

Virtuoso Layout Editor

This tutorial will cover the basic steps involved in using the Cadence layout editor called Virtuoso, extracting layout, and running simulation on the created layout. The inverter layout is used as an example in the tutorial. This document is supposed to be a general overview of the tool and more specifics can be found under *cdsdoc*. To launch cadence documentations application, type '*cdsdoc*' at the command prompt.

Before start, you should have necessary files and setup done to be able to run Cadence software. If you have not done this, see tutorial on "How to setup Cadence tools?" (Available on class website)

Create New Layout View

To create a layout view, go to **File -> New -> Cell View**, select the **Virtuoso** tool in the tool selection menu and type in Cell Name as shown in Figure 1. Click OK, two windows will pop up; a layout window and an LSW window. The layout window is the main window where you do your design layout. The LSW or the layer selection window gives you a list of all available layers in the current technology. They will include layers like poly, nactive, pactive, nselect, pselect, metall, cc, via etc. depending on the technology you are fabricating in.

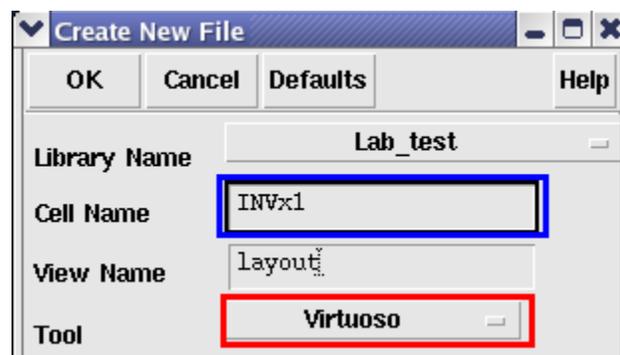


Figure 1: New Cell View

Setup Display Options

To start the layout the first thing that you need to do is fix you grid sizing. Go to **Options -> Display**, the options window shown in Figure 2 will pop up. Set the grid control parameters in the right hand upper corner to the following values.

Minor Spacing = λ
 Major Spacing = 5λ
 X Snap Spacing = $\lambda/2$
 Y Snap Spacing = $\lambda/2$

Remember λ is half the feature size so if you are using a 0.6 μm . process then the λ value is 0.3. Also make sure that the **Snap Modes Create and Edit** settings in the right bottom corner are set to **anyAngle**. Set the **Display Levels** to **Start = 0** and **Stop = 20**. Select Library on the bottom and click **Save To**. Once you save the options to the library, you do not have to set up the display options again when working in the same library. Click **OK** and then go the **Window -> redraw** to update new setting on the screen. If you are working in **NCSU_TechLib_ami06** attached library, you can simply follow the setup exactly as shown in Figure 2.

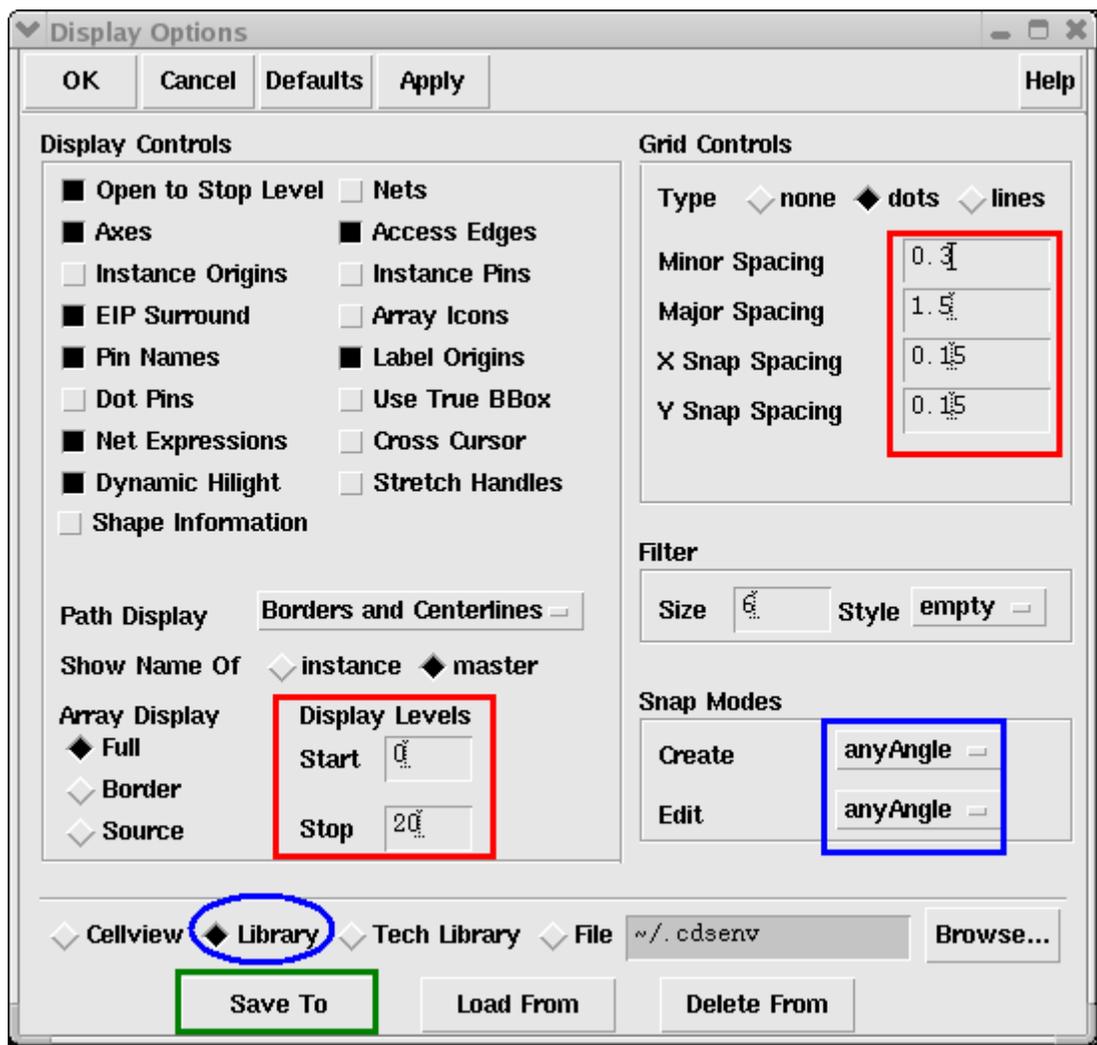


Figure 2: Display Options

Getting to know Virtuoso Menu

Tools Design Window Create Edit Verify Connectivity Options Routing NCSU

The following menu items are frequently used during creating circuit layout.

Design Menu:

- *Save*: save you layouts.
- *Plot*: Used to generate a postscript file for your layout for printing. Select submit and then plot options. Specify a file name and click ok. Then click ok on the main submit plot form and it will generate a ps file for you layout.

Create Menu:

- *Rectangle*: creates a rectangle of the layer selected in the LSW.
- *Path*: Creates a path of the layer selected in the LSW. Double click to end the path.
- *Instance*: used to import another existing cell view into this cell view.
- *Pin*: Create pins as explained in later section.

Edit Menu:

- *Undo*: undo the previous commands.
- *Redo*: redo undone commands.
- *Move*: click on any object and move it around in the layout.
- *Copy*: Create a copy of any object in the layout.
- *Stretch*: Click on the edge of a rectangle and size it.
- *Delete*: Delete an object in the layout.
- *Properties*: Change the properties of objects in the layout. Change the layer definitions and the changes are immediately reflected in the layout.

Verify Menu:

- *DRC*: Check the layout for design rule violations.
- *Extract*: Create a extracted view of the layout. This view is used for simulations.
- *Markers: Explain*: click on the marker to find out the design rule violated.
- *Delete all*: Remove the markers after a DRC run.

Drawing Transistors

To make an inverter, first of all you need to make p and n transistors. First we will make the p transistor. Most of the technologies you will be designing in will use an n-well process. Therefore the black background in the main layout window will act as your p-substrate. Thus you can put n transistors directly in the p substrate. However your p transistor will have to be placed in n-well that you will have to draw specifically. Figure 3 shows the different layers required to build a p transistor.

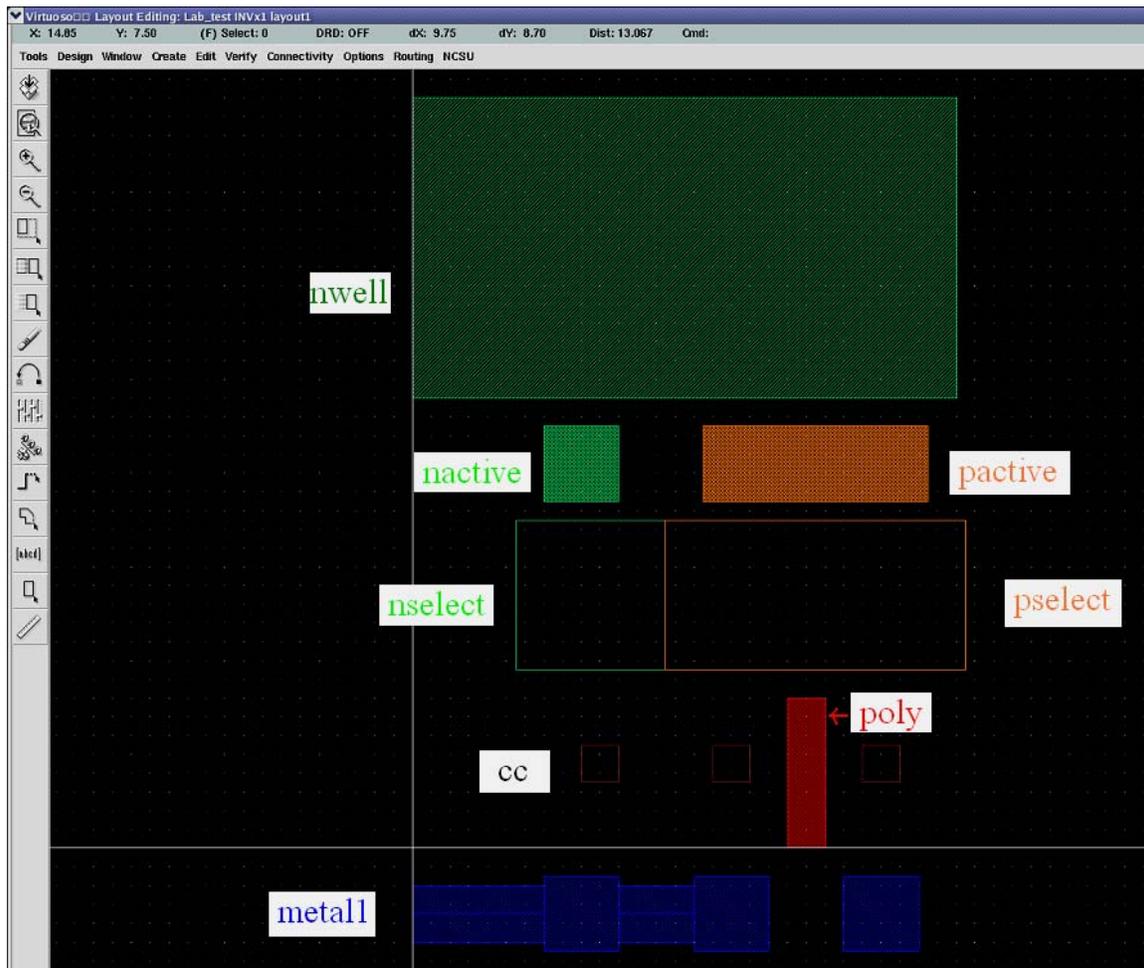


Figure 3: Layers for building PMOS

The p transistor is made by using the layers as shown in the figure above. You can create each shape shown in Figure 3 by first select layer you want to create on LSW window and use create rectangle command (**Create -> Rectangle**) or create path command (**Create->Path**). The size of the layers depends on the DRC rules defined for the library and they can be accessed at the website: www.mosis.org. Go to technical support and then to mosis design rules and select the technology that you are using.

The p transistor is placed in the nwell and a n contact is placed in the nwell to connect it to Vdd. The layers needed to make the p transistor are cc, metall, pactive, poly and pselect. The pselect region should cover the entire active area and the poly gate. The nwell should surround the entire ptransistor along with the select region. Now you need an ncontact to connect the nwell to Vdd. The nwell is made up of cc, nactive and metall surrounded by a nactive rectangle. The contact is placed inside the nwell and then connected to the main Vdd supply rail in the design. The layers shown in the figure above are merged together as shown in Figure 4 to form the entire p transistor.

The order in which the layers are placed is not important the only requirement is that all the layers should be present. The layers are displayed in the layout window as defined by the tool and the overlap of two different layers will be clearly distinguishable.

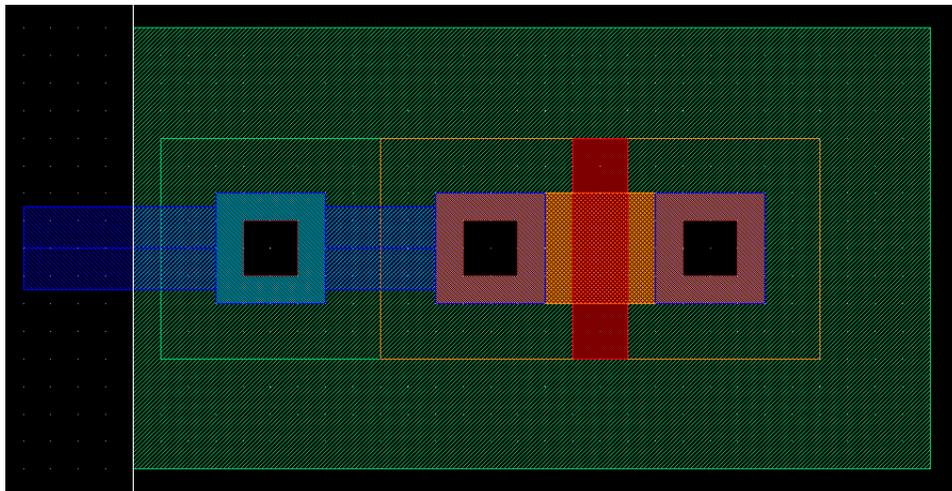


Figure 4: PMOS Layout

The n transistor is laid out in a similar fashion and the layers required for the same are shown in Figure 5. However the n transistor will have an nactive layer instead of pactive layer in the transistor and will be surrounded by n select instead of the p select. As the technology is an nwell technology the black background is the p-substrate and therefore you need not put a pwell layer around the n transistor. However you still need to place a contact to connect the p substrate to ground. The p contact is complementary to the n contact and is made of cc, pactive and metal1 surrounded by a pselect rectangle. The combined p and n transistor are shown in Figure 6.

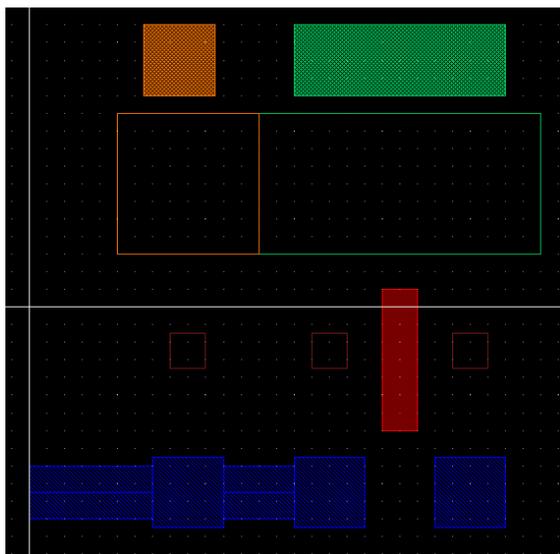


Figure 5: Layers for building NMOS

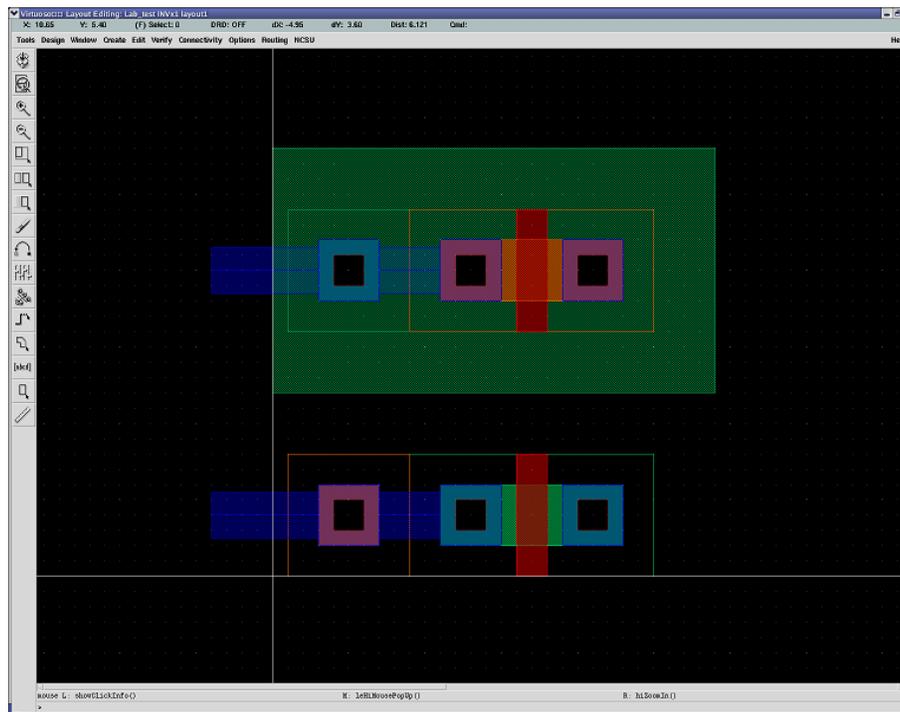


Figure 6: PMOS and NMOS layout

Design Rule Check

There is a rule checking tool that is available with the cadence distribution which checks most of the rules. To run design rule check (DRC), go to **Verify -> DRC**. The DRC dialog as shown in Figure 7 should appear, click OK. Figure 8 shows DRC report displayed on CIW window, in the figure no errors are found on the design. You need to run the design rule checker at every step of you design. It is very difficult to fix up the sizes of various components after they are connected together in the layout so make sure that everything that you place in the layout is compatible with the design rules. Again the design rule for a particular technology is available on mosis website. (www.mosis.org)

Most of the errors found by the design rule checker are explained in detail in the CIW window and are easy to fix. Sometimes a layout may not have DRC errors but only warnings. It is better to solve these warnings before going further. The most common warning is for pins. When you place two pins on the same path the DRC tool will generate a warning marker but will not explain it in the CIW window.

To check which marker stands for which error click on Verify - Markers - Explain and then click on any marker in the layout window. A text box will pop up specifying the design rule that is being violated. Fix the error in your layout and run DRC until your design is completely error free.

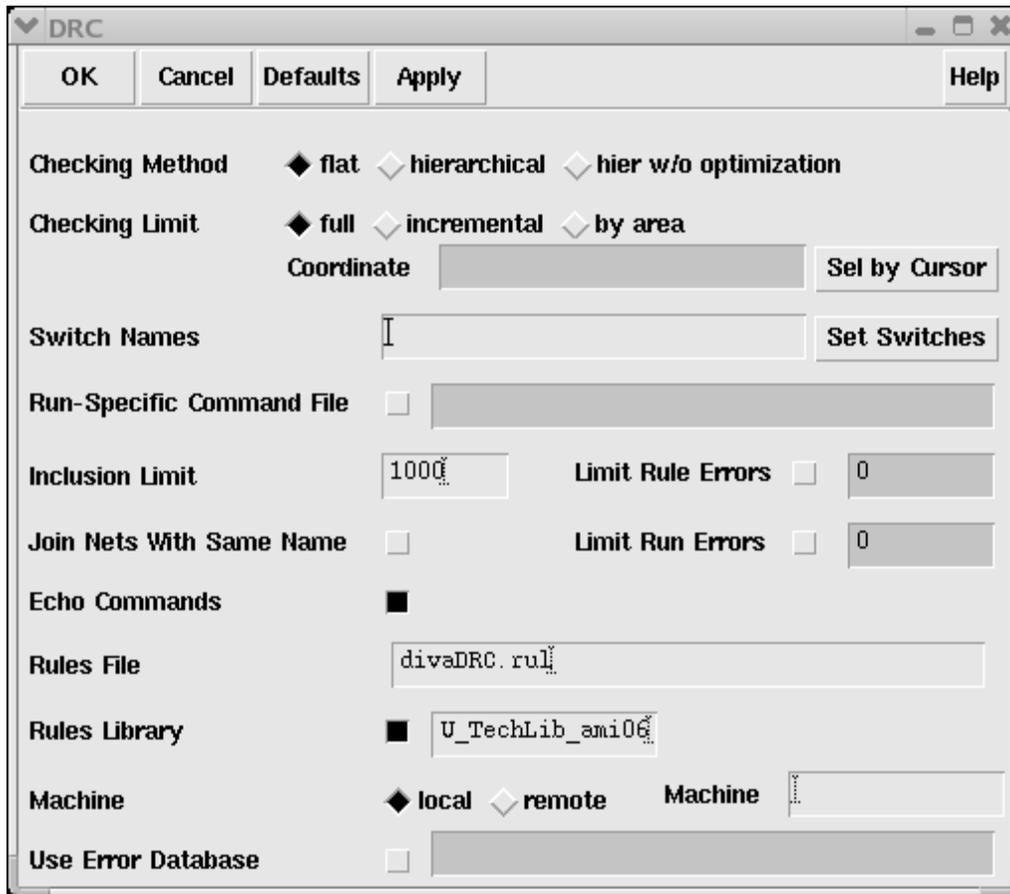


Figure 7: Design Rule Check (DRC)

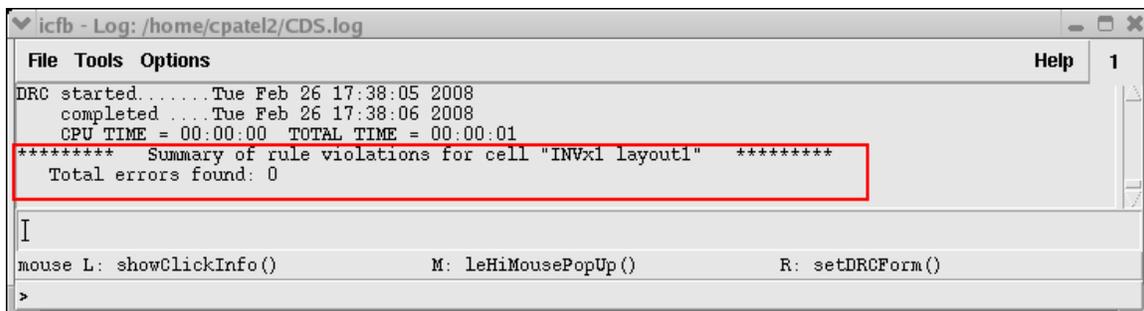


Figure 8: DRC report on CIW

Making Connection

The next step to build the inverter is to connect the drains of the two transistors together to make up the output. The source of the p transistor is connected to Vdd and the source of the n transistor is connected to Gnd. The two gates are connected together to form the input of the inverter. The Path command (**Create -> Path**) is the easiest way to connect components together in a layout. There are rules that define the minimum spacing required between two metal paths and so on which are listed on the mosis site. Figure 9 shows the entire layout for the inverter.

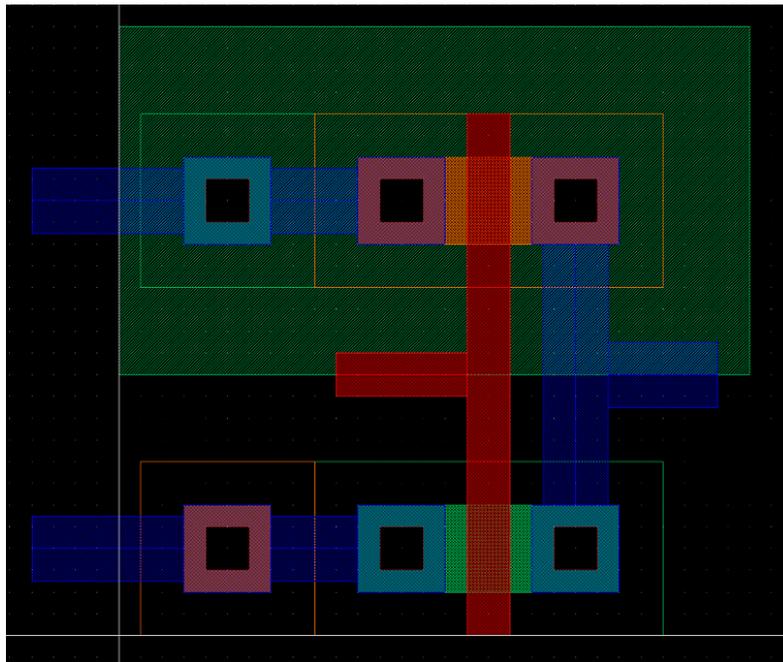


Figure 9: Inverter Layout



Tips:

- Using following shortcut keys help working with layout faster
 - **r** to Create Rectangle
 - **m** to Move
 - **s** to Stretch
 - **p** to Create Path
 - **u** to Undo
 - **c** to Copy
- To zoom in on the schematic, hold **right-bottom** mouse to create zooming area and release to zoom in.
- To zoom out, press **Shift+z**
- To fit whole schematic on screen, press **f**

Create Pins

To complete your layout you need to place input or output pins at the various inputs and outputs of your circuit. To create pin, go to **Create -> Pin**, the dialog box as shown in Figure 10 should show up. You should see that at the top of dialog says Symbolic Pin, you need to select **shape pin** mode as shown in Figure 10. The dialog should then change to **Create Shape Pin** as shown in Figure 11. In the **Terminal Names** box enter the name of the pins that you want to place in a space separated format (e.g. *In Out vdd! gnd!* for the inverter). Next clicks on **Display Pin Name** radio button to turn the option on. Next select the **I/O type** for the pin. This will have to be changed individually for each pin. Thus the pin In in the inverter will have an I/O type of input whereas the pin Out will have I/O type as output, and vdd! and gnd! will have I/O type as input/output. You should always use vdd! and gnd! for supply pins.

To place the pins now select the proper layer from the LSW. Thus in case of the inverter the input pin has to be placed in the poly gates therefore select poly in the LSW. Now draw a small poly rectangle on the path connecting the two gates. The name associated with the pin will be displayed and clicks anywhere near the pin to put the name along side the pin. Now go to the pin menu and select I/O type as output for pin out. Select metal1 from the LSW because the output is the metal1 path connecting the two drains. Create a small rectangle of metal1 on the path and place the pin name beside it. Remember that you have to place the pins in the layout in the order that you specified them in the Terminal names list in the pin dialogue box. Also remember to change the I/O type for each pin and the layer that they are to be placed in.

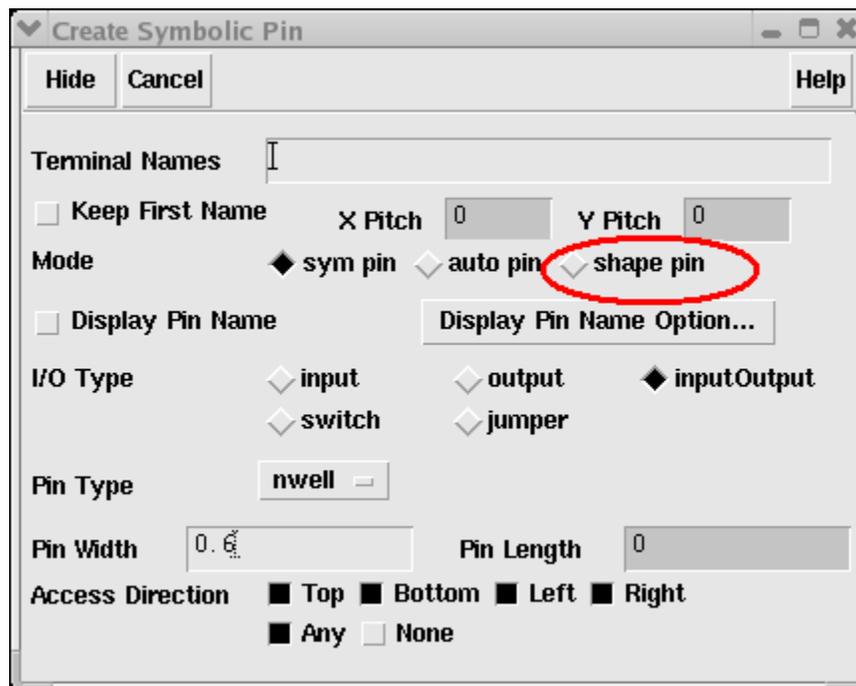


Figure 10: Create Symbolic Pin

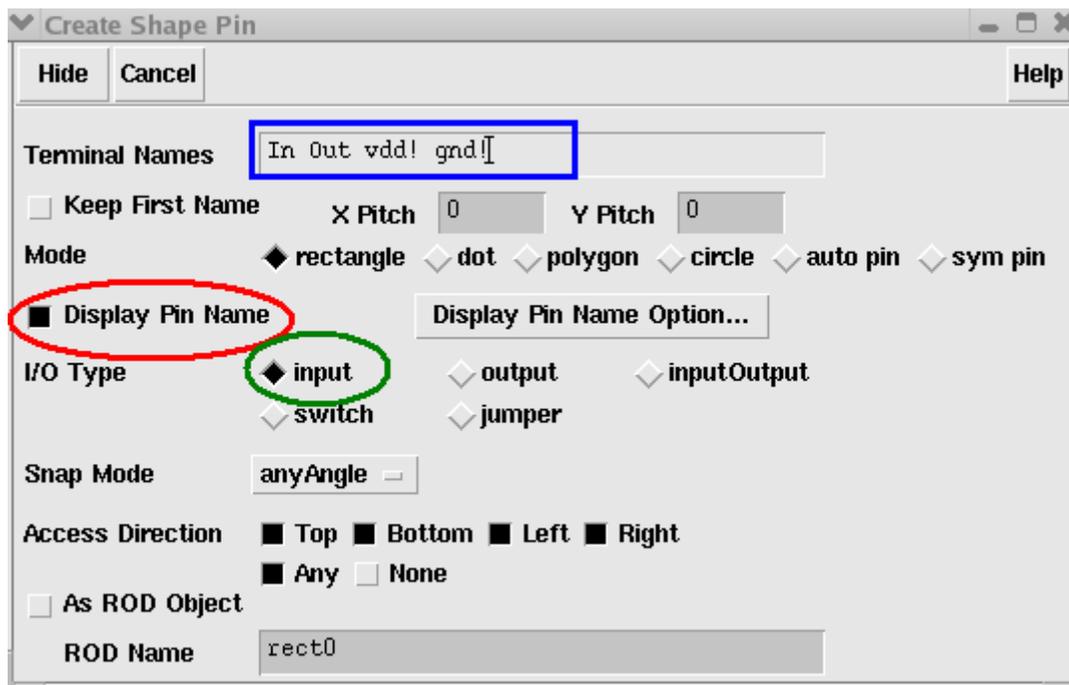


Figure 11: Create Shape Pin

The last step is to change net name for power pins. Click on vdd! pin to select and press **q** (or go to **Edit- > Properties -> Object**). The property dialog as shown in Figure 12 should show up, change **Net Expression Property** to **vdd!** and **Default** to **vdd!** as shown in the figure. Do the same on gnd! pin, except this time change **Net Expression Property** to **gnd!** and **Default** to **gnd!**.

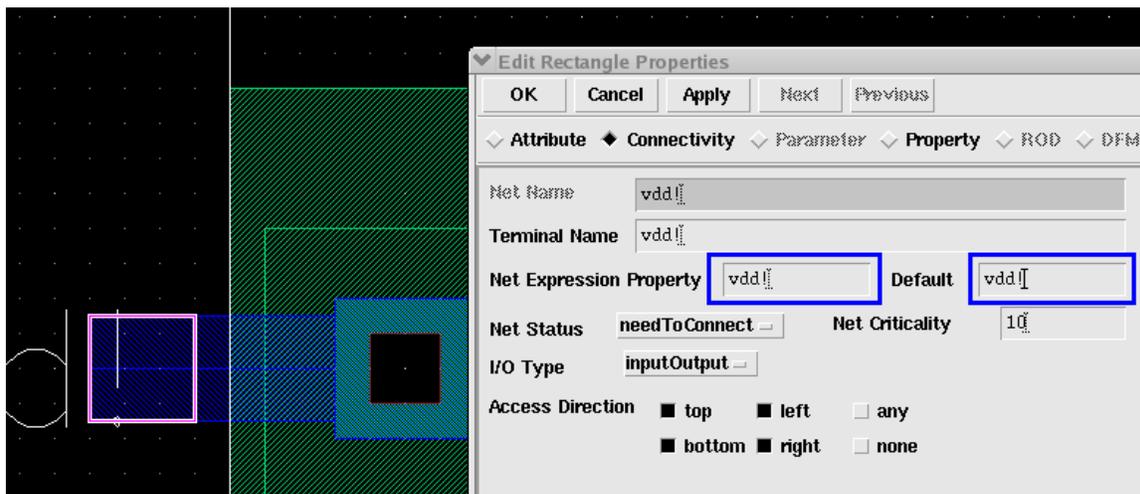


Figure 12: Net Expression for vdd! pin



More info:

- In this tutorial we use **vdd!** and **gnd!** as pin names and net names for power supply to the circuit. This is actually not required by the tool. However, if you have followed schematic design in Virtuoso Schematic Composer Tutorial, you will need to name these pins as vdd! and gnd!, so that you have same pin names for power pins in schematic and layout.
- Having different pin names in schematic and layout will cause problems when you run Layout Versus Schematic (LVS) checking tool.
- If you have created simulation view as explained in Simulation with Analog Design Environment Tutorial, naming power pins and nets as vdd! and gnd! helps avoiding recreation of simulation view. Therefore, you can use the same simulation view for both schematic and layout simulation

Layout Extraction

Layout extraction tool is used to generate actual circuit netlist from your layout. The extracted view can be used to run Layout Versus Schematic (LVS) and to run simulation. To run layout extraction tool, go to **Verify -> Extract**. The extractor dialog as shown on the left of Figure 12 should appear. Click on **Set Switches** and then the switches selection window should pop up as shown on the right of Figure 12. Select **Extract_parasitic_caps** and **Keep_labels_in_extractd_view** options as shown in the figure (hold Ctrl key and click on options to select multiple options). Click OK to close set switches dialog and click OK again on Extractor dialog to run the layout extraction.

You should see extraction report on CIW window (if you have large layout, the extraction might take a few minutes before the report shows up) as shown in Figure 13. You should not have any errors/warnings, if there're any, fix it. The most common errors/warnings is having multiple pins on the same nets which means that you might have a shorting net (e.g., vdd is shorted to other nets) or have multiple pins placed on the same net (e.g., having pin named In and INPUT on the same input net).

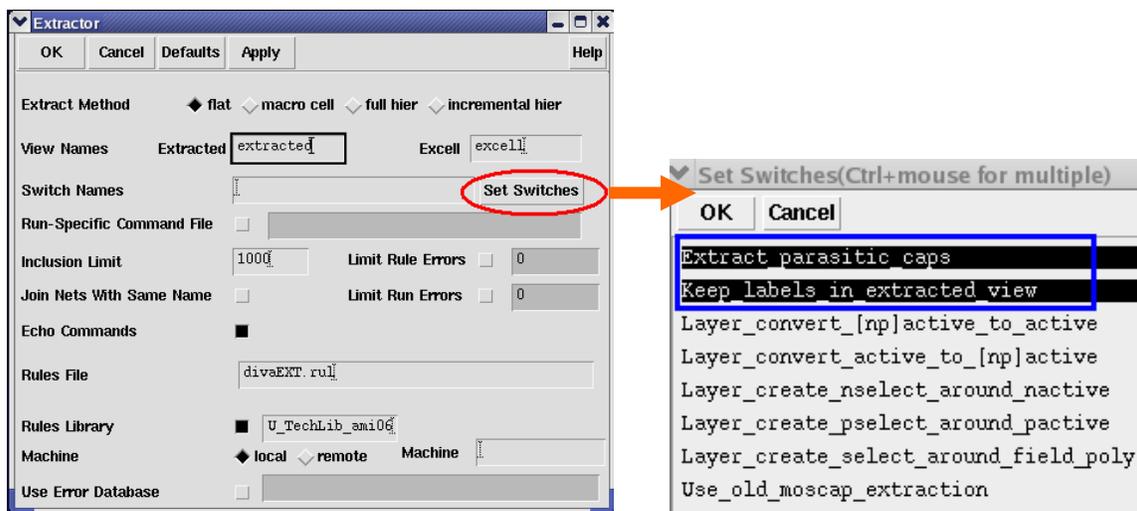


Figure 12: Layout Extractor

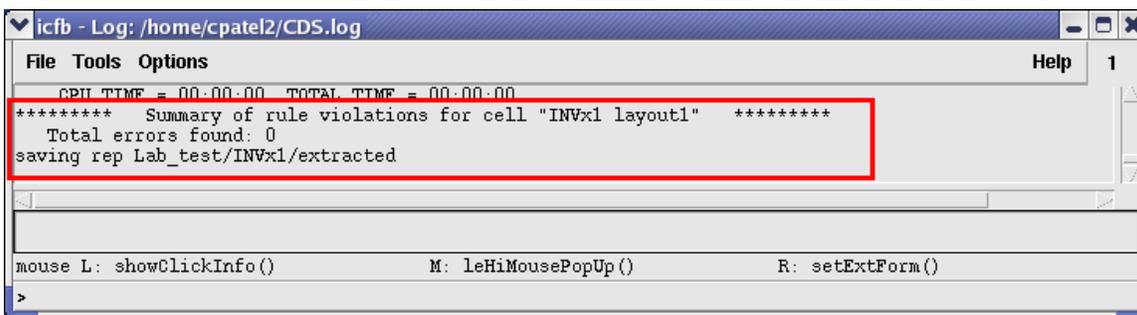


Figure 13: Extraction report on CIW



More info:

- The extraction tool recognizes transistors by looking for the regions where poly layer overlap the active layers. (poly overlap pactive for PMOS and poly overlap nactive for NMOS). The overlap regions also define the size (width and length) of the transistors.
- If the "Extract_parasitic_caps" switch turned on during extraction, the extraction tool calculates the parasitic capacitances on all layers on the layout. For example, extracting wire capacitances on the Metal wires. For some extraction rules of a particular technology, there is an option to also setting the calculation of parasitic resistances.

Multiple Views on Cell Design

If you have followed steps in Virtuoso Schematic Composer Tutorial and this tutorial, you should have multiple views on the inverter cell design. Each of the view represents the same circuit design but at different levels. Figure 14 shows the three (schematic, layout and extracted views) of the four cell views that you should have so far for the inverter design.

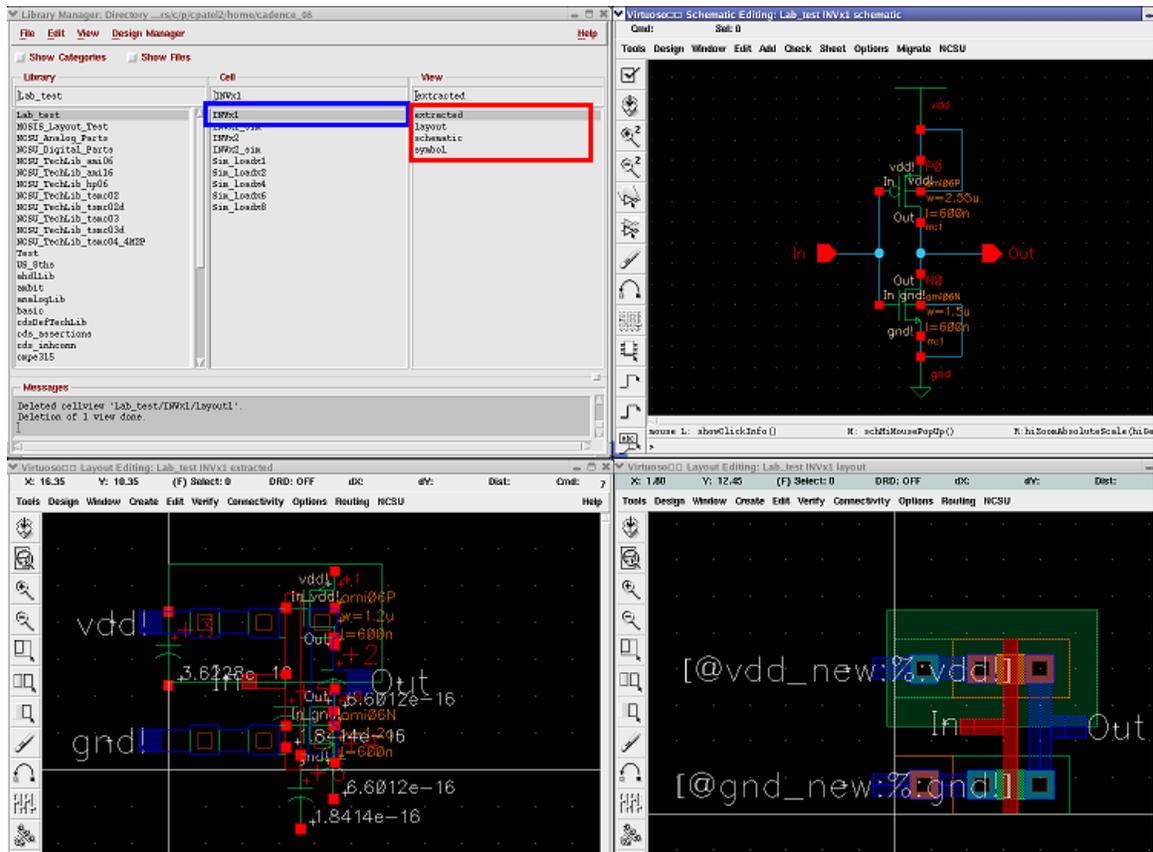


Figure 14: Multiple Cell Views of Inverter

Binding Extracted View to Simulation

In Simulation with Analog Design Environment Tutorial, we have shown you how to create the config view for the simulation using Hierarchy Editor. Now, you can use the same cell view to switch between schematic and extracted layout (and other views that you have e.g., vhdl) in the simulation. The process that used in changing cell view for the simulation is called cell binding.

Start with open the config view by double clicking on the config view from library manager. The open configuration dialog as shown in Figure 15 should appear, select yes to open both config and schematic views. At the Virtuoso Schematic window,

go to **Tools -> Hierarchy-Editor**. The Hierarchy-Editor Menu should then appear on the menu bar as shown in Figure 16.

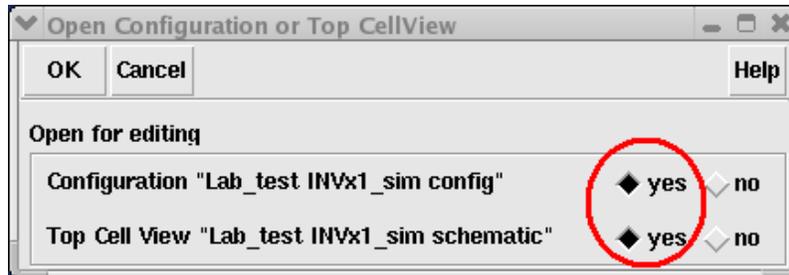


Figure 15: Open Configuration

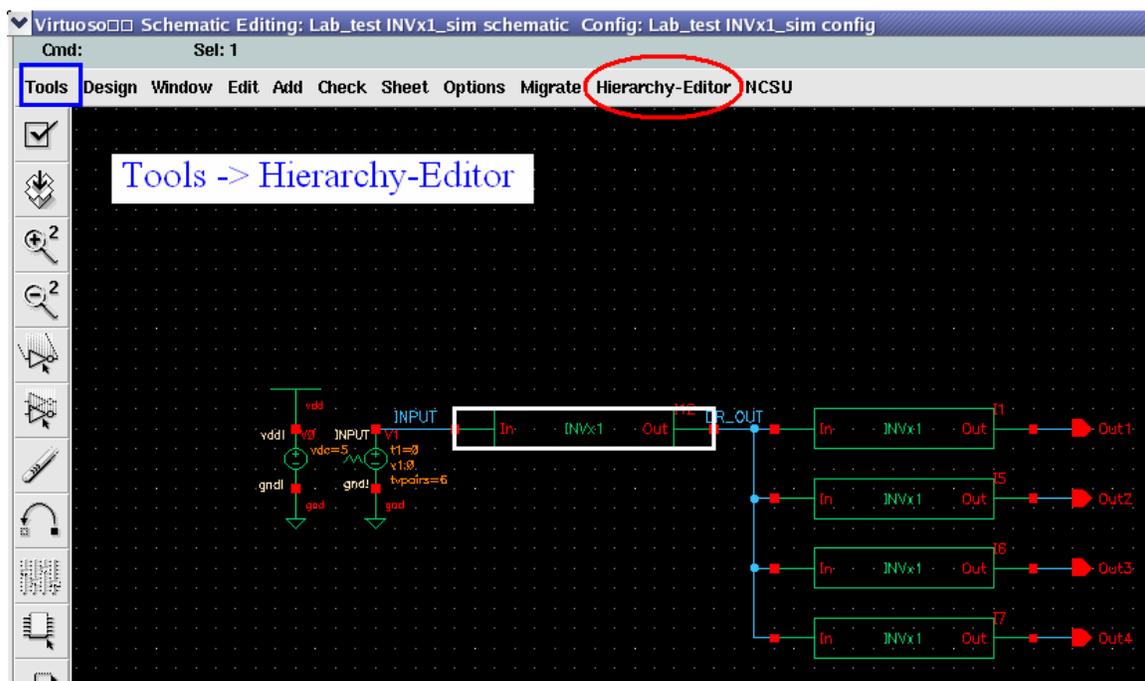


Figure 16: Hierarchy-Editor on Menu Bar

Next, click on the symbol of instance(s) you want to change the binding (you can select multiple instances by holding **Shift** key and click on multiple instances). Go to **Hierarchy-Editor -> Set Instance Binding**, the dialog as shown in Figure 17 should show up. Change **Apply To** from drop-down box to **current instance** or **all selected instances** depending on how you select the instance(s) (if you select only one instance, use current instance. If you select multiple instances, use all selected instances) and then change **View To Use** to **extracted** (or any other views that you want).

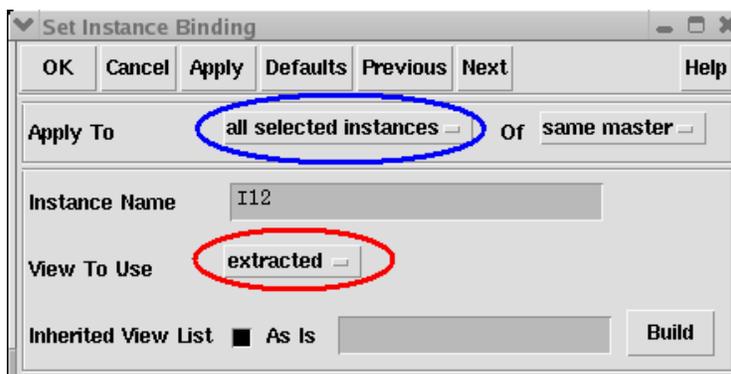


Figure 17: Set Instance Binding

Make cell binding changes to all instances that you want and then go to Configuration window (The configuration window should have shown up at the same time as the schematic view, if not go to **Hierarchy-Editor -> Edit Configuration**). Since you have made changes to cell binding, you should see the  icon on the bottom-right corner of the configuration window and the message says update needed as shown in Figure 18. Click on  icon at the top menu to update all the changes made in schematic to the configuration, then the dialog as shown in Figure 19 should pop up. Check all cell views and then click OK to update.

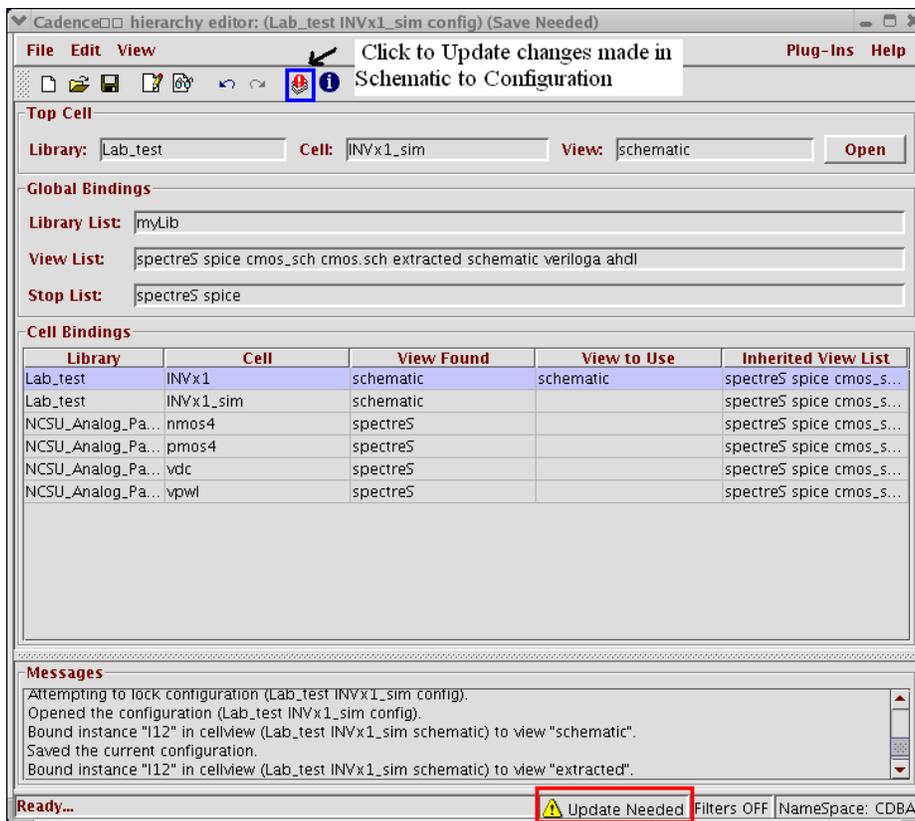


Figure 18: Update Changes to Configuration

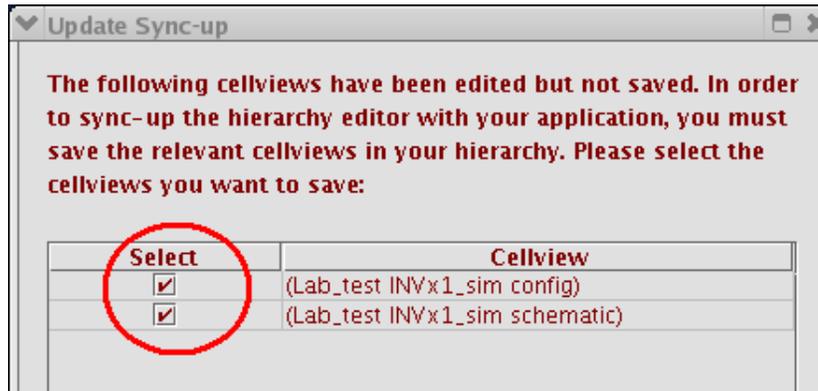


Figure 19: Update Sync-up

Hierarchy Editor is a very powerful tool. You can bind one or more instances of a master cell to the schematic view and select other instances of the same master to extracted view. For example, if you have multiple instances of INVx1, you can bind one group of instances to schematic view of INVx1 and the other group to extracted view. Figure 20 shows the configuration view after update sync-up between schematic of sim viw and configuration has been made. Look carefully at the cell binding table, you could see that in this case some of INVx1 instances were bound to extracted view and others were bound to schematic view. (That is why there are two rows of the same INVx1 cell).

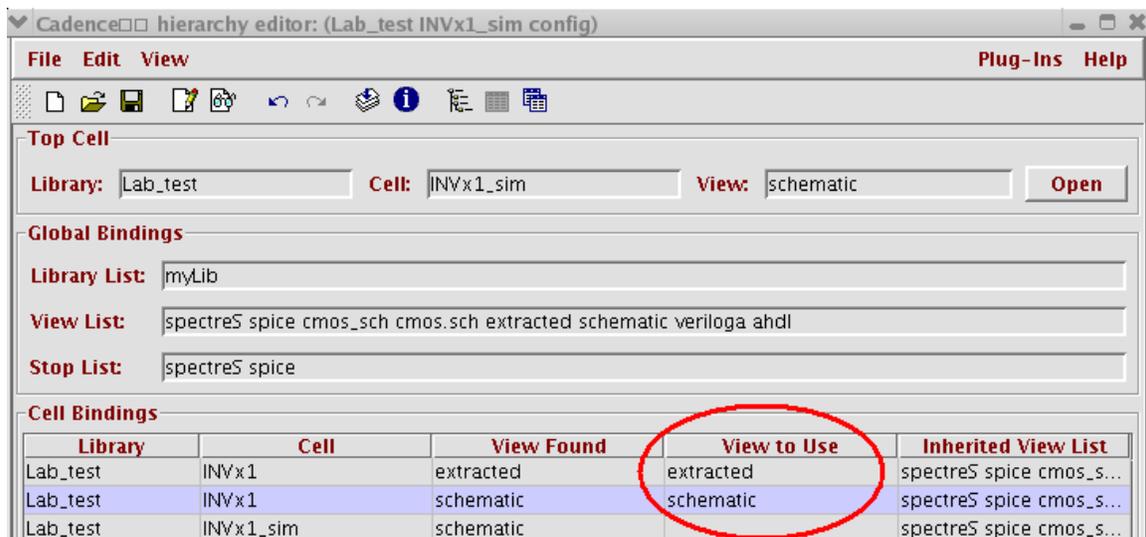


Figure 20: Configuration View after update

If you have many cell bindings in the configuration and want to make sure that you made the correct binding, you can ask Hierarchy Editor to explain the current binding by select the cell you want to check on the Cell Bindings table, click on **i** icon at the menu bar on the top of the window. Figure 21 and 22 show cell binding

explanations of INVx1 extracted and INVx1 schematic respectively. Do not forget to save configuration view once you are done.

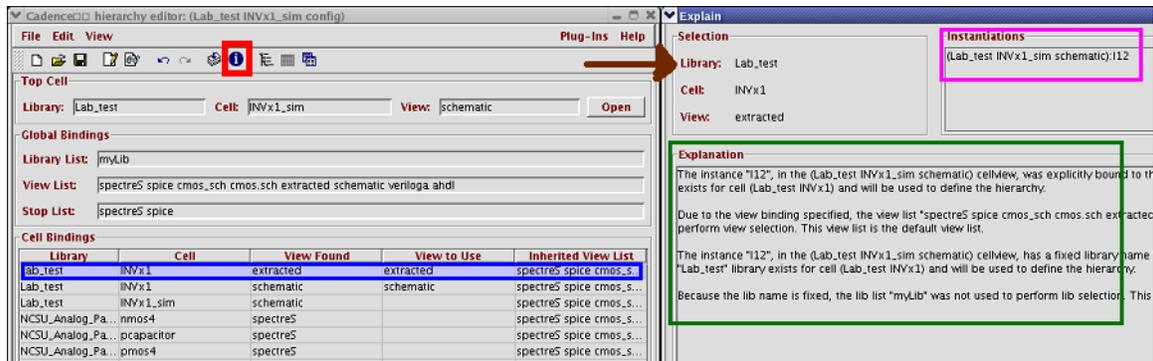


Figure 21: Cell Binding Explain of INVx1 extracted view

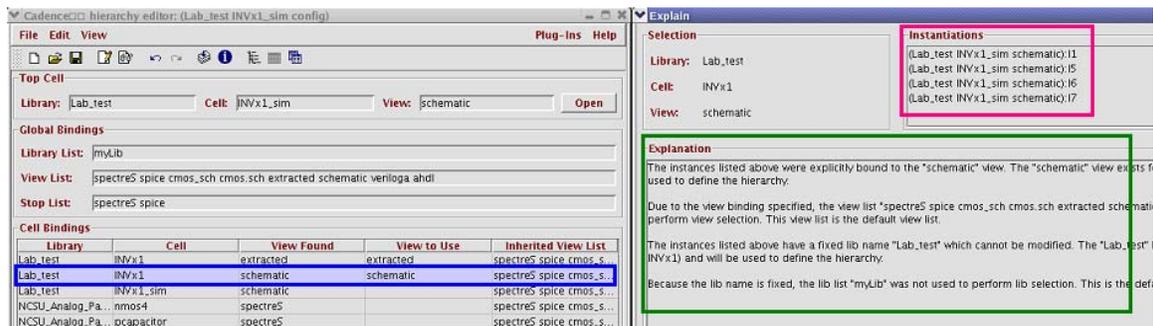


Figure 22: Cell Binding Explain of INVx1 schematic view

Running Simulation on Extracted Layout

The steps involved in simulation of extracted layout is similar to simulation of schematic, the only requirement is that you need to make the cell binding to the extracted view as explain in the previous section. Please refer to Analog Design Environment Tutorial (available on class website) for more information on how to run simulation.

It is a good practice to check the netlist created from simulator in Analog Design Environment to make sure that all the cell bindings you have made are correct before running the simulation. If you use SpectreS simulator, go to **Simulation -> Netlist -> Create Final**. It may take few seconds up to hours for the simulator to generate final netlist depends on the size of the circuit (for the circuits that you design for CMPE315 assignments, they should not take longer than 5 minutes). The final netlist created should show up as shown in Figure 23. In the netlist, there are 5 instances of the inverter, INVx1_g1, as highlighted in blue in the figure. In the sub-circuit section of the netlist (**subckt** is the keyword for sub-circuit definition) as highlighted in red, the view used for binding to this INVx1_g1 is the schematic view as shown on the first line of the highlighted region.

```

/home/cpatel2/cadence/simulation/INVx1_sim/spectreS/config/netlist/spectreFinal
File
* # FILE NAME: /HOME/CPATEL2/CADENCE/SIMULATION/INVX1_SIM/SPECTRES/CONFIG/
* begin cppStatements
#include </home/cpatel2/cadence/simulation/INVx1_sim/spectreS/config/netlist/.cppDe
* end cppStatements

* netlist/INVx1_sim.C.raw
* Netlist output for spectreS.
* Generated on Feb 26 18:19:08 2008

* global net definitions
.GLOBAL vdd!

simulator lang= spectre
simulator lang= spectre
* Include Quantity files

* File name: Lab_test_INVx1_sim_schematic.S.
* Subcircuit for cell: INVx1_sim.
* Generated for: spectreS.
* Generated on Feb 26 18:19:08 2008.

v1 (1 0) vsource type= pwl wave= [ 0.0 0.0 100e-12 0.0 200e-12 5.0 2e-9 5.0
+2.1e-9 0.0 4e-9 0.0 ]
v0 (vdd! 0) vsource type= dc dc=5.0
xi7 (dr_out out4) INVx1_g1
xi6 (dr_out out3) INVx1_g1
xi5 (dr_out out2) INVx1_g1
xi1 (dr_out out1) INVx1_g1
xi12 (1 dr_out) INVx1_g1

simulator lang= spice

simulator lang= spectre
simulator lang= spice

* File name: Lab_test_INVx1_schematic.S.
* Subcircuit for cell: INVx1.
* Generated for: spectreS.
* Generated on Feb 26 18:19:08 2008.

simulator lang= spectre
* terminal mapping: In = in
*                      Out = out
subckt INVx1_g1 in out
mp0 (out in vdd! vdd!) ami06P region= sat w=2.55e-6 l=600e-9 as=3.825e-12
+ad=3.825e-12 ps=8.1e-6 pd=8.1e-6 m=1
mn0 (out in 0 0) ami06N region= sat w=1.5e-6 l=600e-9 as=2.25e-12
+ad=2.25e-12 ps=6e-6 pd=6e-6 m=1
simulator lang= spice

```

Figure 23: Netlist of config view using INVx1 schematic

Figure 24 shows the netlist resulted from using multiple views (schematic and extracted) for cell binding. In the figure, there are 4 instances, xi7, xi6, xi5 and xi1, that were bound to schematic view and there is one instance, xi12, that was bound to extracted view.

```

* Generated on Feb 26 18:25:50 2008.

v1 (1 0) vsource type= pul wave= [ 0.0 0.0 100e-12 0.0 200e-12 5.0 2e-9 5.0
+2.1e-9 0.0 4e-9 0.0 ]
v0 (vdd! 0) vsource type= dc dc=5.0
xi7 (dr_out out4) INVx1_g1
xi6 (dr_out out3) INVx1_g1
xi5 (dr_out out2) INVx1_g1
xi1 (dr_out out1) INVx1_g1
xi12 (1 dr_out 0 vdd!) INVx1_g2

simulator lang= spice

simulator lang= spectre
simulator lang= spice

* File name: Lab_test_INVx1_extracted.S.
* Subcircuit for cell: INVx1.
* Generated for: spectreS.
* Generated on Feb 26 18:25:50 2008.

simulator lang= spectre
* terminal mapping: In = in
*                      Out = out
*                      gnd! = 1
*                      vdd! = 2
subckt INVx1_g2 in out 1 2
m3 (out in 2 2) ami06P region= sat w=1.20000004244503e-6
+l=600.000021222513e-9 as=1.80000003617564e-12 ad=1.80000003617564e-12
+ps=4.20000014855759e-6 pd=4.20000014855759e-6 n=1
m4 (out in 1 1) ami06N region= sat w=1.20000004244503e-6
+l=600.000021222513e-9 as=1.80000003617564e-12 ad=1.80000003617564e-12
+ps=4.20000014855759e-6 pd=4.20000014855759e-6 n=1
c5 (in 1) capacitor c=184.14e-18 n=1.0
c6 (in 2) capacitor c=184.14e-18 n=1.0
c7 (out 1) capacitor c=660.12e-18 n=1.0
c8 (2 1) capacitor c=362.28e-18 n=1.0
c9 (2 out) capacitor c=660.12e-18 n=1.0
simulator lang= spice

simulator lang= spectre
* End of subcircuit definition.
ends INVx1_g2
simulator lang= spice

* File name: Lab_test_INVx1_schematic.S.
* Subcircuit for cell: INVx1.
* Generated for: spectreS.
* Generated on Feb 26 18:25:50 2008.

simulator lang= spectre
* terminal mapping: In = in
*                      Out = out
subckt INVx1_g1 in out
mp0 (out in vdd! vdd!) ami06P region= sat w=2.55e-6 l=600e-9 as=3.825e-12
+ad=3.825e-12 ps=8.1e-6 pd=8.1e-6 n=1
mn0 (out in 0 0) ami06N region= sat w=1.5e-6 l=600e-9 as=2.25e-12
+ad=2.25e-12 ps=6e-6 pd=6e-6 n=1
simulator lang= spice

```

xi7, xi6, xi5 and xi1 are instances of INVx1_g1

xi12 is an instance of INVx1_g2

INVx1_g2 is created from extracted view

INVx1_g1 is created from schematic view

Figure 24: Netlist of config using both extracted and schematic views