

Cadence Tutorial for EE5323 VLSI Design I

Setting up your Account

1. Setting up the Process Design Kit (PDK):

1.1 Copy the file NCSU-FreePDK45-1.3.tar from /home/class/ee5323ta to your home directory by using the command from your home directory:

```
cp /home/class/ee5323ta/NCSU-FreePDK45-1.3.tar .
```

1.2 Extract the archive using the command

```
tar -xvf NCSU-FreePDK45-1.3.tar
```

1.3 You should now be able to see a directory called FreePDK45 in your home directory. This directory contains an open-source, Open-Acess based PDK for the 45nm technology node and the predictive technology model which you will be using throughout this course.

1.4 Go to ~/FreePDK45/ncsu_basekit/cdssetup

```
cd ~/FreePDK45/ncsu_basekit/cdssetup
```

Here, you will have to modify 2 files - cds.lib and setup.csh using a text editor (e.g. vi or nano):

1.5 Add the following line at the end of your cds.lib

```
DEFINE freepdk_cells $PDK_DIR/osu_soc/lib/freepdk45_cells
```

1.6 Modify setup.csh as follows:

1.6.1 Comment out the line which defines the environment variable PDK_DIR. Lines can be commented out by adding a # at the beginning.

1.6.2 Set the PDK_DIR variable to the root directory of the FreePDK distribution

```
setenv PDK_DIR ~/FreePDK45
```

1.6.3 Comment out the line which defines the environment variable CDSHOME

1.6.4 Set the CDSHOME variable to the root directory the Cadence installation

```
setenv CDSHOME /home/vlsilab/cadence/ic615
```

1.6.5 Save and exit the file. Your process design kit is setup and ready to be used now.

1.7 Create your temporary Cadence work directory

```
mkdir ~/cds_ncsu45
```

```
cd ~/cds_ncsu45
```

Consider running Cadence from this directory to avoid cluttering up your workspace

1.8 Copy setup.csh (the file you modified in step 1.6) into this directory

```
cp ~/FreePDK45/ncsu_basekit/cdssetup/setup.csh .
```

Source this script using the following command

```
source setup.csh
```

This script copies the files needed by Cadence and initializes the environment.

1.9 Invoke Cadence by typing “virtuoso &” or “icfb &”. This should bring up the command interface window (CIW) and library manager. Check that the version should be 6.1.5 at the top of the CIW. You are now ready to design circuits in Cadence.

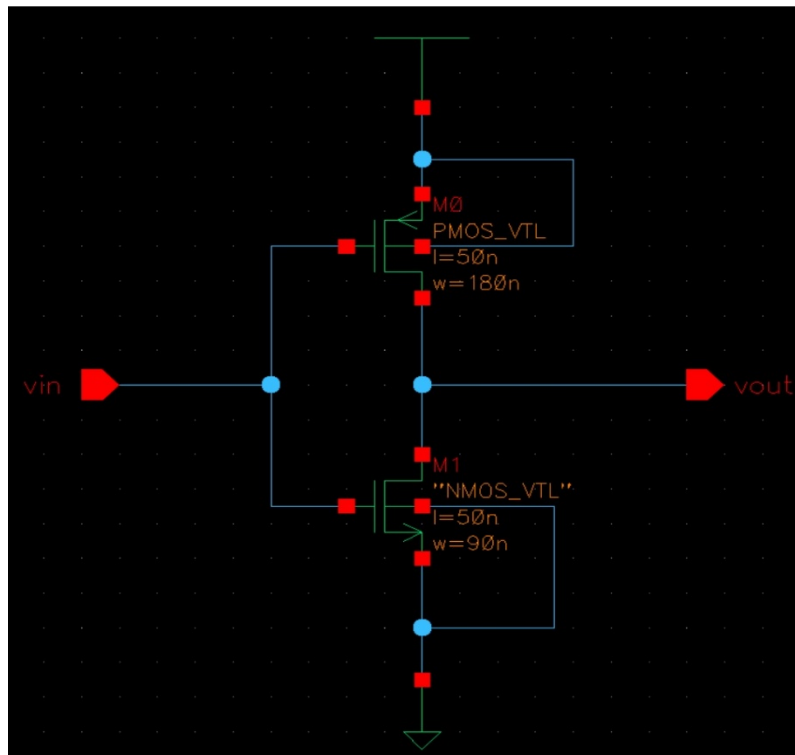
1. Example: Design and Simulation of an Inverter

This example will help you familiarize yourself with Cadence. This will show the most important commands and steps used when working with schematics in Cadence. As an example, you will design a simple inverter and simulate the delay of it. Before starting with the design example, there are a couple things worth mentioning:

- Most of the commands in Cadence can be accessed in multiple ways: pull-down menus, shortcut keys, buttons in toolbars, etc. In the described example, all the commands are referenced by their position in the pull-down menus. The shortcut keys can be found from the pull-down menus as well. It is highly recommended to use shortcut keys to increase your efficiency with Virtuoso. Below lists some most frequently used shortcut keys:
- q - Edit property of object
- i - Create instance
- w - Add wires

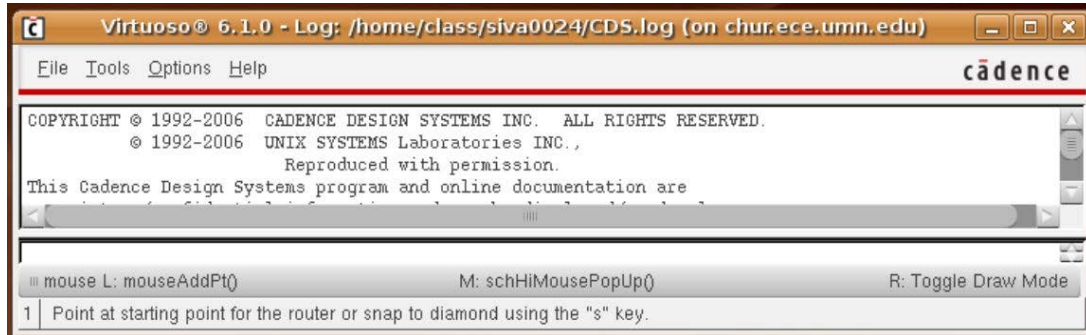
- m - Move
- c - Copy
- s - Stretch
- r - Rotate
- k - Create a ruler
- z - Zoom in
- Z - Zoom out
- u - Undo
- X - Descend edit
- x - Descend read
- b - Return
- e - Display
- M - Merge multiple shapes into a single piece
- Ctrl+F - Hierarchical layout view (hide details of sub-instances)
- Shift+F - Descended layout view (show details of sub-instances)
- The most frequently used key in Cadence is ESC. It is used to cancel on-going commands.

The following picture shows the schematic of an inverter, which is ready for netlist extraction. The following section explains how to draw it in Cadence.



1.1 Create a library for your new design

Cadence Virtuoso is shown in the following picture



From Virtuoso,

File->New->Library

Type a new name(I used "sample" and this is what I will refer to from here on).

Select the "Attach to an existing technology library" option, click OK.

Then in the popup window, select **NCSU_TechLib_FreePDK45** and click OK

In Virtuoso command window, you will see the following messages (among other warnings) if everything is OK

```
Created library "sample" as
"/home/class/your_user_name/cds_freepdk/sample"
```

```
Design library 'sample' successfully attached to technology library
'NCSU_TechLib_FreePDK45'
```

1.2 Create a new cell

From the Virtuoso command window,

File->New->Cellview

In the popup window, select "sample" (or the name of the library you created in step 1.1).

Cell name "inverter". view name "schematic" and open with "Schematics L". Click Ok

Click "Always" if "Upgrade License" warning message shows up. Virtuoso schematic editor opens up.

Remember that anytime you get an Upgrade license warning, select the "Always/Yes" option every time. Also a good way to browse through the library and cells is through the Library Manager (At CIW, Tools -> Library Manager)

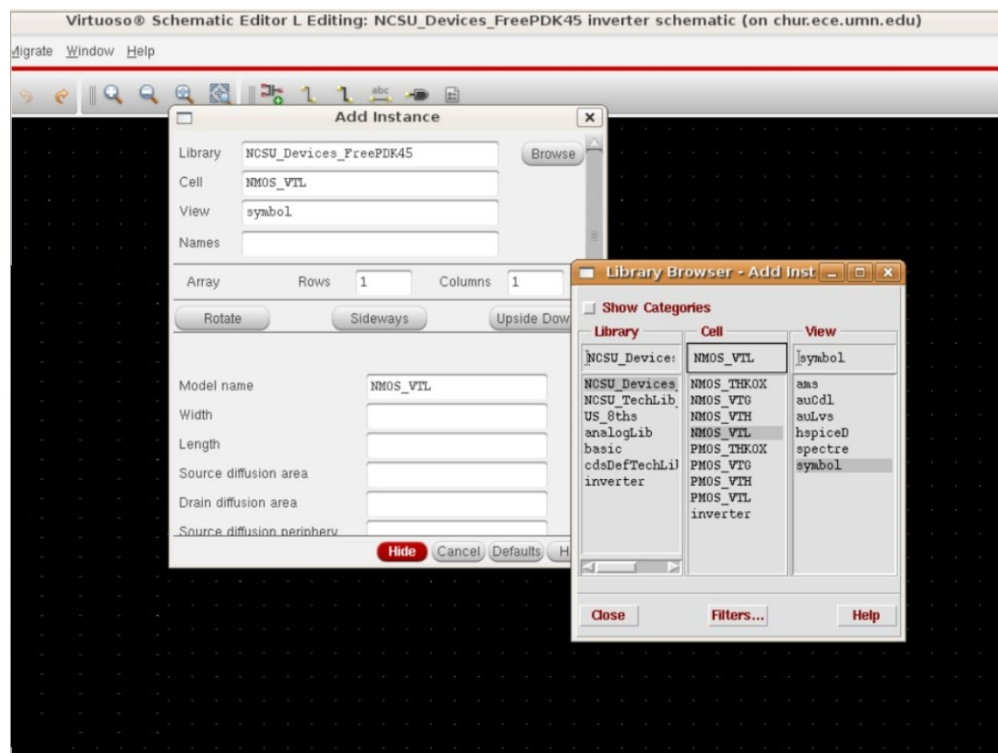
1.3 Design your Circuit

1.3.1 Place Components

For this inverter, you will need an NMOS transistor, a PMOS transistor, power and ground nodes.

From Schematic window,

Create->Instance (or press 'i') to open Add Instance window. Click on "Browse". Now the Library Browser window open. These windows are shown in the following picture



Make sure the Library in the library browser is set to **NCSU_Devices_FreePDK45**. Use the library browser window and click on NMOS_VTL, then select the symbol view. You can now enter the width and length of the transistor in the *Add Instance* window. Use the same procedure for PMOS transistors. Also, from the library **Basic**, you can get the symbols for VDD and GND (they define the net names for the power and ground nodes). If you make any mistake, you can always use

Edit->Delete or

Edit->Rotate or

*Edit->Move or
Edit->Stretch*

To change the properties of some of the components:

Edit->Properties->Objects

Select the NMOS_VTL transistor and make sure the width and length are set to 90.0n M and 50.0n M respectively. If not, you can change the properties of the transistor. Similarly, set the width and length of the PMOS_VTL transistor to 180.0n M and 50.0n M.

1.3.2 Connect Components

Connect the components as shown in the schematic above by using
Create->Wire (Press 'w')

1.4 Add Pins

By adding pins, you can identify the I/O ports of the schematic. At a later stage, you can also use pins as connection points for hierarchical designs. To learn more about this, see the section about [creating symbols](#).

Create->Pin (Press 'p')

Type the pin name, such as VIN, select the direction as "input", and place it in the schematic. Do the same for VOUT, selecting the direction as "output".

1.5 Generate Netlist

In the schematic window:

Design->Check and Save

There should be no errors.

Launch->ADE L

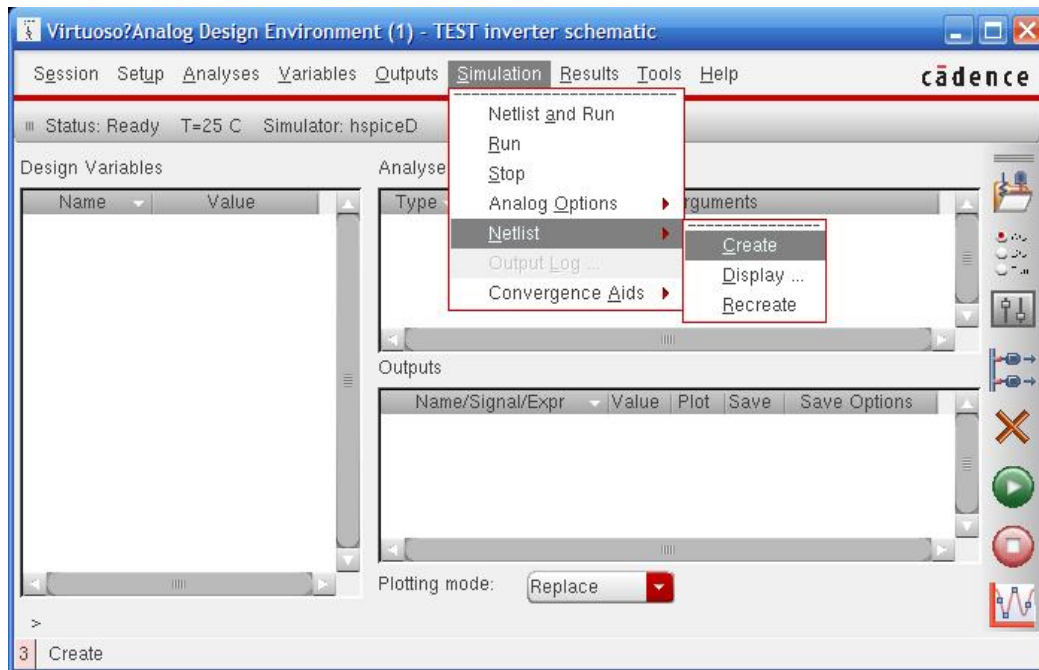
Another license prompt may appear, and click "Always"

The window of Cadence Analog Design Environment will show up.

Setup->Simulator/Directory Host

Set the simulator to hspiceD

Simulation->Netlist->Create



A new window will pop up showing the generated HSPICE netlist. You may save this file by clicking the menu bar:

File->Save As

Specify the full path name and file name in the *Save As* window. For example, if we want save the file into a folder named "simulation" under folder "~/cds_ncsu", we should type in "~/cds_ncsu/simulation/inverter.sp". And the file name is "inverter.sp". If you do not provide a path before file name, the file will be saved under the folder where you start your Cadence.

Alternatively, you can select the text, right click and select "Copy" (or press Ctrl+Insert) to paste in another text file and save it as "inverter.sp" to your desired location.

```

** Generated for: hspiceD
** Generated on: Sep 17 17:38:06 2008
** Design library name: inverter
** Design cell name: inverter
** Design view name: schematic
.GLOBAL vdd!

.TEMP 25
.OPTION
+   ARTIST=2
+   INGOLD=2
+   MEASOUT=1
+   PARHIER=LOCAL
+   PSF=2

** Library name: inverter
** Cell name: inverter
** View name: schematic
m0 vout vin vdd! vdd! PMOS_VTL L=50e-9 W=180e-9
m1 vout vin 0 0 NMOS_VTL L=50e-9 W=90e-9
.END

```

Now we have the netlist ready for editing and simulating. Open the netlist in an editor and comment out the lines that are highlighted in the above picture. You can comment lines in a SPICE netlist by using *. Save the netlist and exit the editor.

Create another SPICE file named "runinv.sp" as below for simulation. It should contain the part of the SPICE file inverter.sp that you commented out in the previous step. You also need to include the models for the transistors and the generated netlist file "inverter.sp".

Finally you provide the values for the input and VDD.

```

**Test inverter

.TEMP 110
**Use high temperature to simulate worst case delay and leakage power

.OPTION

+   ARTIST=2

+   INGOLD=2

+   MEASOUT=1

+   PARHIER=LOCAL

+   PSF=2

```



```

+                POST

.inc
'/home/class/your_user_name/FreePDK45/ncsu_basekit/models/hspice/tran_models/
models_nom/NMOS_VTL.inc'

.inc
'/home/class/your_user_name/FreePDK45/ncsu_basekit/models/hspice/tran_models/
models_nom/PMOS_VTL.inc'

.include inverter.sp

v1 vdd! 0 1.1v

v2 vin 0 pwl 0ns 0 1ns 0 1.02ns 1.1V 2ns 1.1V 2.02ns 0 2.5ns 0

.op

.tran 0.1p 3ns

.end

```

Save the file “runinv.sp” and Exit the editor. The reason for using a separate file for simulation is because later on your circuit netlist may change, but the simulation file can be kept the same. Note that you should always leave the first line of your .sp file for comments since HSPICE automatically ignores it.

Note: Make sure your file contains the line "+ POST" as the generated netlist does include it. If you miss it, probably you will not be able to open your .tr file.

1.6 HSPICE simulation

The "runinv.sp" file includes the circuit netlist, technology models, supply voltages, input signal timing and simulation time. The setup for simulation is now complete. At the Linux command type in

```
hspice runinv.sp
```

If you have followed all the steps correctly, you will see the following message:

```
Using: /usr/bin/time -p /home/vlsilab/synopsys/hspice/linux/hspice runinv.sp
***** HSPICE -- C-2009.09-SP1 32-BIT (Nov 23 2009) linux *****
```

```
Copyright (C) 2009 Synopsys, Inc. All Rights Reserved.
```

```
...
```

```

...

...

***** job concluded

1***** HSPICE -- C-2009.09-SP1 32-BIT (Nov 23 2009) linux *****

*****
**test inverter

** test inverter

...

...

...

lic: Release hspice token(s)

```

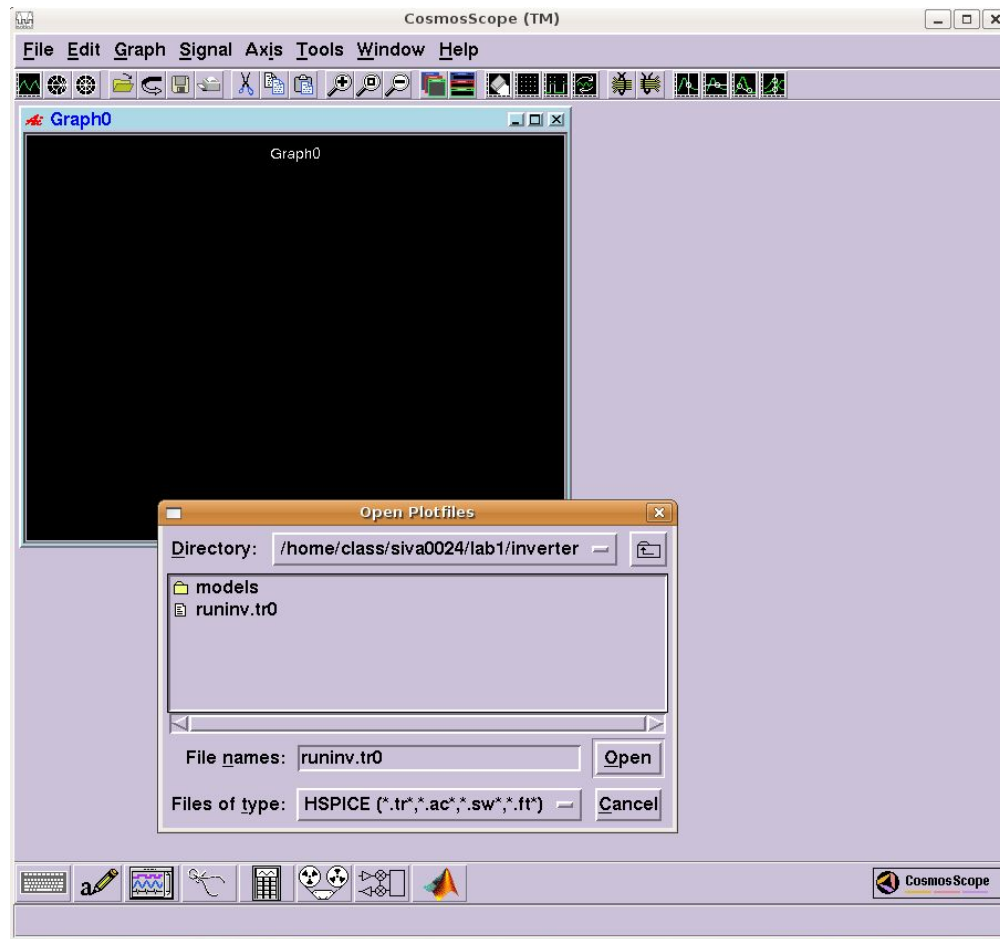
The important message is the “job concluded”. If the job is “aborted”, you will have to debug the errors in your netlist. For detailed HSPICE command and syntax, the readers are referred to the HSPICE manual on the class webpage.

1.7 Use Cosmoscope to view results

To view results, we have to start scope. For this, at the linux prompt type in *scope* Synopsys' CosmosScope opens. From the menu bar,

File->Open->Plotfiles

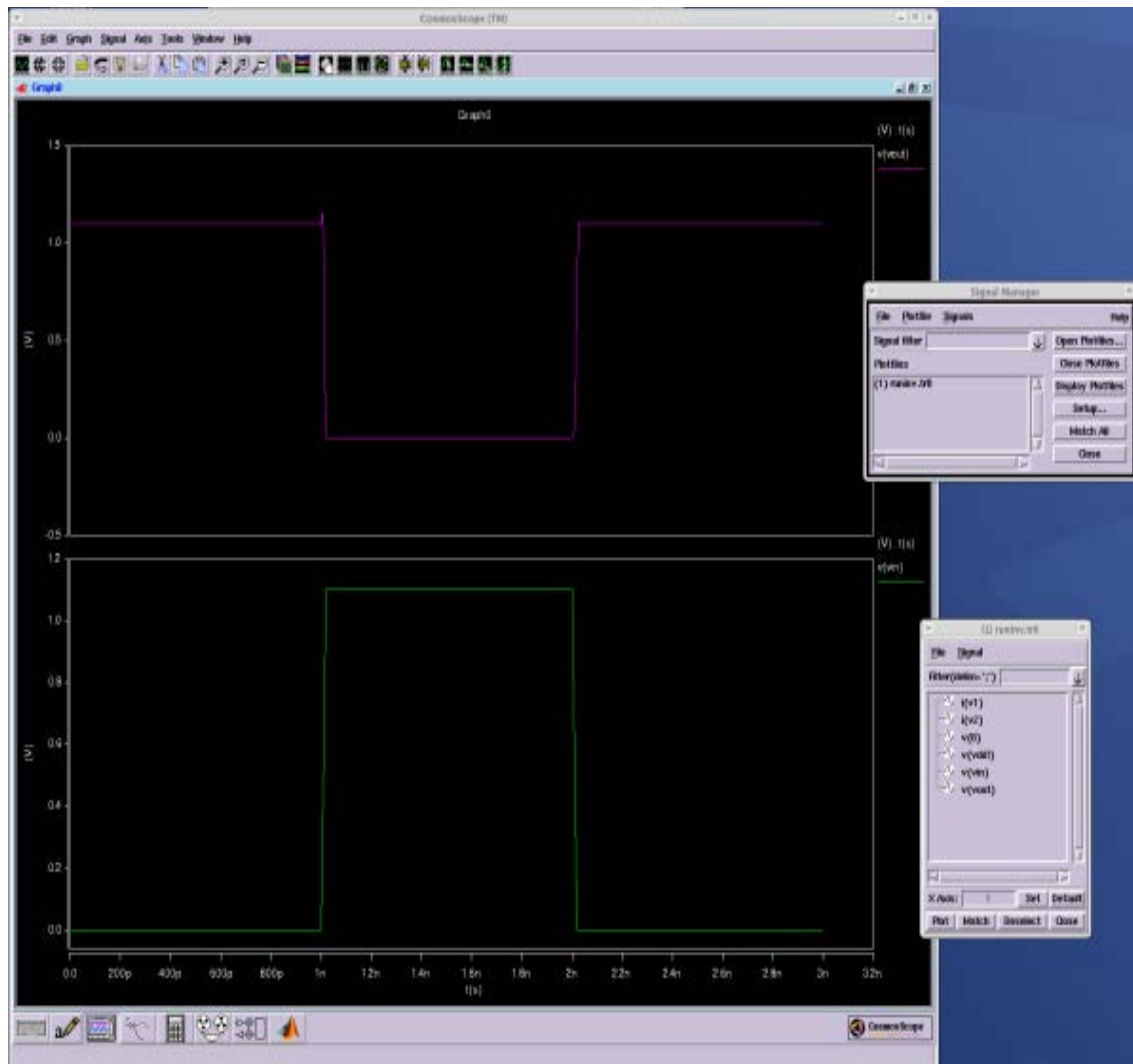
In the Opened window, select "HSPICE (*.tr*,*.ac*,*.sw*,*.ft*)" for “Files of type” and then select the "runinv.tr0" file. Clicking Open brings up the results browser window.



In the results browser window, double click "v(vin)" and "v(vout)" to plot the waveforms of the signals.

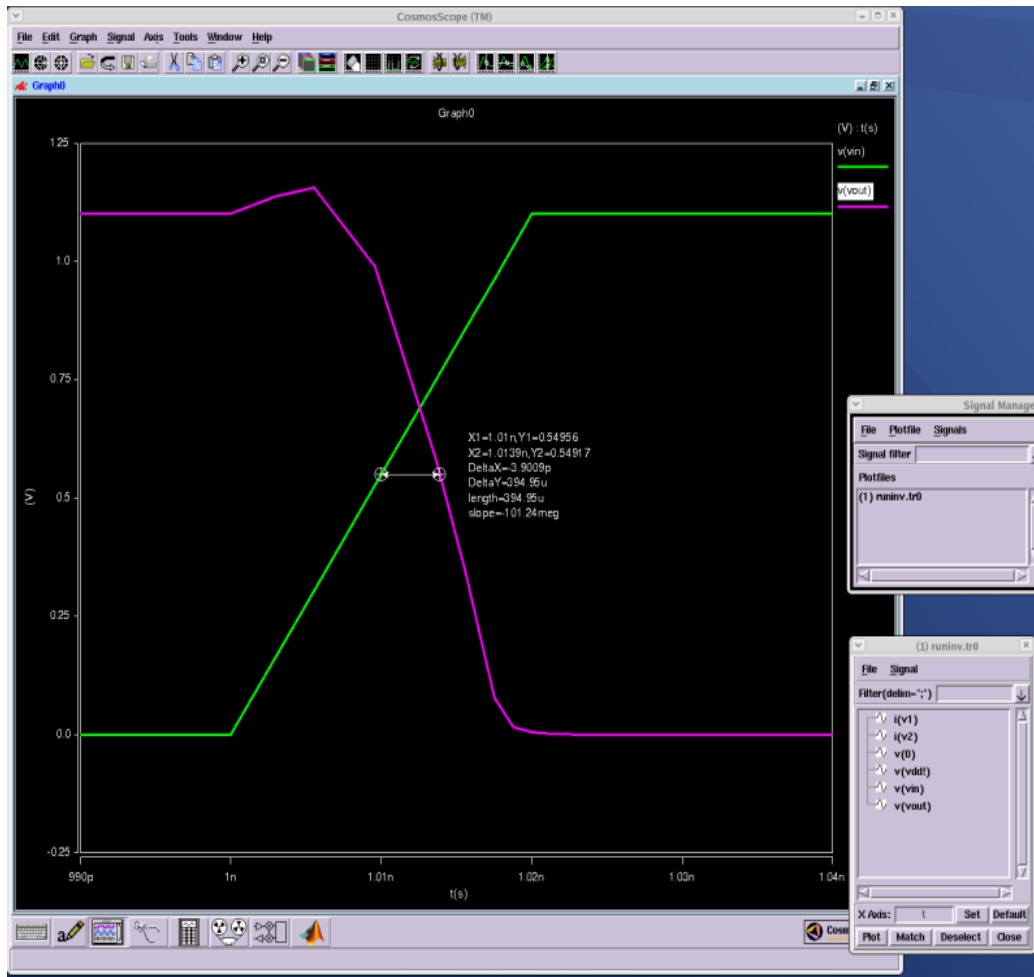
You can change the font of the text, as well as the line width of the waveforms. Right click on the waveform name and options on line width, style, color will appear. To present waveforms directly from scope, you should increase the font size as well as line width.

You can also save the data to a text file. Click on the target waveform name on the right, then go to File-> save waveform to save it in a text format. You can copy and paste these data in excel or read it in Matlab to get a better plot figure.



You can measure the propagation delay by

- 1) Drag the v(vout) signal into the same window as v(vin) signal;
- 2) Select both "vin" and "vout" signals using control key;
- 3) Click the second but last button on the tool bar;
- 4) Drag the start and end circles to the location you want to measure;



You can also use HSPICE command to measure any desired parameters, e.g., clock period, propagation delay, etc. Here gives an example to measure the propagation delay:

.MEASURE TRAN th1 TRIG V(vin) VAL=0.55 TD=10n RISE=5 TARG V(vout) VAL=0.55 TD=10n FALL=5

Here, "TRAN" indicates we want to measure from transient simulations, "th1" is the variable name which will store the measured result, "TRIG" indicates the triggering event, which is defined as "V(vin) VAL=0.55 TD=10n RISE=5", meaning the 5th rising edge (RISE=5) of vin (V(vin)) after 10ns (TD=10n) when it reaches 0.55V (VAL=0.55). Similarly, "TARG V(vout) VAL=0.55 TD=10n FALL=5" represents the 5th falling edge of vout after 10ns when it reaches 0.55V. The delay between these two events will be saved in the variable "th1".

If .measure runs correctly, it will generate a separate file containing the measured results. For example, if you are using runinv.sp, the generated file will be runinv.mt0.

For more details of ".measure", please refer to the HSPICE manual.

1.8 Working with Symbols

If you want to use your design in other schematics, you need to create a symbol for it. This is equivalent to the use of sub-circuits in HSPICE. Using hierarchy in your project makes it easier to organize.

1.8.1 Create a new symbol

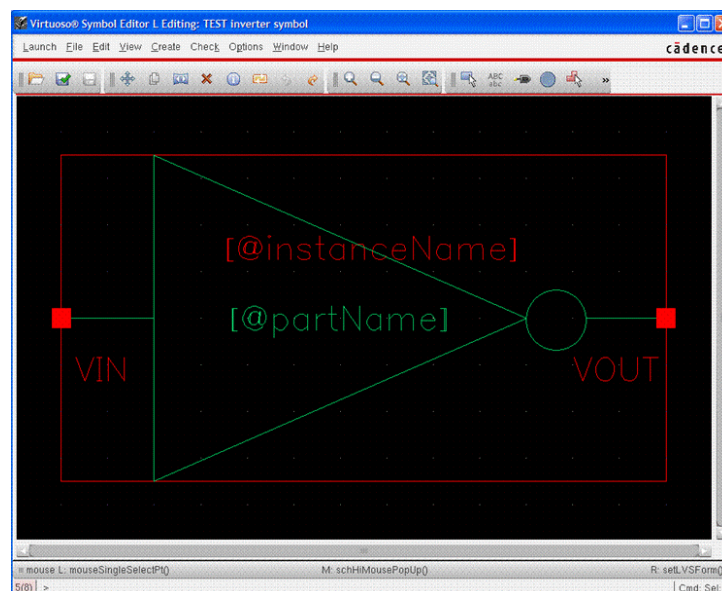
Save the schematic before you create its symbol:

File->Check and Save

Create->Cellview->From Cellview, click OK.

A new popup window titled *Cellview From Cellview* opens up. Make sure that the *Library name* and *Cell Name* fields refer to the names of your cell and the new library you created earlier in the tutorial. *From View Name* should be *schematic* and *To View Name* should be *symbol*. *Tool /Data Type* should be set to *schematicSymbol*. Click on OK.

A *Symbol Generation Options* window opens up. Click on OK. A new window will open with the symbol view. By default, the symbol shape is a rectangle, but you can change it. Since this design is an inverter, we will draw a triangle and put a small circle at the output. To do this, you will want to delete the green rectangle, draw the new shape, and move the terminals to new positions. Use *Create->Shape* to draw a triangle and place a circle. There are several shapes available: line, rectangle, circle, etc. You will also need to change the Selection box (the red rectangle), which defines the limits of the symbol. This can be done by stretching the Selection box. Figure below shows an example of the inverter symbol:



Don't forget to check and save.

1.8.2 Use the symbol in other schematics

Create a new schematic, using the instructions described in [Create a new cell](#). Give a name such as test_inverter. You place this symbol in the new schematic in the same way that you placed any other components, with:

Create->Instance

This time change the Library to the library you created in this tutorial and click on inverter. Your symbol should be here. Now, you can define power supplies in the new schematic. If you place new VDD and GND components, they will be implicitly connected to the correspondent VDD and GND components that are inside the inverter.

To move in the hierarchy, select the inverter, and then:

Edit->Hierarchy->Descend Edit

You can choose the schematic or the symbol for editing.

To return to the previous schematic, use:

Edit->Hierarchy->Return

2. Layout of the Inverter

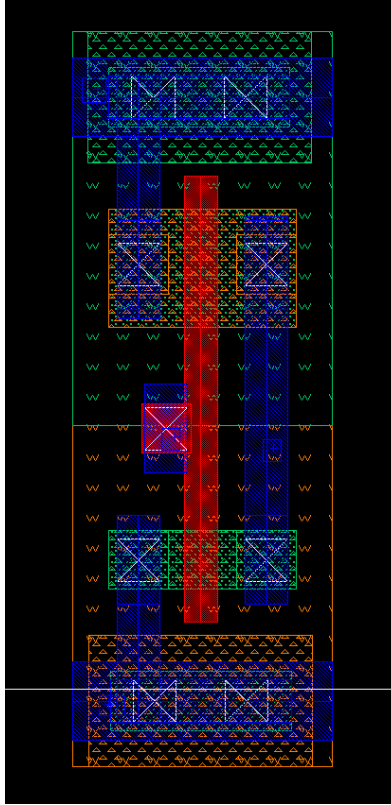
This example will help you create a layout of the inverter that you created in the first example. There are many considerations to take into account when deciding how to do a layout. This is NOT an example on layout techniques but more of a generalized example to help you get familiar with Virtuoso and laying out some basic components.

There are many layout examples for typical logic gates in the library "freepdk_cell". You can also take them as references while drawing your own layout.

Here are some tips for drawing the layout:

- 1) Use ctrl+f or shift+f to hide or show the details of sub-instances. This is very useful when you are doing a top level design.
- 2) Sometimes the edit commands, e.g., move, stretch, etc, do not respond. You can try press ctrl+d first, which will cancel your previous command, and then do the move, stretch or whatever you like.
- 3) You can merge different pieces of the same layer into a single piece by pressing shift+m after selecting all the pieces. This can make your design better organized.

The following picture shows the layout of the inverter. The rest of the section explains how to make each of the separate components in Virtuoso.



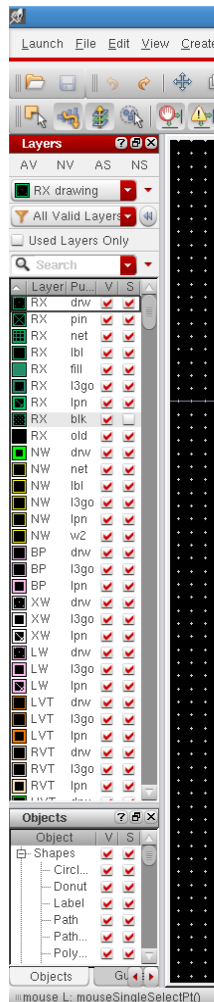
2.1 Create layout view of the inverter

Good layout examples can be found in the “freepdk45_cells” standard cell library. We encourage you to use them in your later designs to save layout effort.

In the Virtuoso command window, Click on *File->New->Cell View*. In that select the library you created in the schematic design entry example. In the *Type* field, select "layout". The *Open with* application should be automatically set to "Layout L". Click on Ok. Click "Always/Yes if there is any license upgrade message". A Layout editing window with sub-windows e.g. “Layers” and “Objects” will open up. You are now ready to layout the inverter you designed earlier.

2.2 Layout Components for your circuit

You will create circuit components by painting shapes on the layout editor. You will use the Layers sub-window to select appropriate layers. The LSW is shown in the following picture.



You can draw a shape by first selecting the layer from the Layers sub-window, then *Create->shape->....* You will be drawing rectangles for the most part. Note that this is similar to how you created symbols in [Section 1.8](#).

2.2.1 Laying out an NMOS transistor

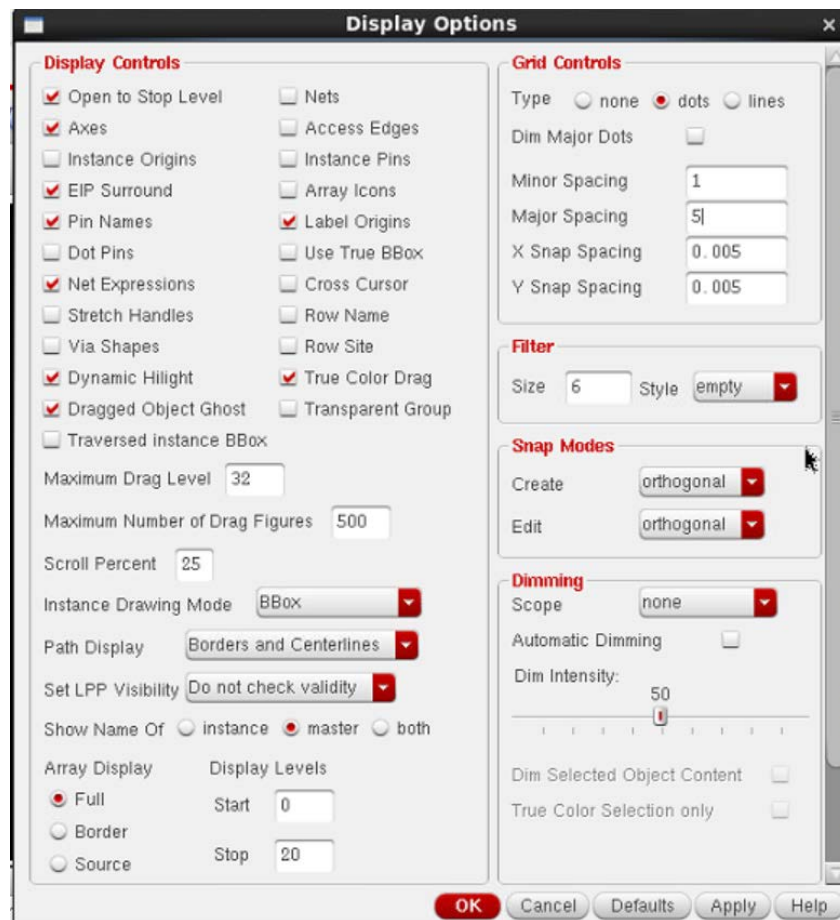
In this section, you will learn how to layout an NMOS transistor in this process. Before going into detail about the layout, it is recommended that open up the design rules of the process in another browser window. You can get the design rules of the FreePDK45 process from [here](#). You will need to refer to the design rules later to fix errors in your layout.

Display and Editor Settings:

Each time you start working, check the display and editor settings. You just need to configure it once per cadence run.

From Options -> Display (Or just press “e”), you can find “Display Options”. You can check or uncheck pin name box on the left depending on your preference.

On the right of the box, X and Y axis spacing are specified. Make sure all the files of the same project shares the same X Snap and Y Snap spacing to avoid grid errors. The minimum X snap and Y snap spacing is 5nm, which is 0.005 micro-meter.



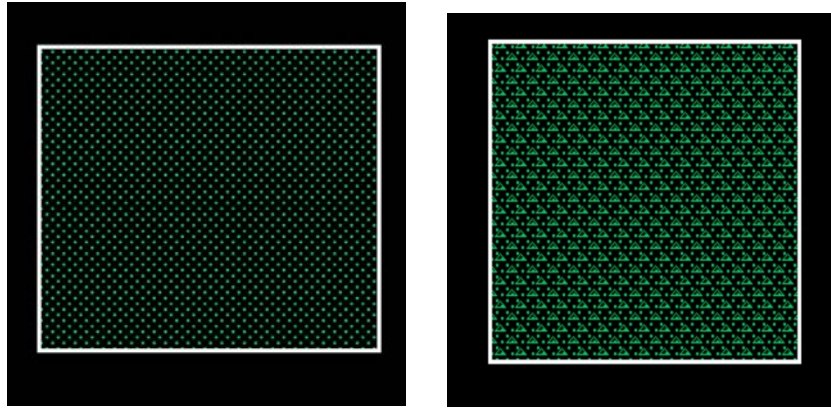
An NMOS transistor uses the following layers: **pwell, active, nimplant, poly, metal1 & contact..** You can start designing the transistor by selecting "active" on the LSW and drawing a rectangle on the layout editor. Note also the letters "drw", "net", and "pin" next to each entry in the LSW. These are the purposes of a shape. The purpose is used to indicate special functionality of a shape. We will be using "drw" for now.

At this point, do not worry about drawing the rectangle to exact dimensions and just draw a rectangle of any arbitrary dimension. You can zoom in and out of the editor by using the zoom Rbuttons located on the top menubar of the editor. You can also draw a box around the area you want to zoom in by holding on to your right mouse click button. When you release the button, the area you selected will be zoomed in.

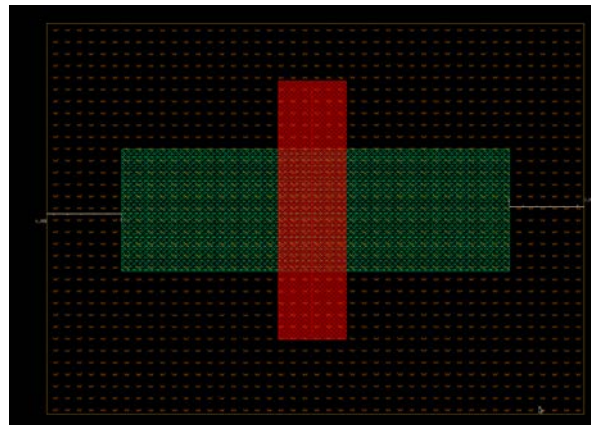
Once you draw the rectangle, you can select it and press "q" or *Edit->Basic->Properties* to change the dimensions of the rectangle. You can also adjust the dimension by pressing "s" or *Edit->Stretch*

You can move objects around by pressing "m" or *Edit->move*. Readers are strongly encouraged to get familiar with other keyboard shortcuts as this will reduce design time later on. To measure the distance between two points, you can press "k" or *Tool->Creat Ruler*. The default

units are "user units" and are in microns. For our inverter, the NMOS transistor has a width of 90n M. So, make sure that the active layer that you have just now drawn is also 90n M in width. Then you repeat the same process but by selecting the "nimplant" layer from the LSW. The active and nimplant layers must overlap and these form the Source and Drain diffusion regions of the NMOS transistor you are trying to create. The following pictures illustrate these two steps.



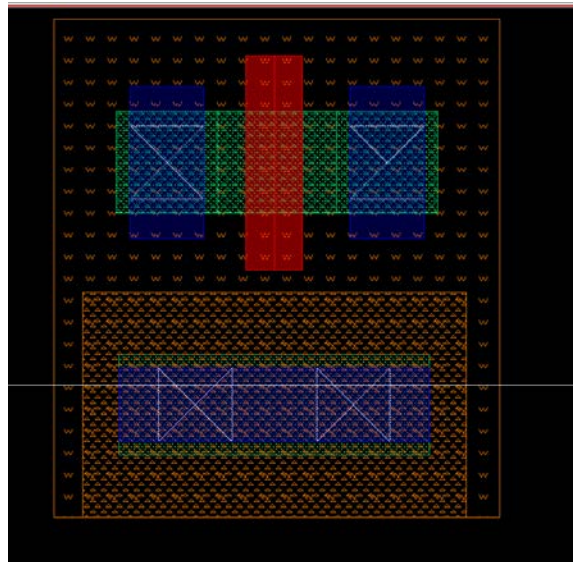
The next step is to draw the gate of the transistor. This is done by selecting "poly" in the LSW and drawing another rectangle to form the gate. Make sure that the length of the gate is set to 50n M to create the transistor that we used for the schematic. Now, we have formed the channel and the source and drain diffusion regions. The next step is to draw a pwell outside the NMOS transistor. Use the same procedure as described earlier to draw the pwell (Note that the minimum well enclosure of active is 0.055um as required by the DRC rules). The resulting transistor is shown in the following picture.



We still have to create contacts for the source, drain and the body terminals of the transistor. This can be done by clicking on *Create->Via* or pressing "o" and then selecting "M1N" (for NMOS) or "M1P" (for PMOS). You can also create contacts to the active layer by selecting "contact" from the LSW and painting a rectangle on the Active layer. Paint two contacts for the Source and Drain regions of the transistor (Note that the minimum spacing between poly and M1 is 0.035um as required by the DRC rules). Now you need to create a body terminal. To do this, click on *Create->via* or press "o". In the window that opens up, make sure that the *Technology Library* is set to **NCSU_TechLib_FreePDK45** and select "PTAP" in the *Via Definition* field. Place the PTAP

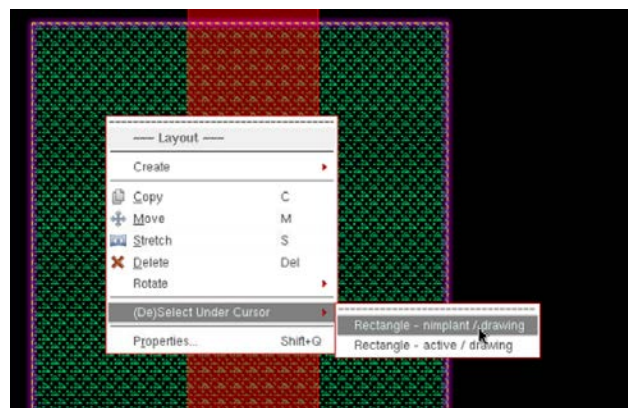
connection in contact with the pwell you have drawn. You can press *Shift+F* to reveal the details inside the PTAP connection.

When you create an instance, you can specify the number of rows and columns, as well as the "Delta Y" and "Delta X", to create an array of the same instances. Array is highly preferred if you have a regular pattern repeating many times, e.g., an inverter chain, a long NTAP containing lots of body contacts, etc. The instance array can make your layout well organized and much cleaner. Draw metal rectangles over the contacts so that you can connect the terminals to other signals in your circuit. Your NMOS transistor is now ready. The final transistor should look like the following picture.



You can create the PMOS transistor similarly. It uses the following layers: **nwell, active, pimplant, poly, metal1 & contact**. Make sure the dimensions of the PMOS transistor match that used in the schematic.

To select any layer from a group of layer in the layout, right click on the group and select (or deselect) from the menu



Sometimes, some other layers are being selected while moving or stretching, and it may require to deselect all. Press Ctrl+D to deselect all, then you can move or stretch the desired ones.

2.2.2 Layout of inverter

Now that you have a PMOS and an NMOS transistor, you are ready to draw a layout of the inverter. Make the rest of the connections by using a poly to connect the gates of the two transistors together. Connect the drains of the PMOS and NMOS transistors using metal 1. Also connect the NTAP and source of the PMOS transistor and the PTAP and source of the NMOS transistor together using metal 1.

Now you need to connect a metal 1 layer to the poly gate. To do this, click on *Create->via* and select "M1_Poly" as the via type. Attach it to the gate and connect a metal 1 rectangle to it.

2.3 Performing DRC of the inverter

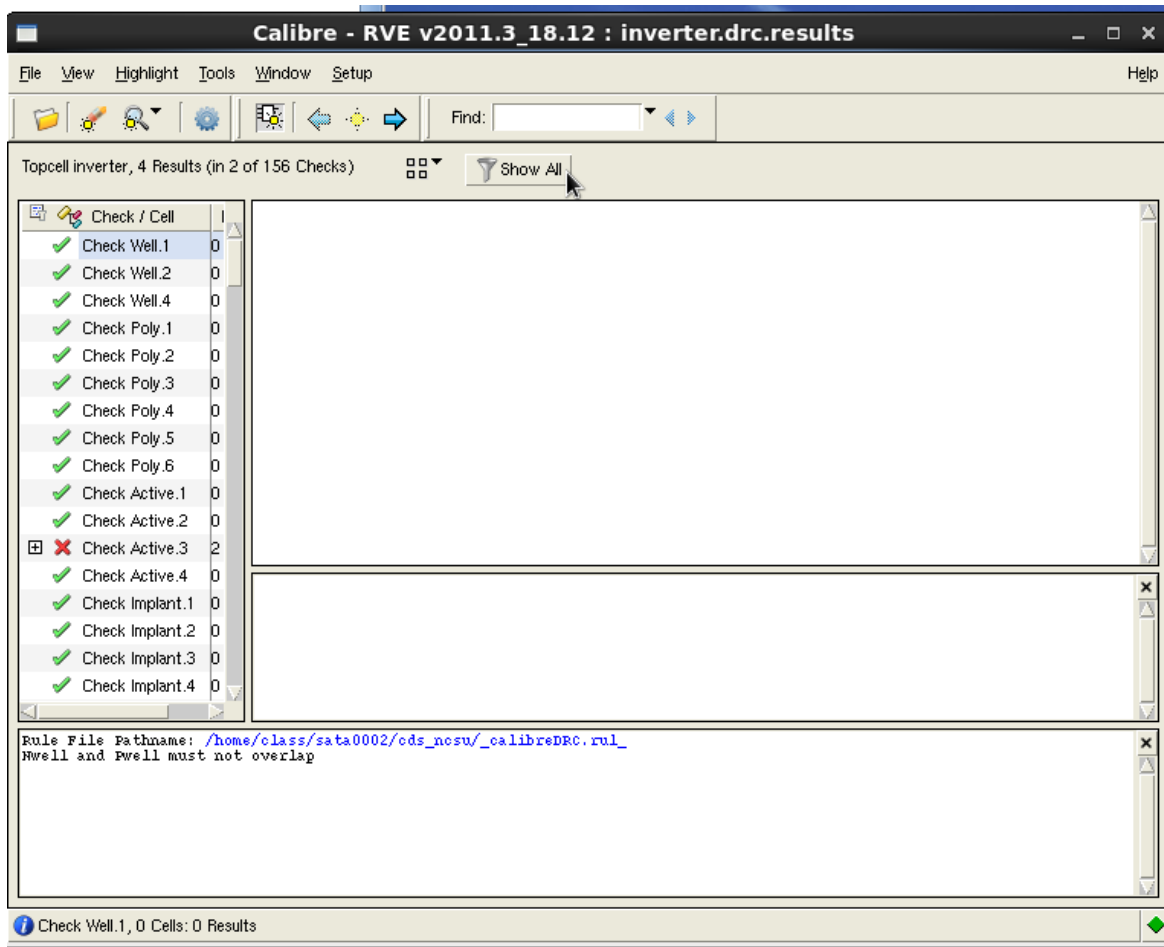
So far, we have drawn rectangles to create transistors and connected them to form an inverter. We now need to verify that the layout has passed all the design rules of the process. To do this, first save the layout and then click on

Calibre->Run DRC from the menu bar on the layout editor. The DRC form appears. Then click on **Run DRC**.

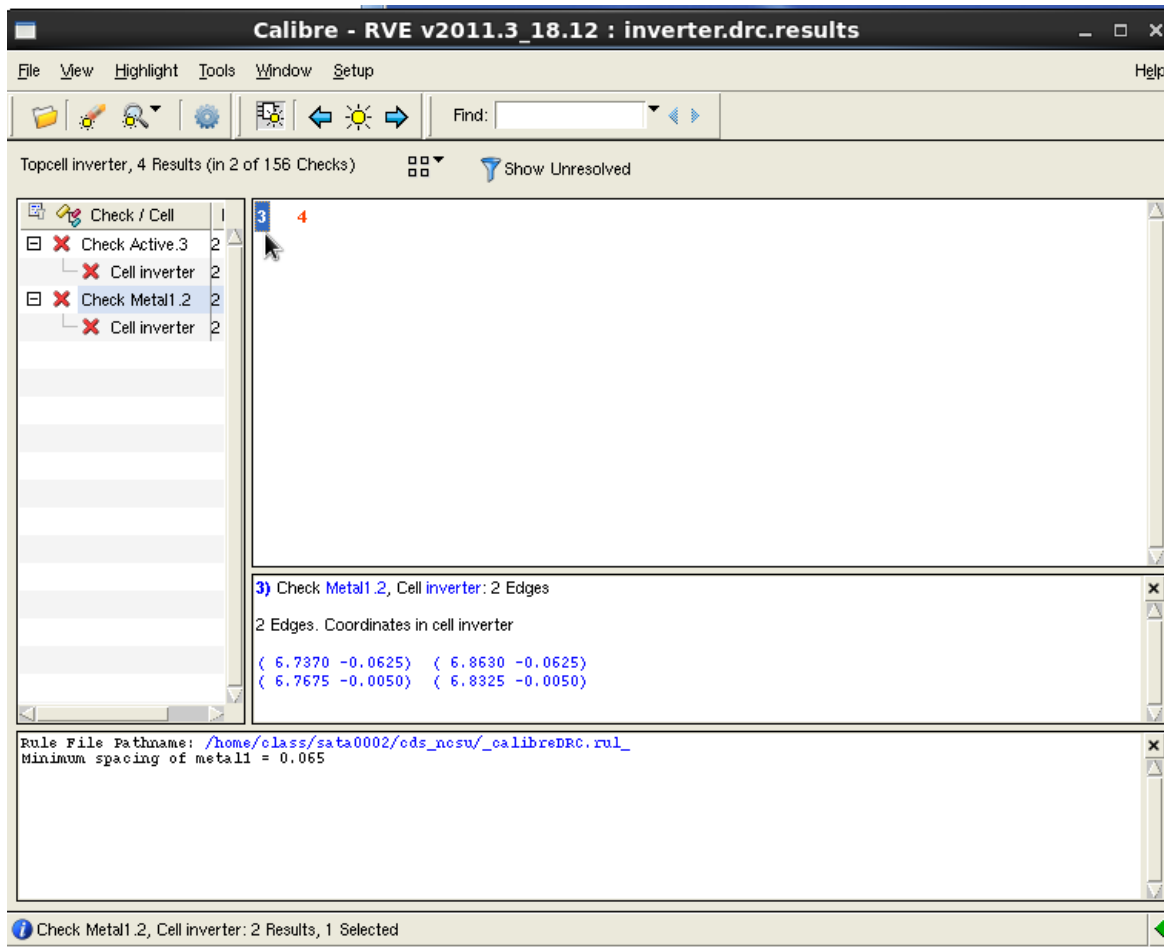
You can ignore any warning messages you get. When CALibre finishes the DRC, it opens up 3 windows: A Calibre results window, Calibre interactive (which is the main DRC form) and a DRC summary report. You can check whether you layout passed all design rules by looking at the Results window.

2.3.1 Viewing and Correcting Errors in DRC

In this section, you will be shown how to find out errors in the layout and to fix them. The following picture shows the DRC results on the inverter that was laid out earlier in this section.



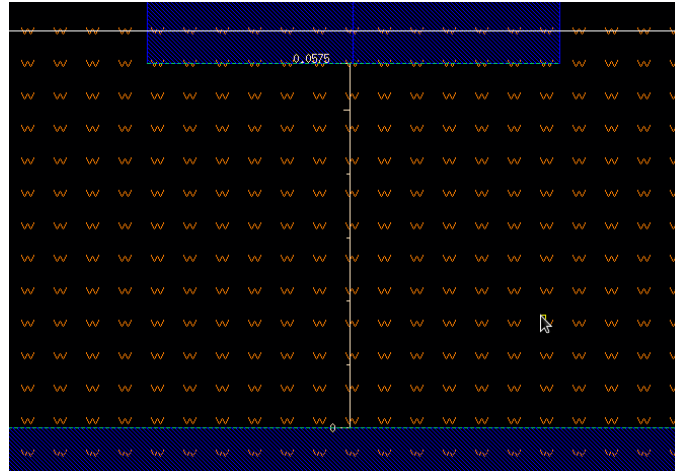
You can click on “Show All” (the mouse pointer is present on the button in the above figure) and select “Show Unresolved”. Now you can see all the unresolved errors as given below.



The presence of a red check mark in the DRC results window indicates that the layout failed the DRC check. The results window give information on what design rule was violated, where it was violated and how many instances have violated the rule. You can also double click the error number (e.g., "3" in this example) to jump to wherever the violation occurs.

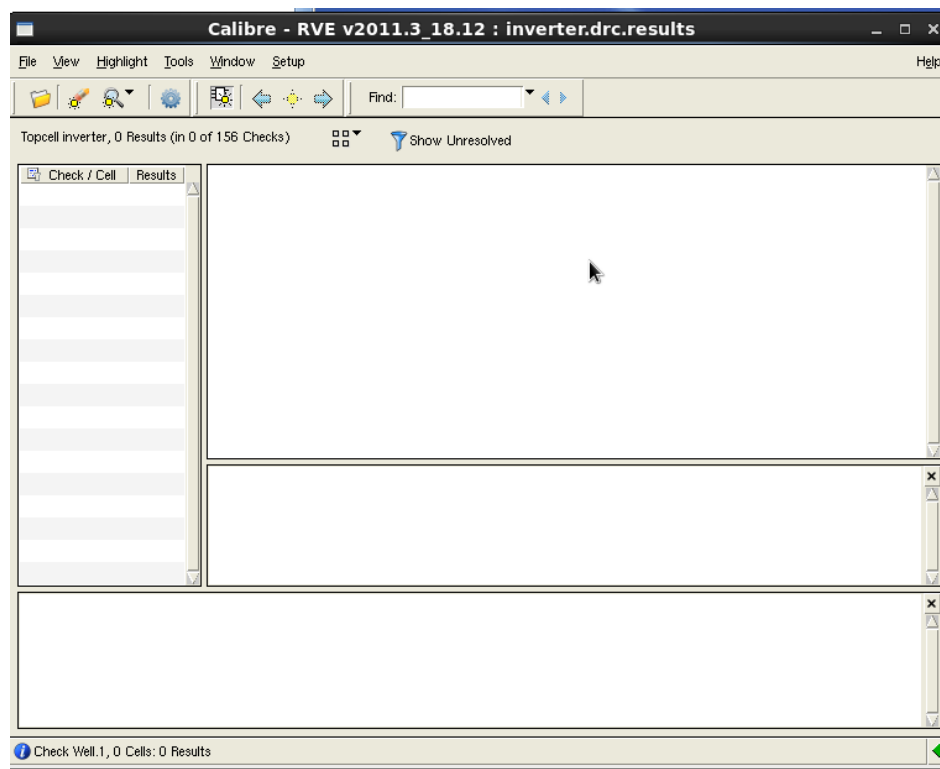
In the "Check/Cell" sub-window in the results window, it lists two kinds of errors and besides that it says how many of those errors are violated. So, we have two errors from either types making it a total of four. Underneath that, it says "Minimum spacing of metal1 = 0.065". This is the design rule being violated and there are two instances in the layout. These can be known by clicking "3" and "4".

The following picture shows a zoomed in version of our poly connections.



You can observe that the two metal edges have been highlighted and their distance is 0.0575u. While according to the DRC it is 0.650u. So, increasing the distance by stretching one of the metal will fix this error.

Similarly once you fix all the errors and re-run DRC, it should come clean with no entry for the “Show Unresolved” option in the DRC result window.

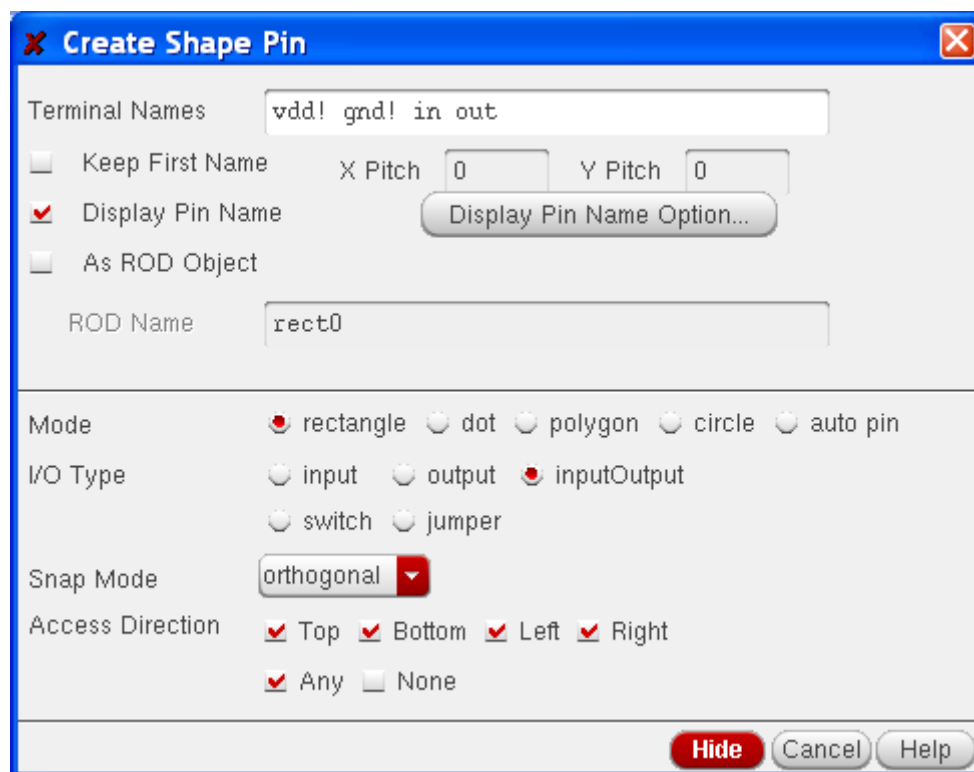


Note that when you are working through this tutorial you may get other errors from DRC. You can get more information about design rules from the link posted earlier in this section or from the NCSU wiki page linked at the bottom of this page. The DRC results window will let you

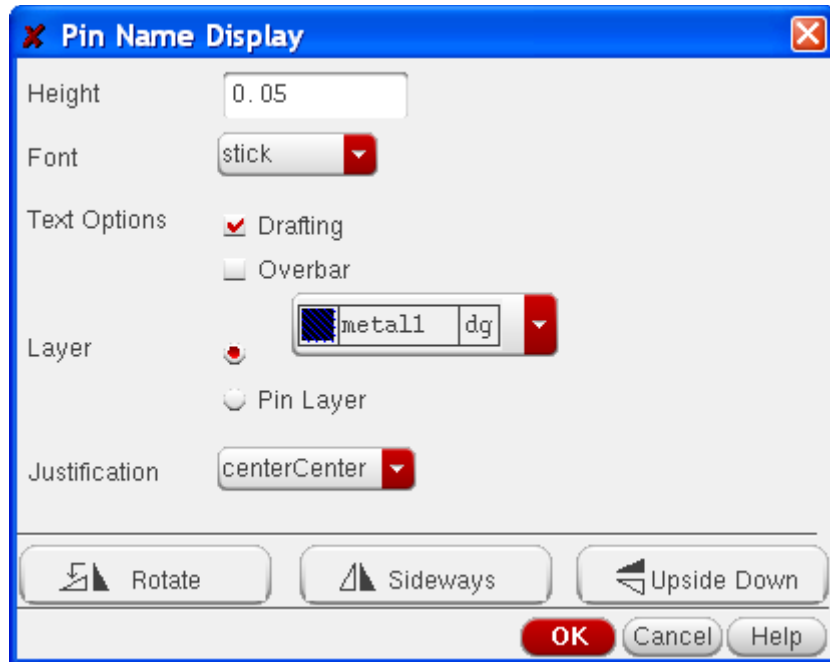
know what rules you have violated. So, in conjunction with the set of design rules in the wiki you can fix those errors.

2.4 Create Pins

Lastly, we need to create pins so that nodes in our layout have names that are human-readable. Before creating pins you should select the layers in which you want to draw the pins. For this example, let's select metal-1 dg layer. Although you can select this layer just before placing them on the layout or edit those layers through properties (shortcut "q") after placing them. Create these pins by selecting *Create->Pin...* You should see a dialog box appear, like the one below. Type the names vdd!, gnd!, in and out in the "Terminal Names" text box as shown below. Select "Display Pin Name". Leave all other options as they are



Next, click the "Display Pin Name Option..." button. You will see another dialog box appear.



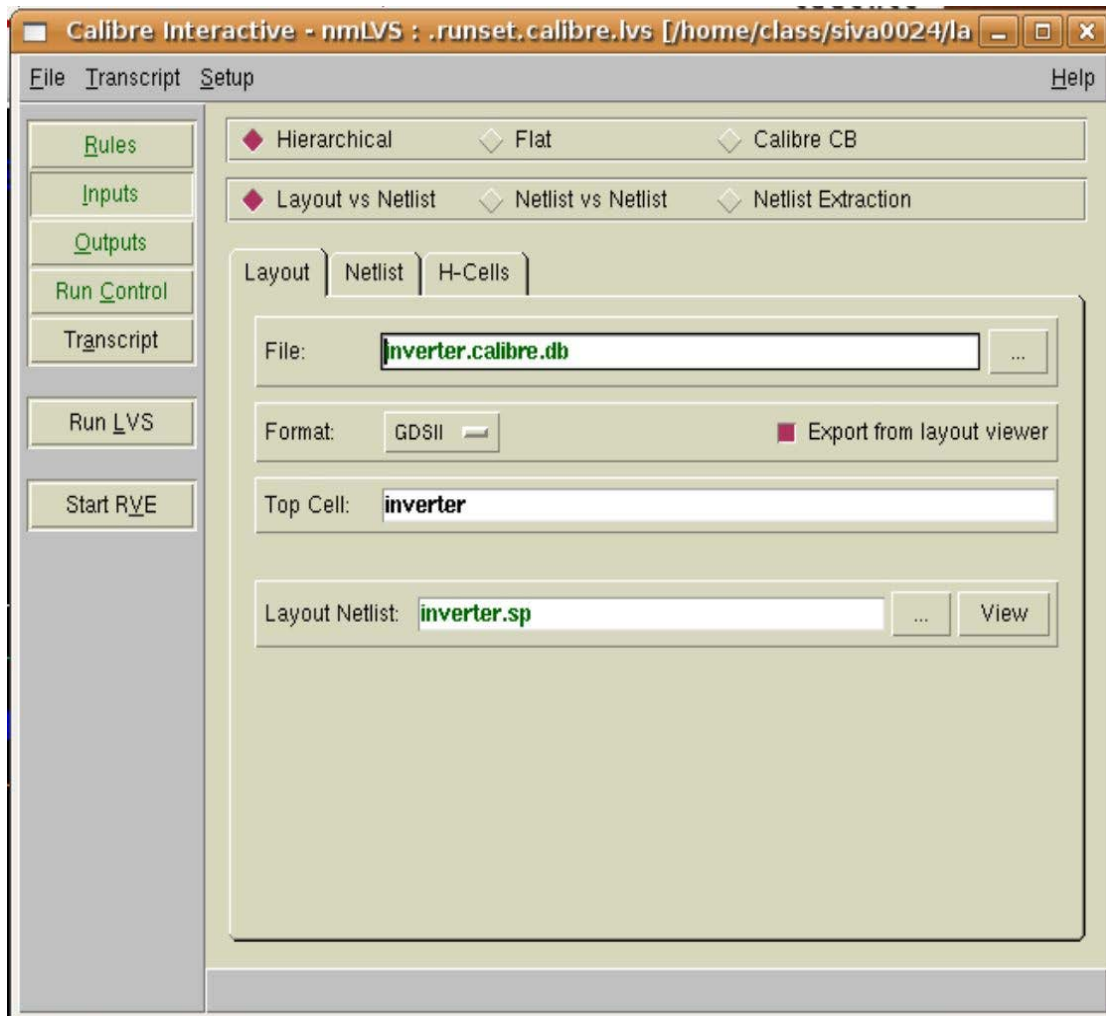
Set the height to 0.05 um and the layer to metal-1 dg. Click OK.

Next, click on layout where you want each pin to be placed. You will need to click three times: twice to create a rectangle for the pin and a third time to place the label. The shape of your rectangle does not matter as long as it only covers area that is already covered by metal1-dg. Re-run DRC to make sure the layout passes all design rules.

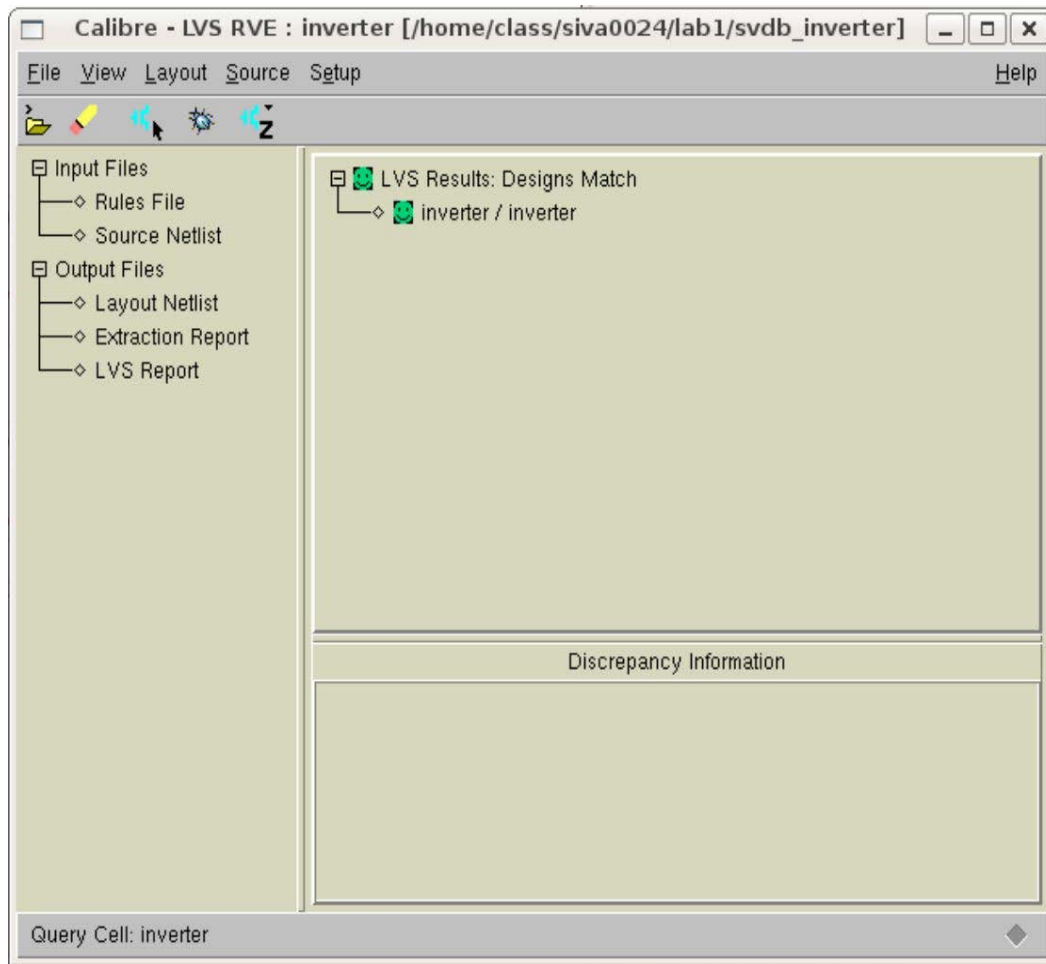
NOTE: Your pin layer has to match with the pin name display layer. Otherwise it will generate error.

2.5 Performing an LVS check of the inverter

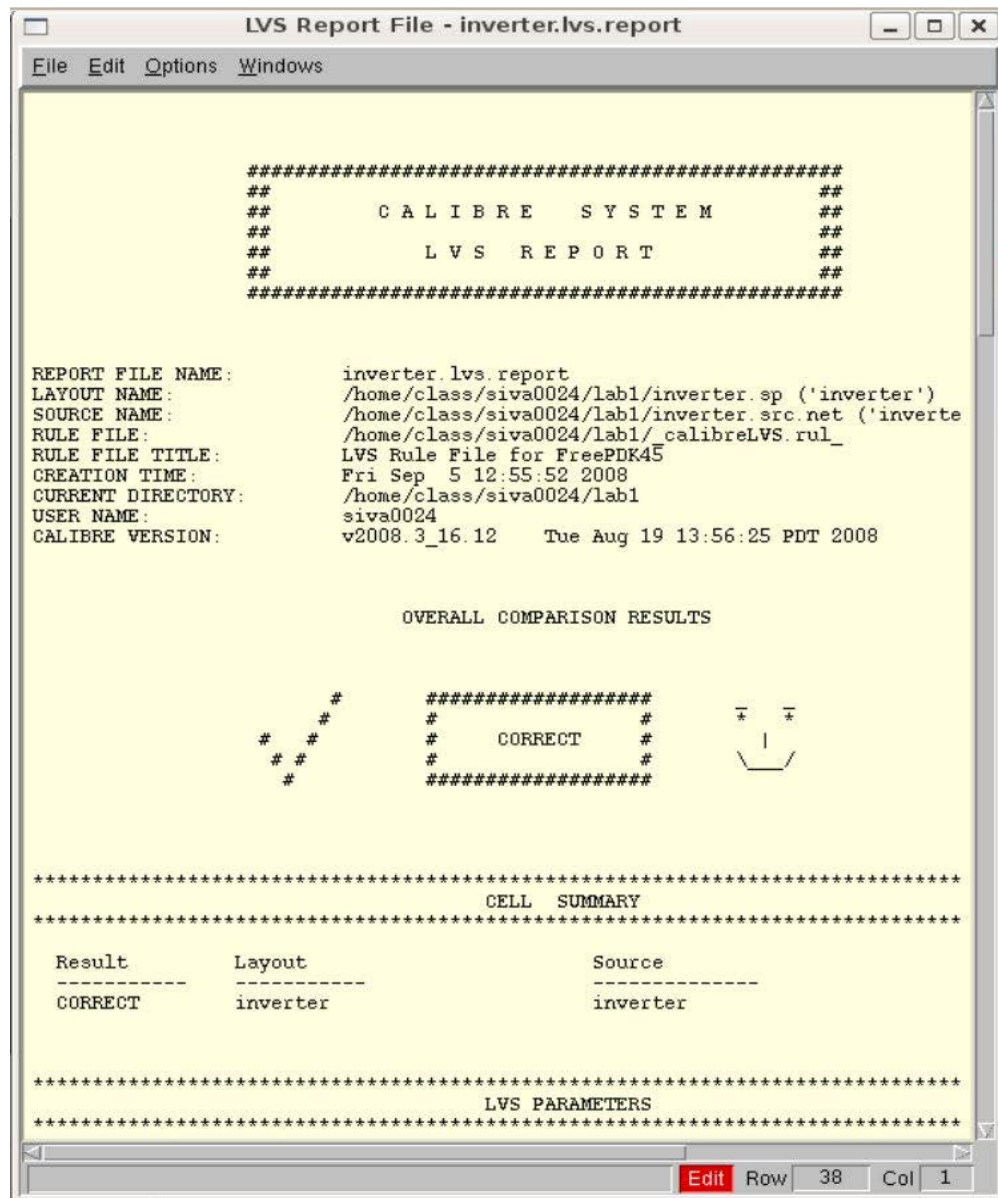
The next step is to perform a layout versus schematic(LVS) check. To perform an LVS check choose *Calibre->Run LVS...*. The LVS form appears as shown below.



Make sure you select the "Export from layout viewer" option under the **Layout** tab and "Export from schematic viewer" under the **Netlist** tab. Under the **Outputs** tab, set the svdb directory to svdb_inverter. Then click on "Run LVS" button. If LVS runs successfully, then you will see the following window with a smilie.

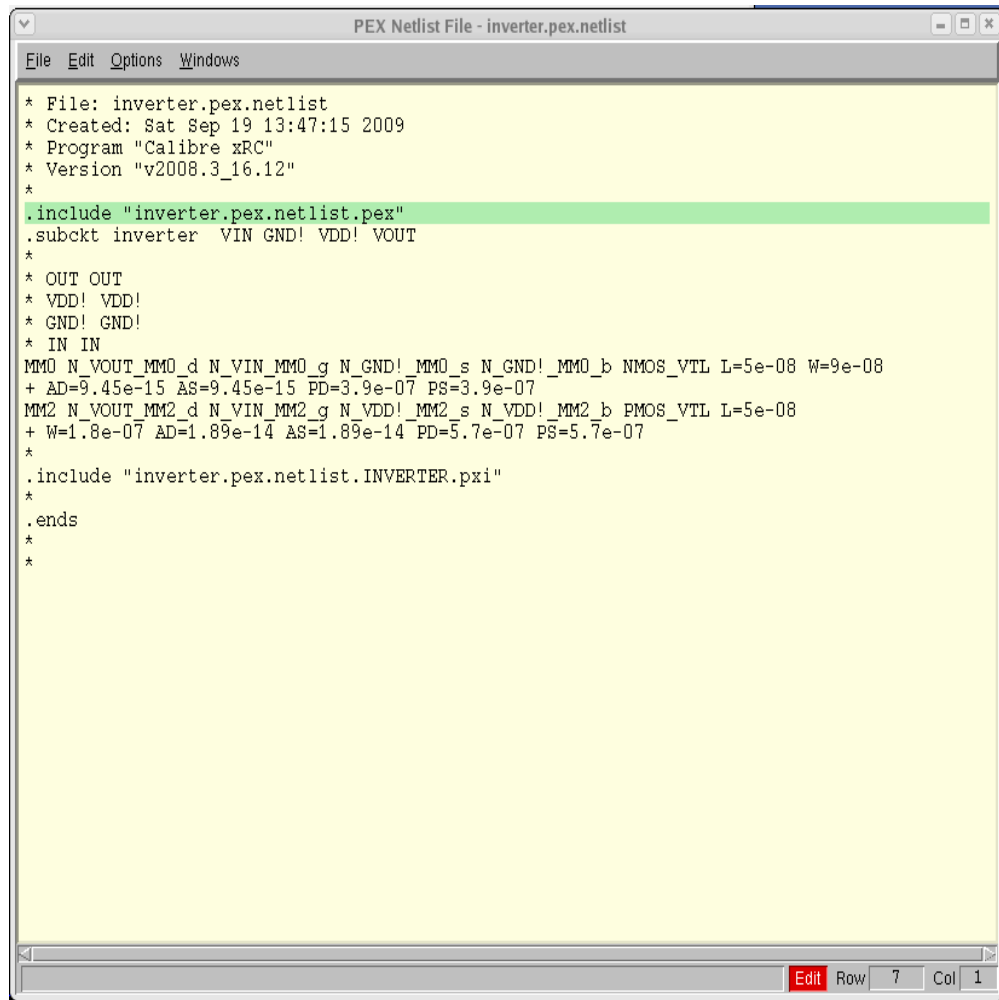


Click on "Transcript" tab in Calibre Interactive - LVS to see the log file. The LVS report is also opened and is shown below.



2.6 Extract Parasitics

To perform a Parasitic Extraction(PEX), choose *Calibre->Run PEX...* The PEX form will appear. You can stick to the default options for now. Click on "Run PEX" button. If PEX runs successfully, you will be able to view the extracted netlist in a pop-up window,



```
PEX Netlist File - inverter.pex.netlist
File Edit Options Windows
* File: inverter.pex.netlist
* Created: Sat Sep 19 13:47:15 2009
* Program "Calibre xRC"
* Version "v2008.3_16.12"
*
.include "inverter.pex.netlist.pex"
.subckt inverter VIN GND! VDD! VOUT
*
* OUT OUT
* VDD! VDD!
* GND! GND!
* IN IN
MM0 N_VOUT_MM0_d N_VIN_MM0_g N_GND! MM0_s N_GND! MM0_b NMOS_VTL L=5e-08 W=9e-08
+ AD=9.45e-15 AS=9.45e-15 PD=3.9e-07 PS=3.9e-07
MM2 N_VOUT_MM2_d N_VIN_MM2_g N_VDD! MM2_s N_VDD! MM2_b PMOS_VTL L=5e-08
+ W=1.8e-07 AD=1.89e-14 AS=1.89e-14 PD=5.7e-07 PS=5.7e-07
*
.include "inverter.pex.netlist.INVERTER.pxi"
*
.ends
*
*
```

The file shown in this window contains the intentionally designed devices only. The files with .pex and .pxi extensions are also included in inverter.pex.netlist (which can be found in your working directory, i.e., ~/cds_ncsu/ in this example).

- The **.pex** file contains one subckt per net: each subckt containing the RC tree structure modeling the net.
- The **.pxi** file contains connections between the parasitic networks i.e. containing the instance calls to net model subckts along with coupling capacitors connecting between these net model instances.

Note: The netlist shown in the pop-up window needs to be modified for simulations. This is because the netlist shown above only defines your circuit (which is "inverter" in this example) as a subcircuit (you can see ".subckt" and ".ends"), but does not initialize a real instance. So if you run simulations on this netlist, you won't see any outputs. You can fix this problem by commenting out the two lines (the line ".subckt ***" and the corresponding ".ends") which define the subcircuit.

It is OK if you do not understand the .pex and .pxi files. The top-level netlist file however, needs

to be understood.

Once you have the extracted netlist, you can simulate it to see how it performs as opposed to the simulation results from the schematic. Refer to [Section 1.6](#) for more information on simulation.

2.7 Layout tips

This tutorial is less focused on the techniques used for layout and is geared mainly towards introducing the reader to tools and the steps involved in creating a layout/design. Nevertheless, here are some pointers to do layout.

- When you have multiple identical instances to add into the layout and these instances will be placed in a very regular pattern, e.g., a ring VCO with several identical inverters, you can add them together as an array to make your design cleaner. To be specific, when adding an instance, change the number of "Row" and "Column" in the pop-up window and adjust "deltaX" and "deltaY" to control the distance between the instances.
- When putting together your layout, place large metal rectangles along the both sides of the diagram as your VDD and GND rails. Try to place all of your layout within these rails. Also, when connecting to these (or any routing connection) it is almost always a good idea to put as many via connections as possible. You can use the same multiple contact placement to make this fast and neat.
- Though it's not really obvious from this example, it is also good practice to try and make your layout as symmetric as possible.
- When routing large layouts it's a good idea to try and keep track of what you are routing with. Route with poly as little as possible since its resistance is higher than metal. It also helps to set some general directions for different layers. For instance, for horizontal traces use metal2, 4, for vertical traces use metal3,5. Metal 1 can be used for both directions. This will help keep your layout neat and organized.

More Information

This tutorial is a watered down version of the more elaborate tutorial developed at NCSU. Interested readers are referred to [NCSU FreePDK Wiki](#) for more details.