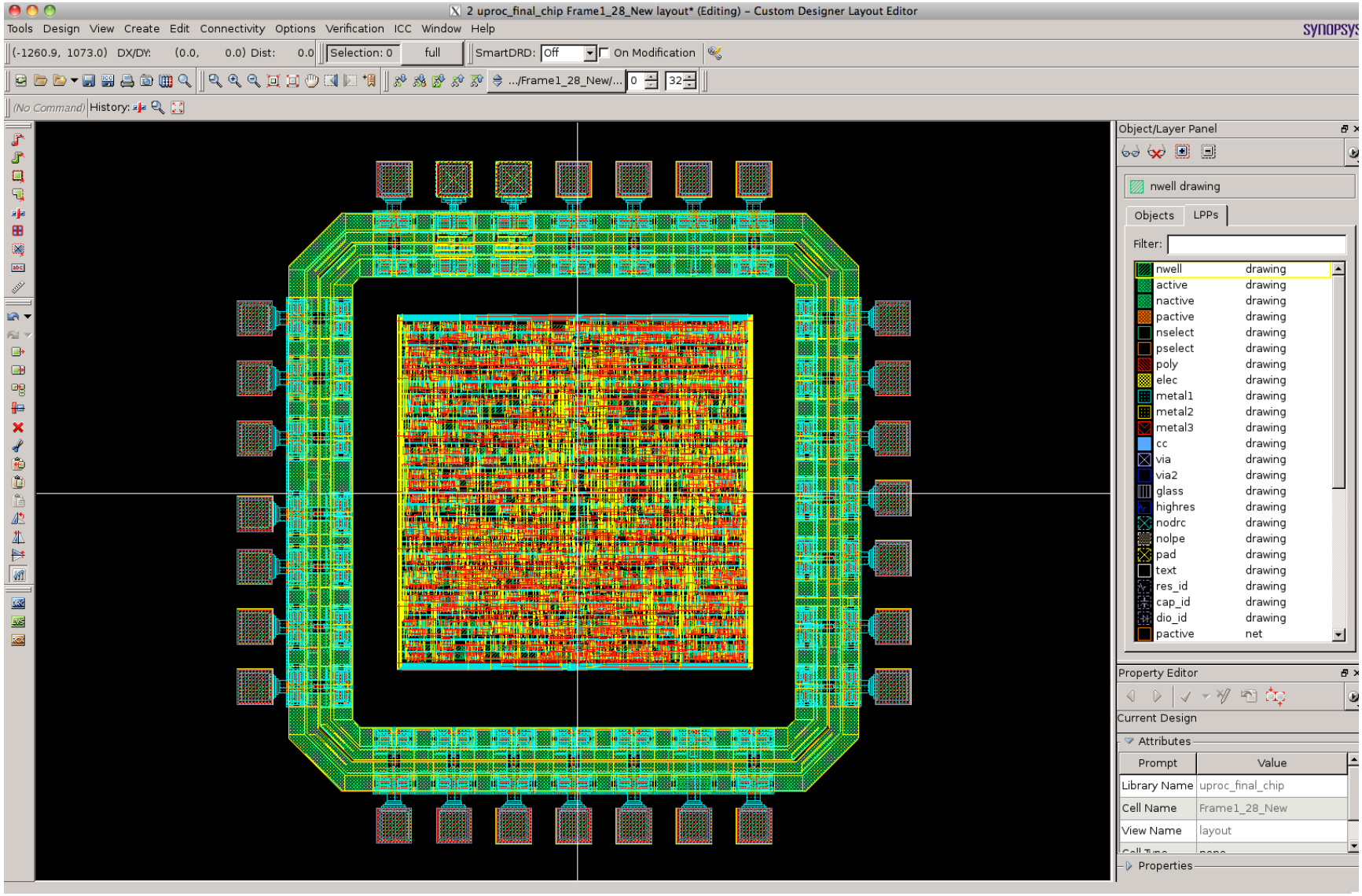
Fix your core ports (read and perform carefully.)

The GDS import will have included text for the pins of your core but pins must be changed from metal to text layer or you will get a DRC error as the layers are impossible to manufacture. Click the title and change to text layer from metal 2 or 3.



Edit your chip layout

Use “i” (or edit->add instance) to create an instance. Select your core and place in the center. Read the all layout slides prior to making connections to the core.



Complete the pin-out spreadsheet and save for chip arrival and testing. Spreadsheet is on website with this tutorial.

This will help guide you in the layout and schematic editing you will do in the next steps.

Rotation of Die								
270	180	90	0					
Pin	Pin	Pin	Pin	Function	Enable	DataInput	DataInputN	DataOutput
8	15	22	1					
9	16	23	2					
10	17	24	3					
11	18	25	4					
12	19	26	5					
13	20	27	6					
14	21	28	7					
15	22	1	8					
16	23	2	9					
17	24	3	10	VDD	N/A	N/A	N/A	N/A
18	25	4	11					
19	26	5	12					
20	27	6	13					
21	28	7	14					
22	1	8	15					
23	2	9	16					
24	3	10	17					
25	4	11	18					
26	5	12	19					
27	6	13	20	GND	N/A	N/A	N/A	N/A
28	7	14	21					
1	8	15	22					
2	9	16	23					
3	10	17	24					
4	11	18	25					
5	12	19	26					
6	13	20	27					
7	14	21	28					

Layout hints

Press "z" and draw a rectangle to zoom to area

Press "shift f" to see internals of lower level cells.

Press "control f" to hide internals of lower level cells.

Press "f" to fit the view.

Press "Ctrl-r" to redraw after changing selection and visibility of layers.

When "creating interconnect", start with "p" and use "Ctrl-v" and "Shift-v" to switch metal layers through a via up (1 to 2 or 2 to 3) or down (3 to 2 or 2 to 1) respectively.

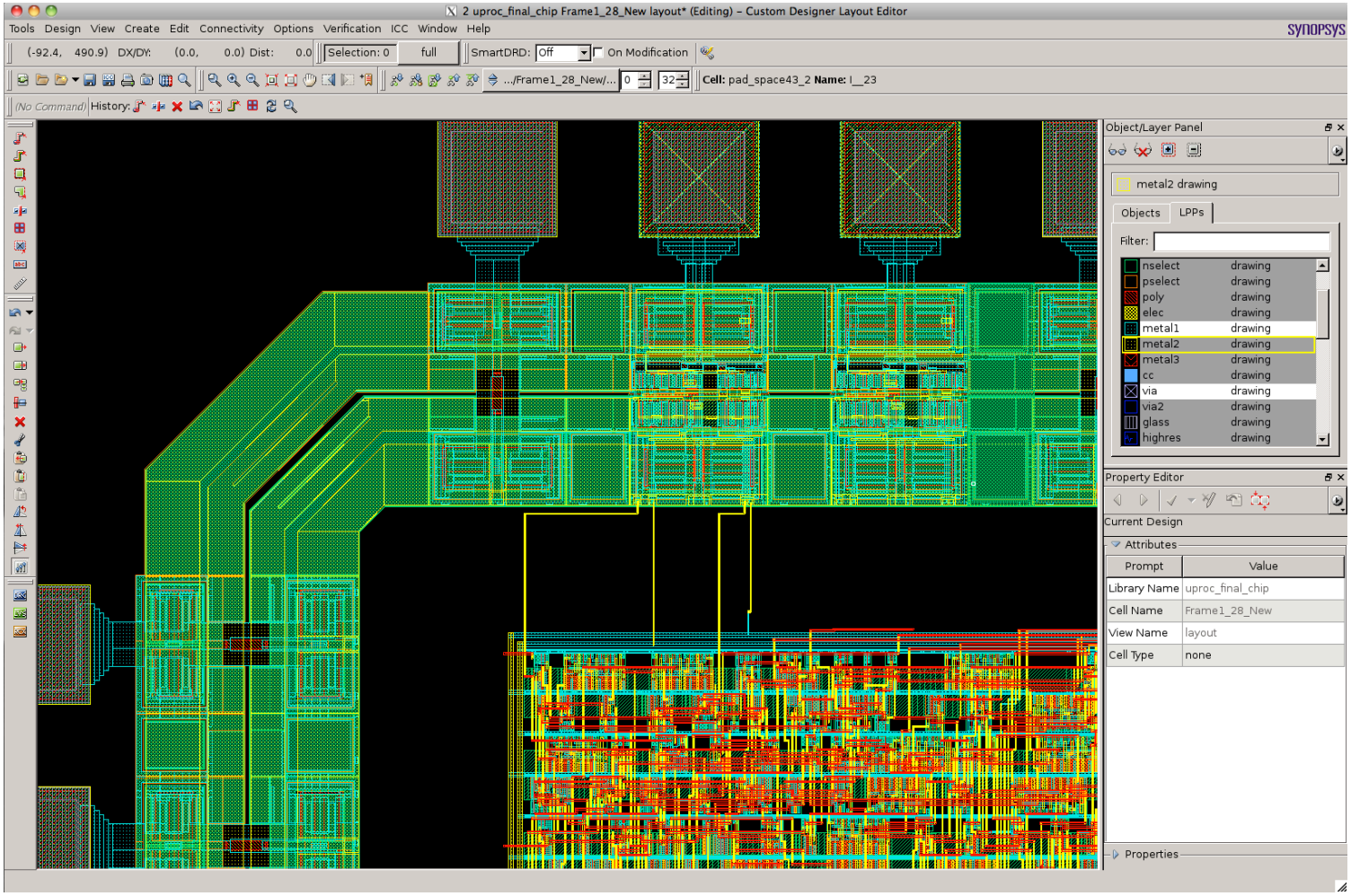
Press "q" to see the properties of an object.

Double click an object to descend the hierarchy and Ctrl-E to return.

Almost everything you will do will be in the metal layers, so turn on visibility for metal 1, 2, 3, via, via 2 and text.

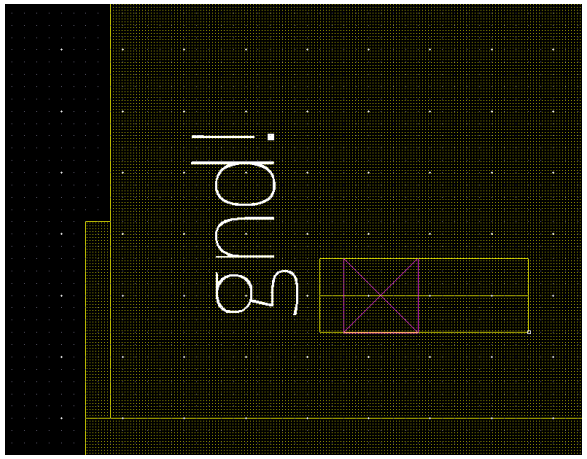
Add connections to layout

"Create Interconnect" or press p to add wires to make IO connections, shift-v and ctrl-v to switch metal layers through a via as you connect from one pin to the other. Here is a picture of some connections between the core and padring.

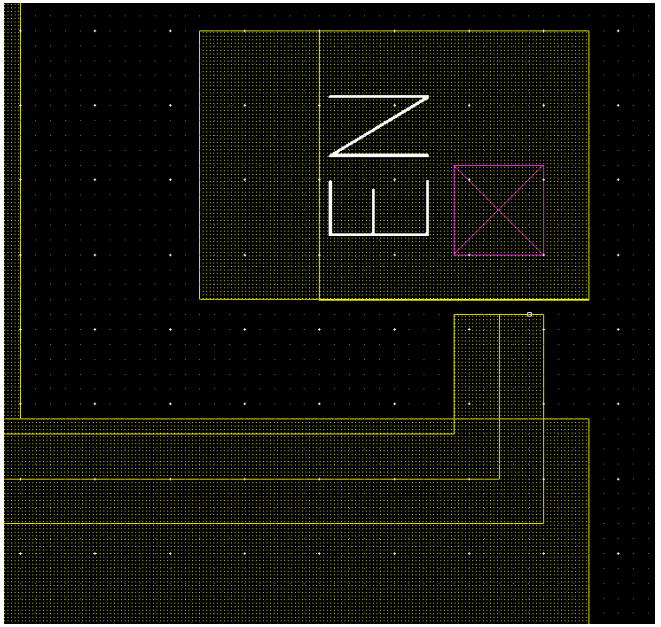


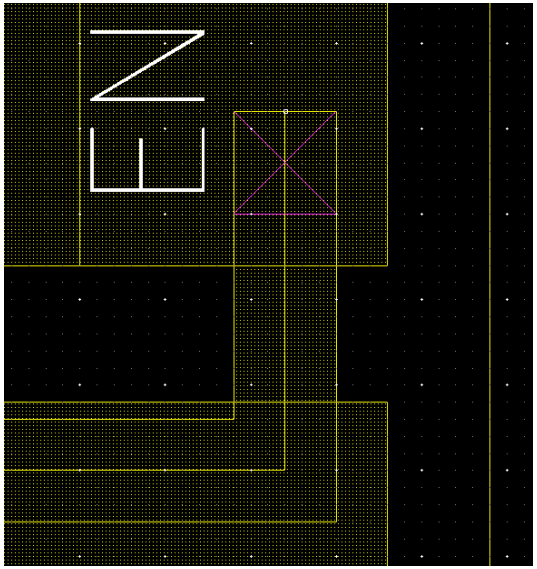
Add all connections to layout -1

1) To connect two pins together, start by pressing “p”, select the correct metal layer in the layer list and single click next to the pin.
I try to cover the entire pin with the new metal.



2) Single click to make a turn.

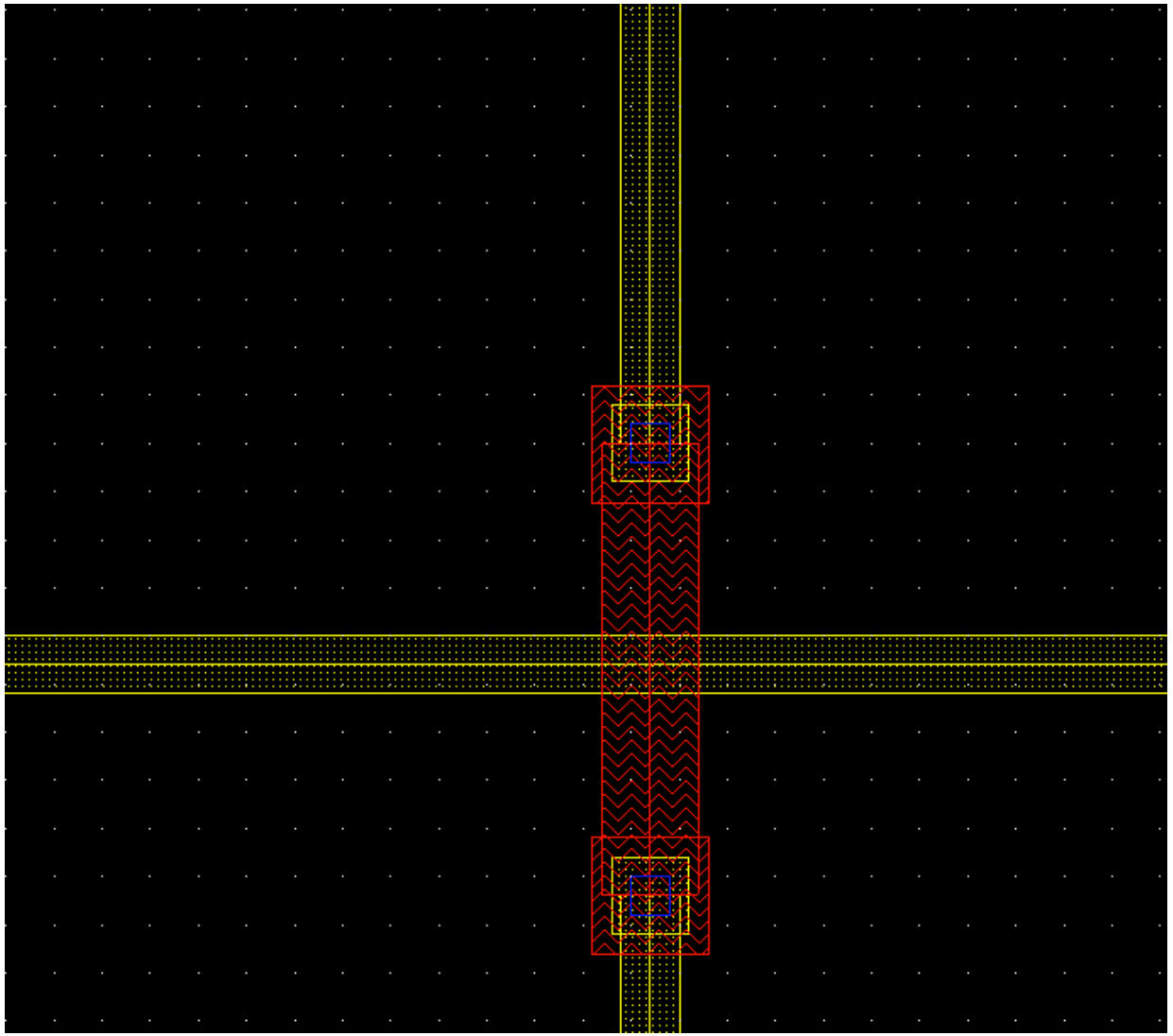




3) Double click or press enter to finish the operation. Cover the entire destination pin.

Add all connections to layout - 2

Use “shift-v” to go up a metal layer or “control-v” to descend a layer in order to avoid a collision with the same metal layer.



Avoid unclean corners which can lead to DRC errors later.

This connection caused a DRC error. Try to make connections “cleanly”. Not double turns or with extra weird structures.

Add all connections to layout

Each IO has an "enable" which should be tied to ground (input) or vdd (outputs). Tie the signal to either a !gnd pin (purple box in the inner IO ring) or !vdd pin (purple box in the second ring from the inside).

1) If IO is an output, tie the "DataOutput" signal to the appropriate pin on your core. Let the two "DataInput" signals float.

2) If IO is an input, tie the "DataInput" signal to the appropriate pin on your core. Tie the "DataOutput" pin to ground. "DataInputN" can float.

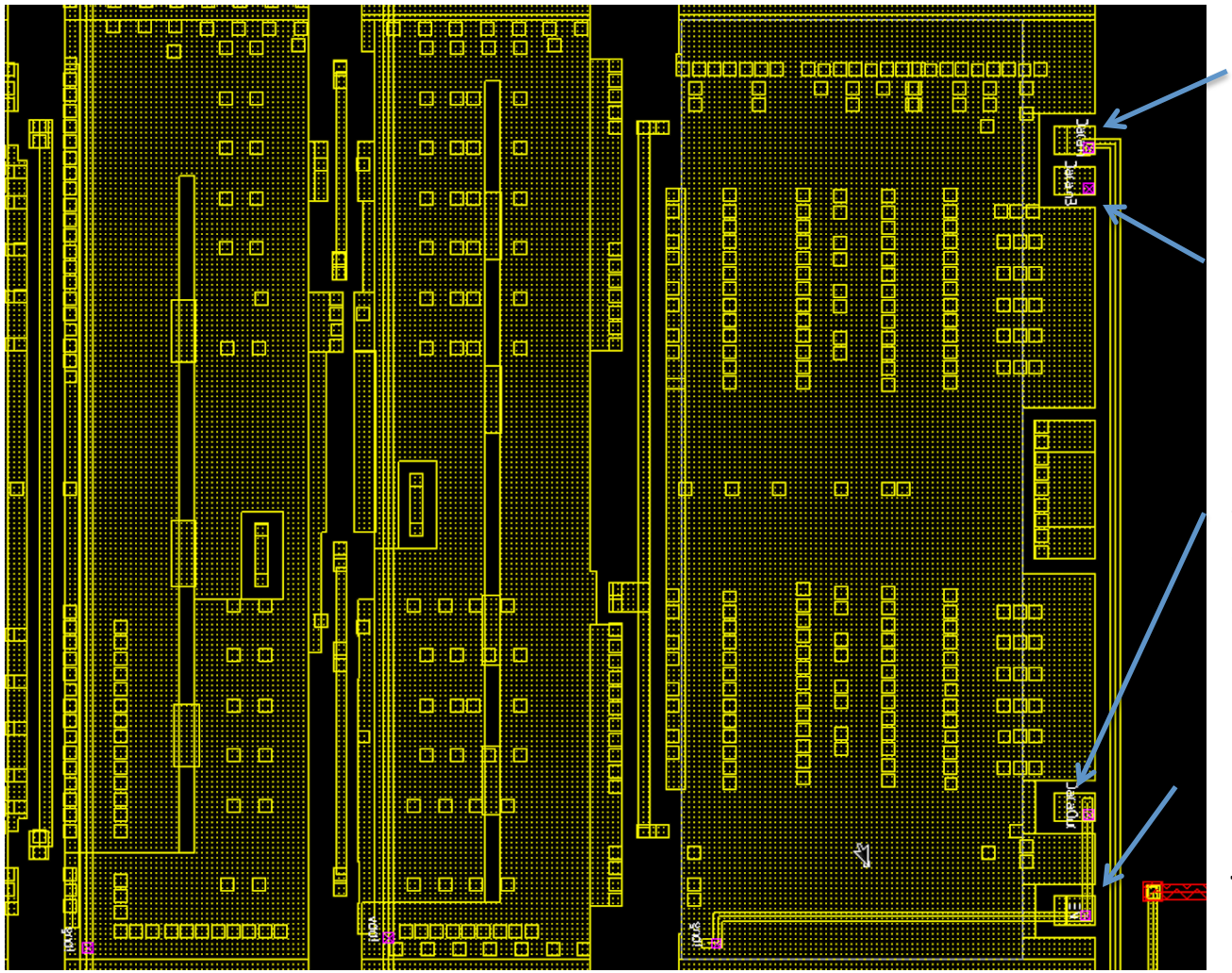
3) If the IO is not used, then ground the "enable" and the DataOutput" as both are inputs to the core and you should never float inputs. Plus, you want to ensure that the drivers are turned off.

On all four sides we need to connect the power ring of the IO to the power ring of the core. We need similar connections between the ground rings as well.

When you press "p" for creating interconnect, a "width" option appears in the menu bar. Use 0.9u for signals and 4.5u for power. This reduces the resistance and the voltage drop when a significant current is being drawn.

Add connections to layout – Input Example

Here a IO is configured as an Input. The input out signal is on the top right and the bottom shows the enable being grounded. Only metal 2 is shown. The purple boxes are metal 2 pin that designate an intentional connection.



DataInput, Output of IO metal 2, used when enable is tied low as in this case. Float otherwise.

DataInputN, Output of IO metal 2. Not generally used and left floating.

DataOutput, tied low for safety. Not used in this case. Input to core so it cannot float.

Enable, input to IO metal 2, tie high for Output Tie low for Input. Tied low here. See connection to purple ground pin

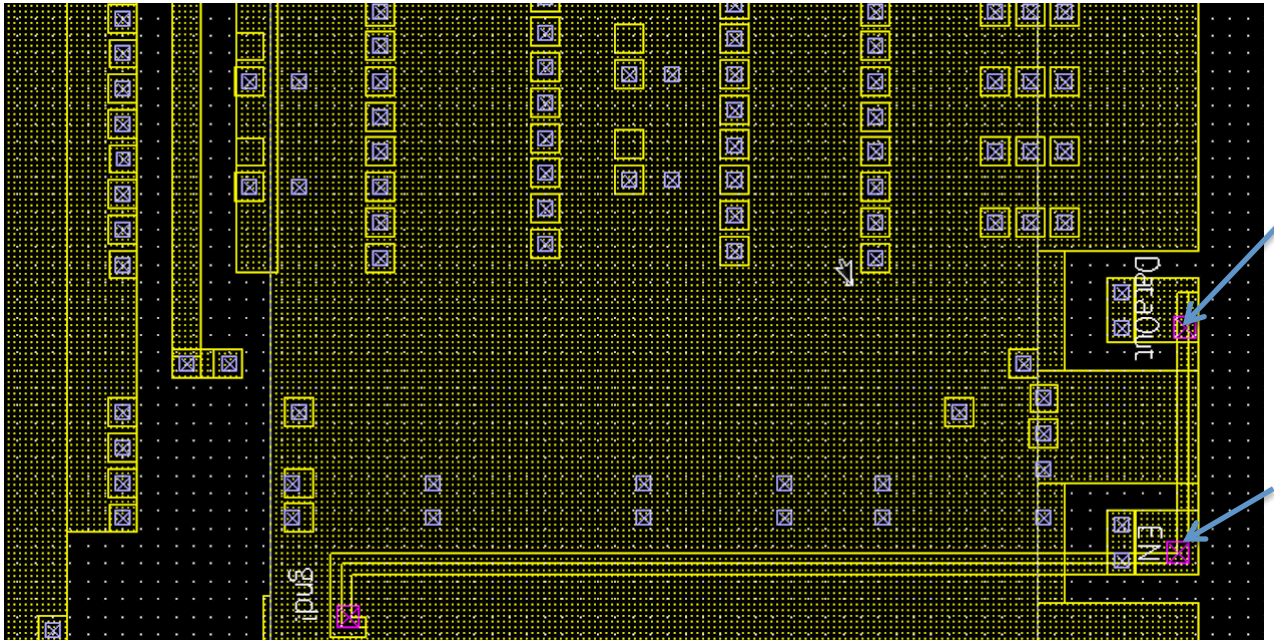
Add connections to layout – Unused Example

Here a IO is configured as an Input and not used. DataInputs are not connected and both DataOutput and Enable are grounded to provide an unused input that will not have floating input problems.

No IO on your padring should have a floating Enable or DataOutput pin as these are inputs and must be provided with valid signals. This is how all unused pins should appear.

DataInput, Output of IO
Floating for unused case..

DataInputN, Output of IO
Floating for unused case.



DataOutput, input to IO
metal 2, used when
enable is high.

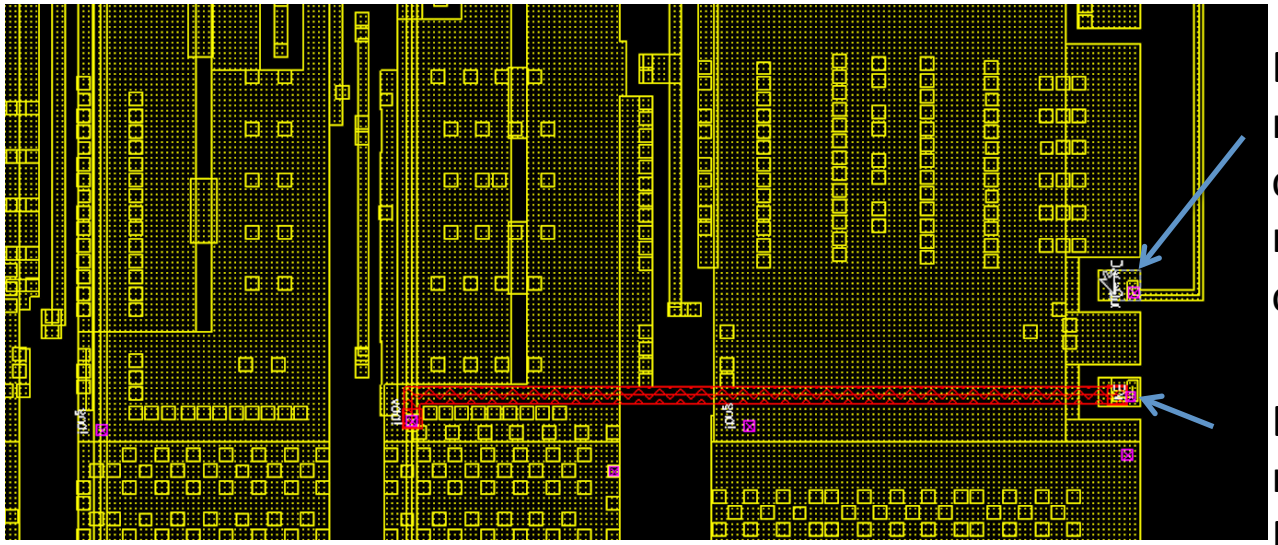
Enable, input to IO
metal 2, tie high for Output
Tie low for Input or **unused**
as in this case

Add connections to layout – Output Example

Here a IO is configured as an Output. The input out signals are not shown and are not connected as they are unused outputs from the cell. The purple boxes are metal 2 pin that designate an intentional connection.

DataInput, Output of IO metal 2, Not connected in the case of output.

DataInputN, Output of IO metal 2, Not connected in the case of output

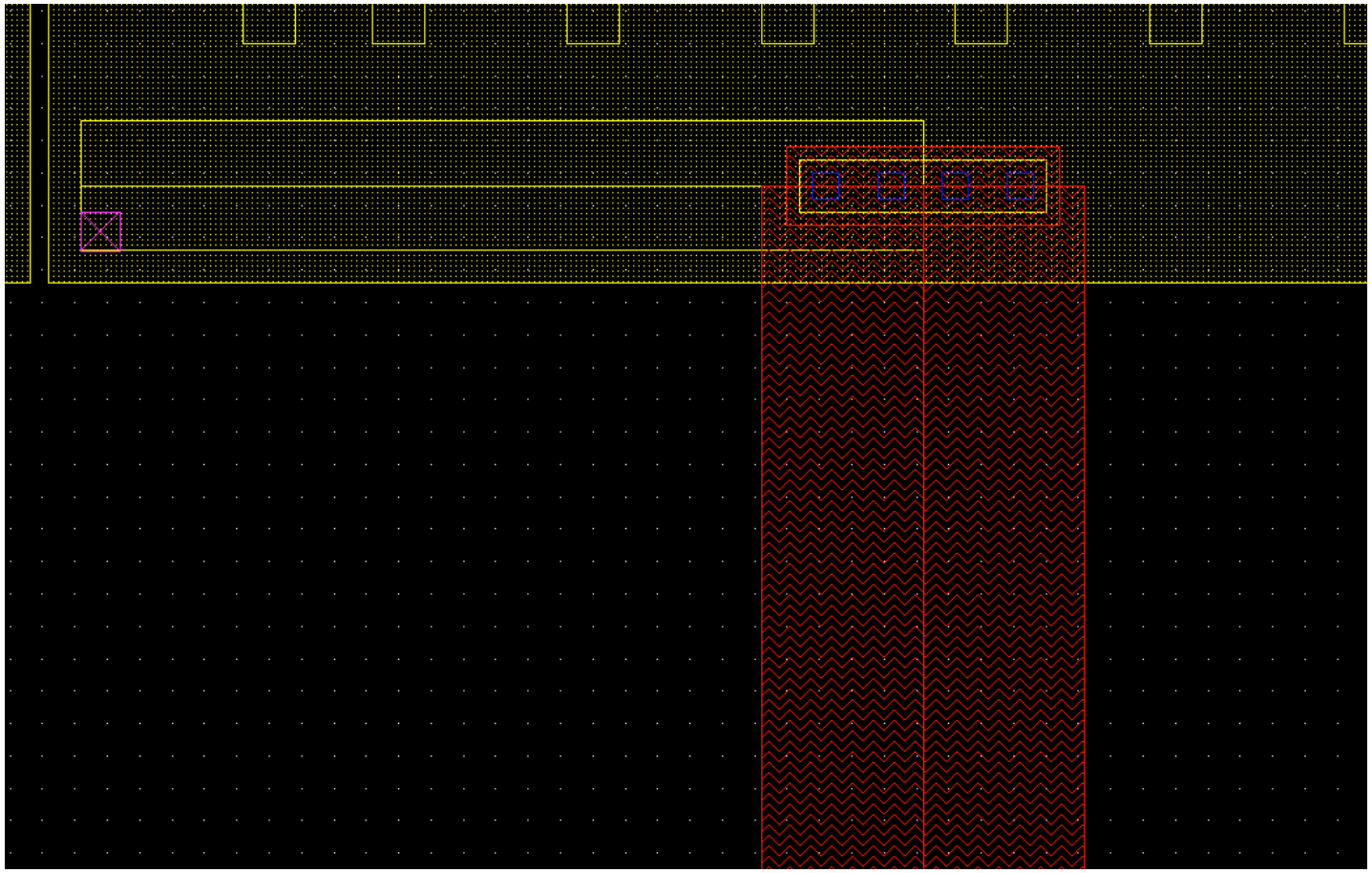


DataOutput, input to IO metal 2, connected to core. This is input so must be driven by core output.

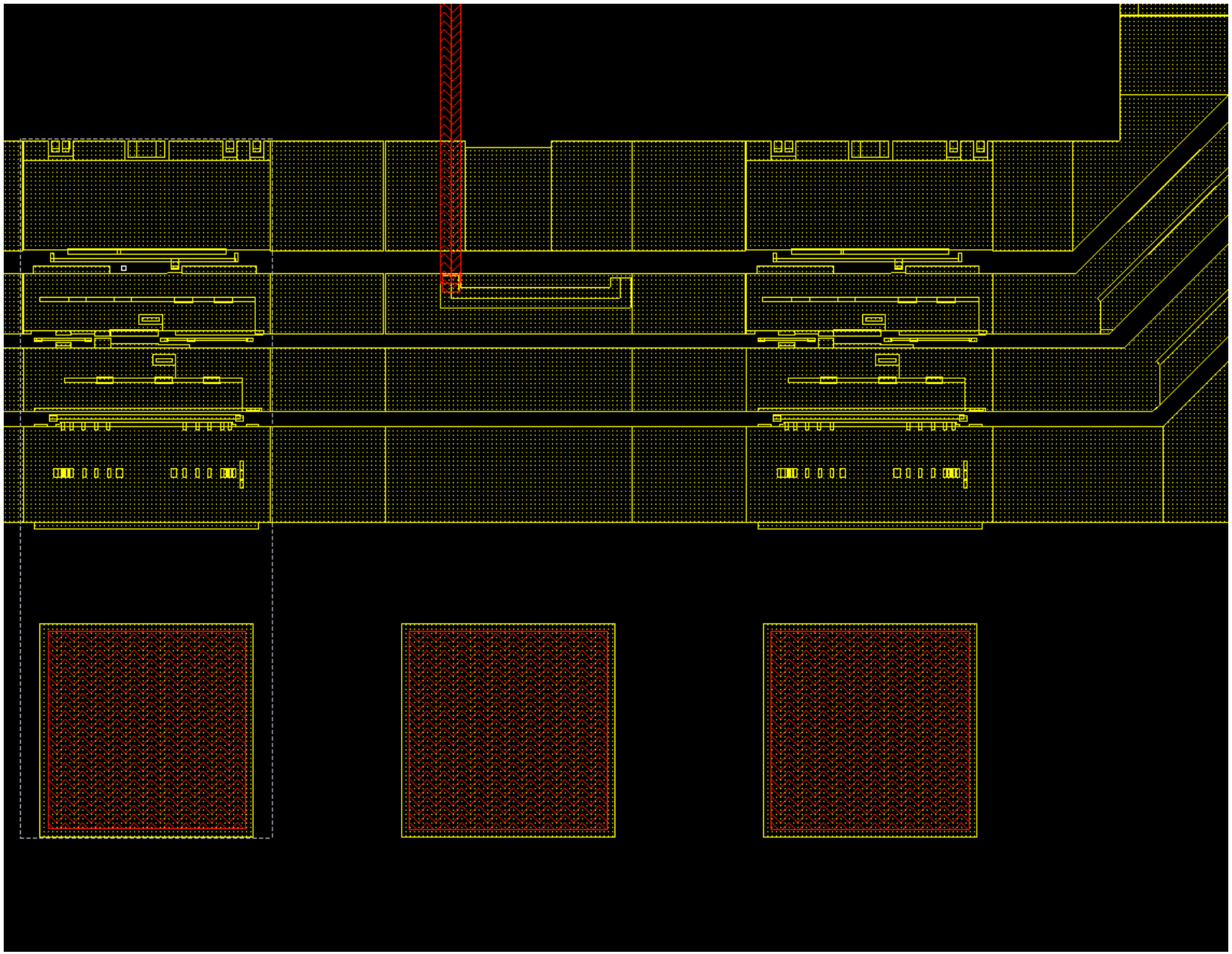
Enable, input to IO metal 2, tie high for Output. Requires metal 3 to "jump" over metal 2 ground ring and connect to vdd ring.

Add connections to layout – Connecting the ground rings

Use a thick metal 2 line between ground in a inner pad ring of your core to the inner ring of your chip to connect ground. Do this on all four sides for robustness. Experiment with the width field in the menu bar as you create this bus. I had success with 4.5u which matched the power rings of my core.

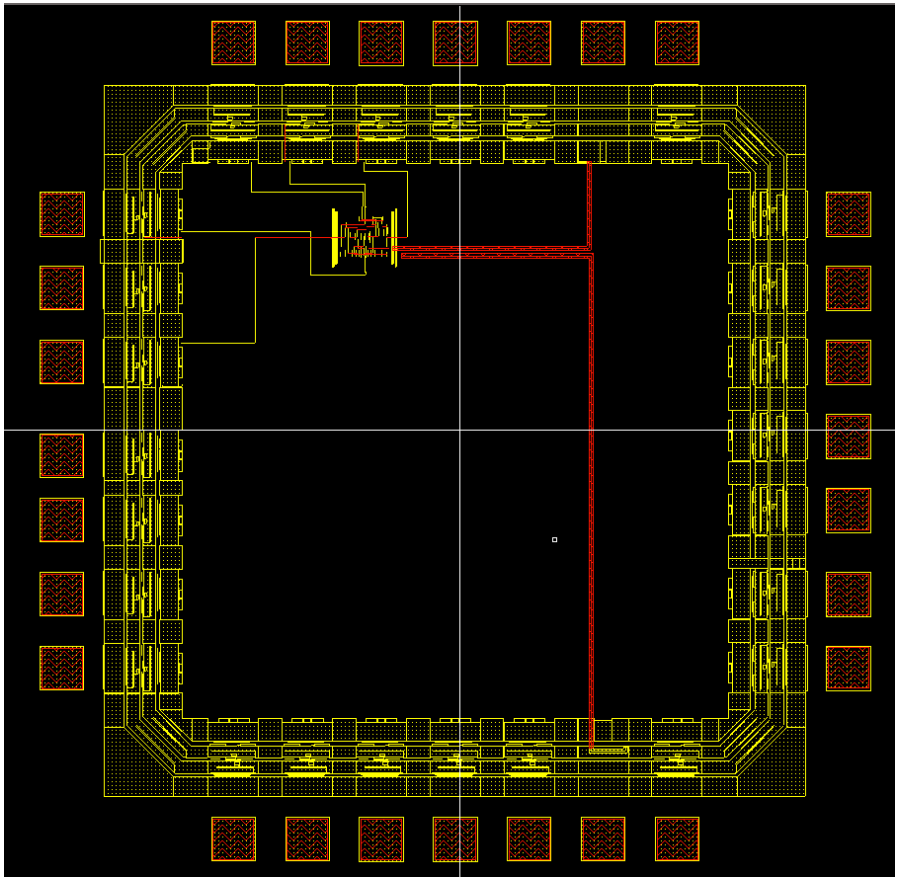


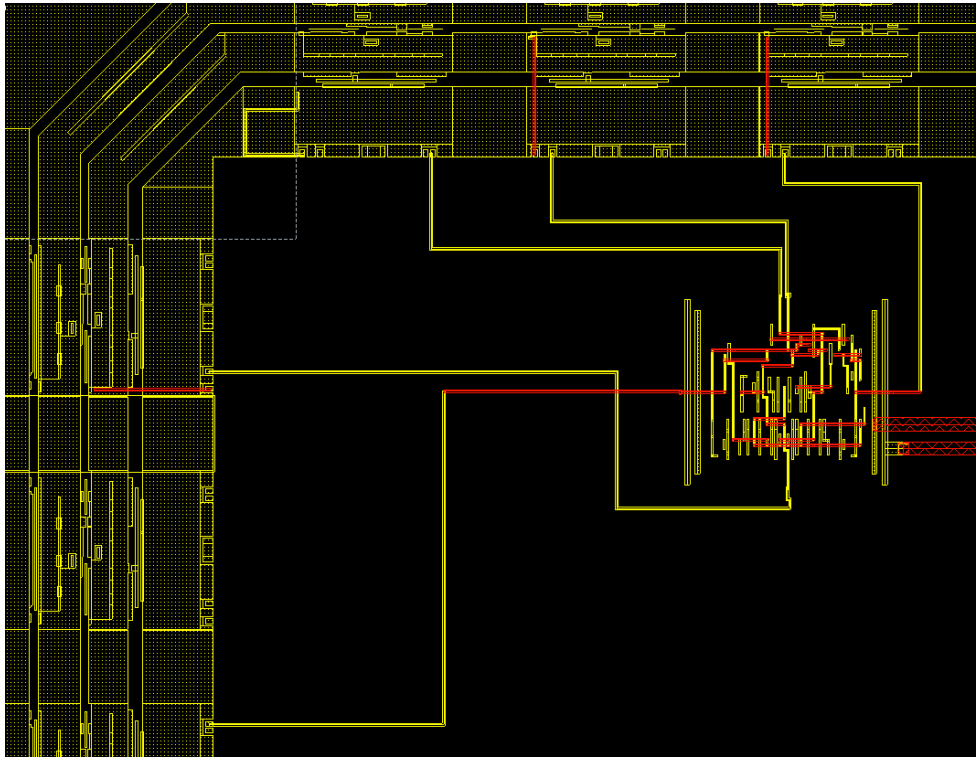
Add connections to layout – Connecting the Vdd rings



Add connections to layout

Example of all connections with Metals 2 and 3. In this example I only had one ground and power connection as this is a tiny circuit. Please use more – preferably one per side for ground and Vdd. The core in this example is very small.

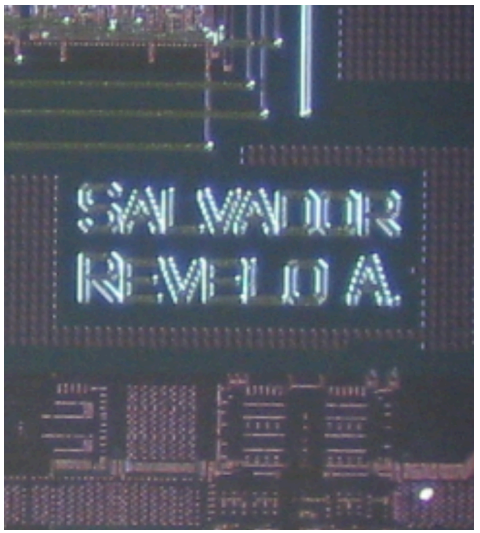


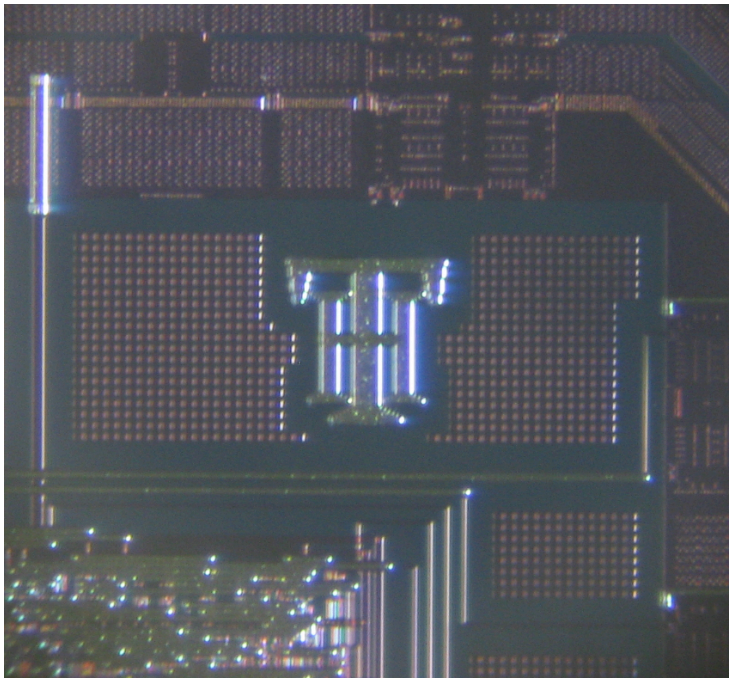


Silicon Art

Keep in mind it is possible to put artwork in the metal 3 layer as floating metal that spells words or draws pictures. Do this but I would recommend doing it after completing all DRC and LVS checks.

Pictures often introduce DRC problems and it is easier to fix on a clean chip.





Now complete the chip layout.

Edit your chip Schematic

In your chip level schematic view, you will see the 28 pin pad frame. You need to instantiate you core in the middle and connect to the padframe.

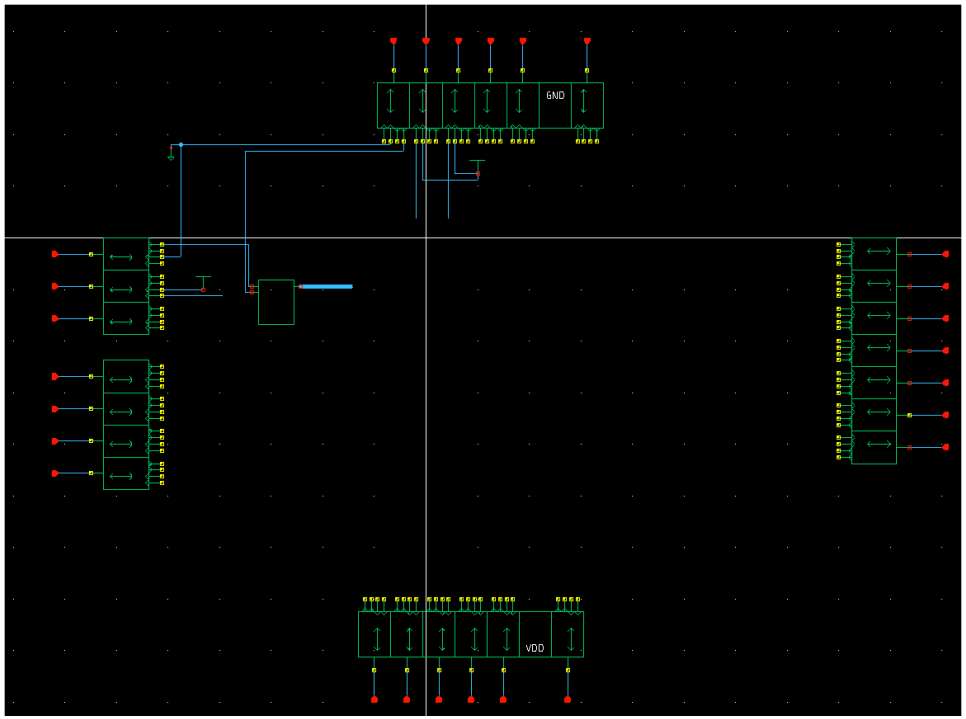
"I" will bring up the instantiation window, "w" will start to draw wires, "L" to add wire name. You can make all connections through labeling. This approach is less likely to introduce errors through making wrong connections.

Instantiate gnd and vdd symbols from the UofU_Analog_Parts library.

The connections in the schematic should match the layout for LVS later.

Save and check the design in the end.

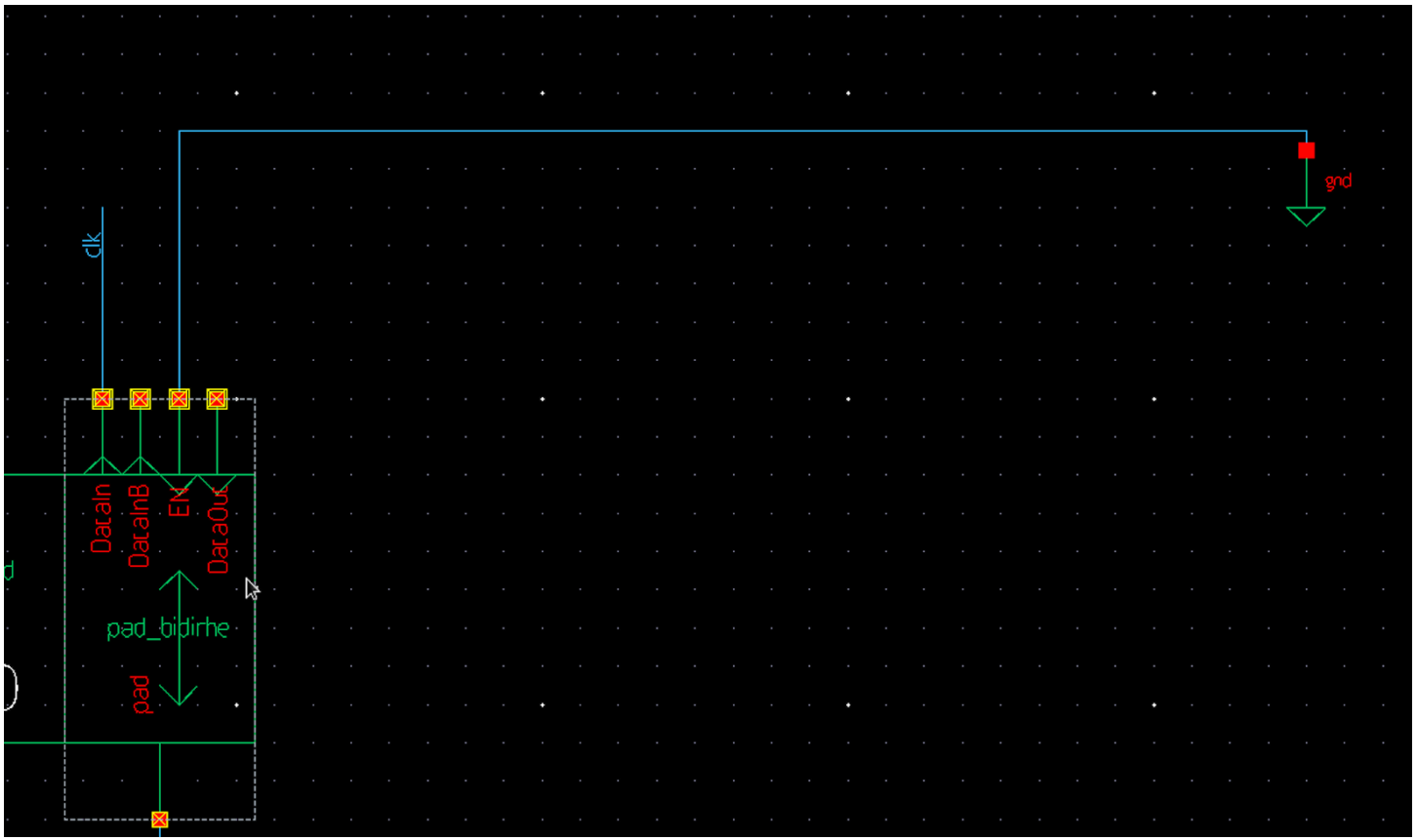
Read the next three slides before beginning to start the schematic entry.



Edit your chip Schematic

Example of an input.

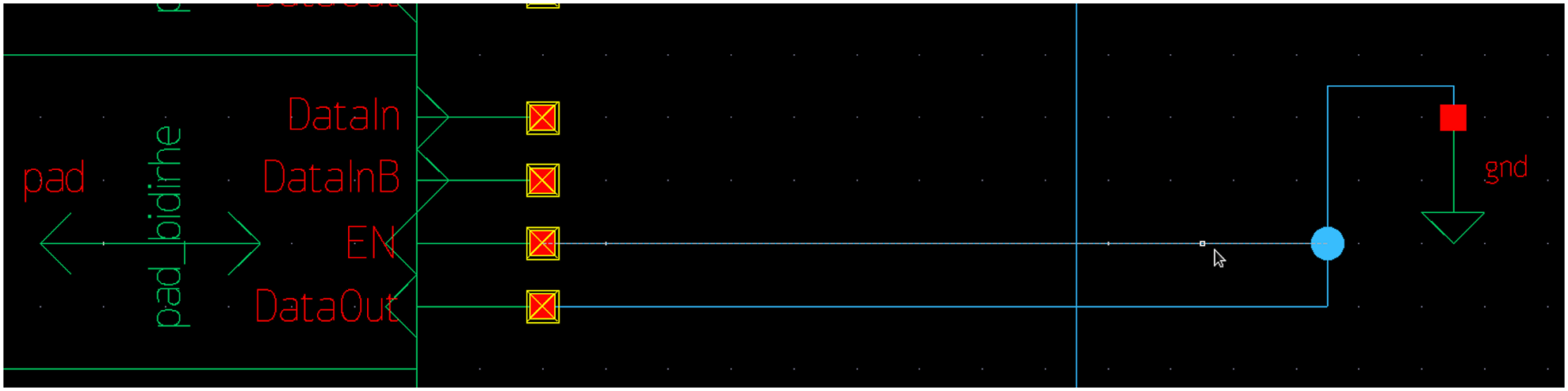
Identical connections to the layout. “clk” is a core input connection. Here it is an output of the IO. The enable is grounded so the IO acts like a input.



Edit your chip Schematic

Example of an unused IO.

Identical connections to the layout. “clk” is a core input connection. Here it is an output of the IO. The enable is grounded so the IO acts like a input.

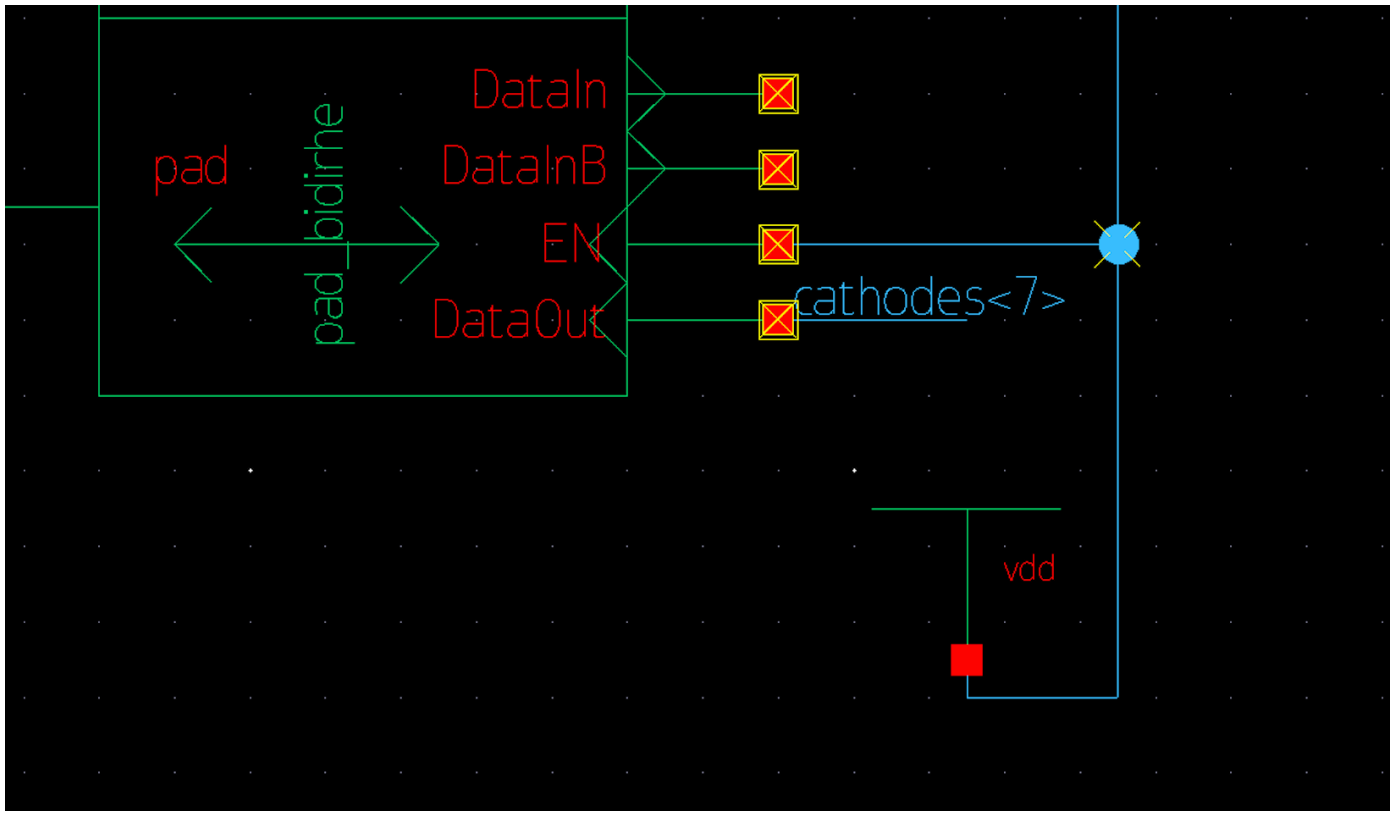


Edit your chip Schematic

Example of an output.

Identical connections to the layout. “cathodes<7>” is a core output connection. Here it is an input of the IO. The enable is connected high so the IO acts like a output.

Now add the core to your schematic and complete all connections.



Now complete the chip schematics.

RunDRC, extraction and LVS

Currently, MOSIS only supports the LVS and DRC tools from Cadence and Mentor. So we need to read the design library into Cadence's virtuoso.

For Chilean students:
If you don't have Cadence, go to your working directory and you should see a sub-directory that has the same name as your library. Use the following command to create a zip file to send to UTEP.

tar –cvf your_name_library_name.tar library_name
gzip your_name_library_name.tar

Export Final GDS to be Fabricated

To export design data in GSDII format:
From the Console, choose File > Export > Stream.
In Main tab

Specify the output: Enter Run Directory and GDS file in Stream file name.

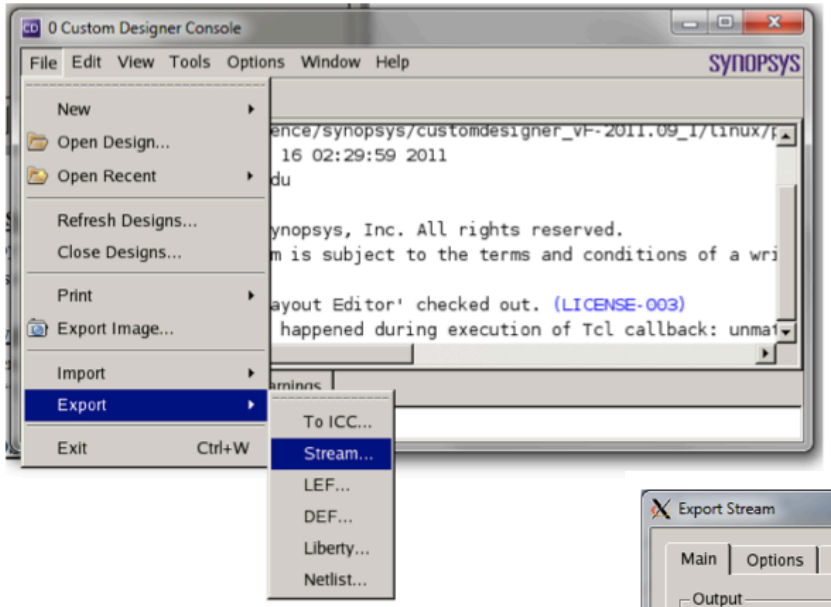
Specify the input: Browse the library and cell that you want to export and leave the view as layout. Select the cell that includes the pad ring and core connected.

Choose the Options tab and specify option details. We can change the options I left it default.
Choose the Map Files tab and specify mapfile details: Browse the layer map file.

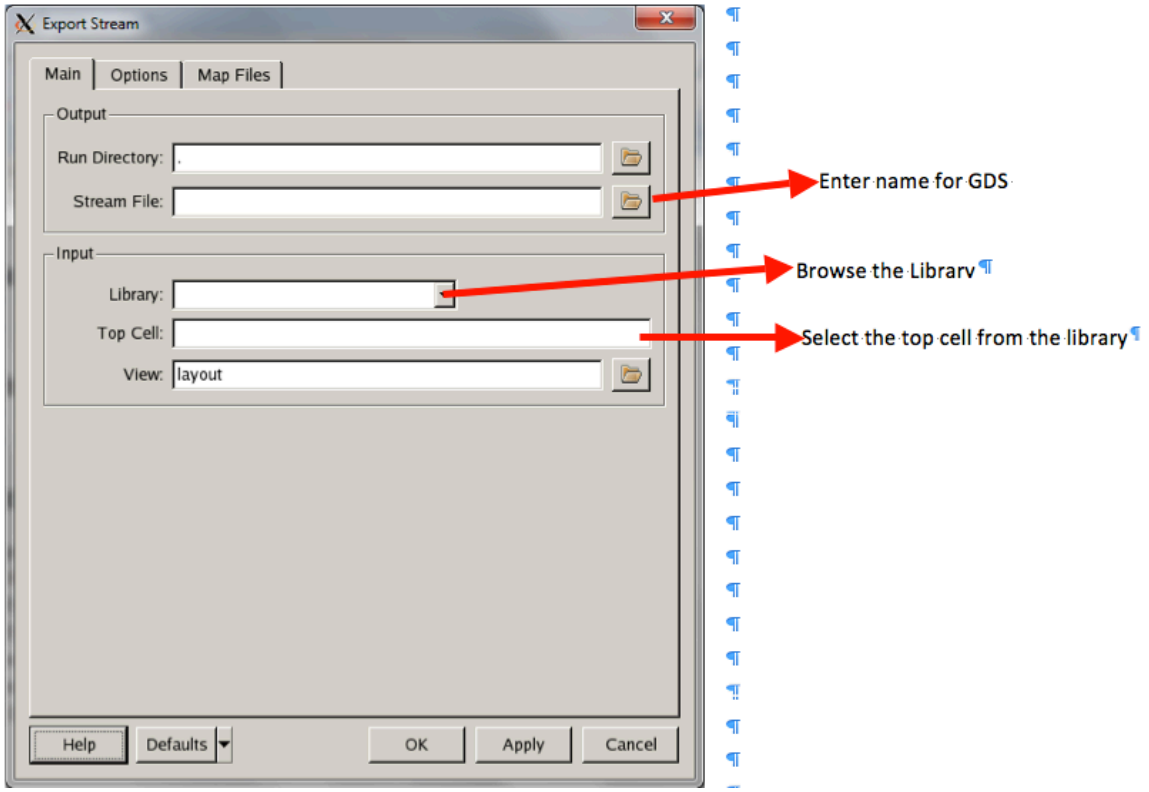
The layer map should be different than the one used to read your core. CD2GDS.layermap

Click OK.

Before these files can be submitted, you must run DRC, Extract and LVS with Cadence Virtuoso and then run a density check using Assura (through Virtuoso as well).



CD2GDS.layermap



Final Final Check

If you have problems with MOSIS submission, you may want to duplicate your Custom Designer library (add "_gds" to the end of the library name). Erase only the layout view of the top chip cell and then re-import your GDS file (to be sent to MOSIS) in as layout. You will need to specify the GDS2CD.layermap for this. You should be able to then re-run DRC, extraction and LVS in Virtuoso. If this passes, your GDS file is has the correct layers and matches your schematic.

Final Final Final Check

Although our designs are too large (too many transistors) to run Spice simulations of the entire chip, it is possible to run mixed-mode simulations (part Spice and part Verilog). Having a simple simulation that proves all core to padring connections are correct would be an additional validation. More on this later.