

Cadence Virtuoso Schematic Design and Circuit Simulation Tutorial

Introduction

This tutorial is an introduction to schematic capture and circuit simulation for ENGN1600 using Cadence Virtuoso. These courses use the NCSU FreePDK45 library for a 45nm technology. The NCSU library provides the models for a 45nm Bulk-Si technology from Fujitsu (details about the PDK can be found at <http://www.eda.ncsu.edu/wiki/FreePDK45:Contents>).

This tutorial will guide you in the process of designing CMOS circuits using both user defined, transistor-level schematics. It will also walk you through simulating the circuits in Spectre.

In order to launch Cadence Virtuoso (either on the instructional machines or on your laptop), you will need to connect to the Computation and Visualization cluster network at Brown (CCV). For additional details on how to connect to CCV and launch Cadence, please refer to the CCV tutorial accessible from the ENGN1600 course webpage. Note that only the first part of this tutorial about logging on to the CCV machines is relevant to you.

Create your working environment - Design Library

After starting cadence, the first thing to do is create a new library. From the main Virtuoso window, select **Tools > Library Manager ...** This will open the Library Manager (Figure 1) from which we can browse the existing libraries. From the Library Manager window go to **File > New > Library ...**

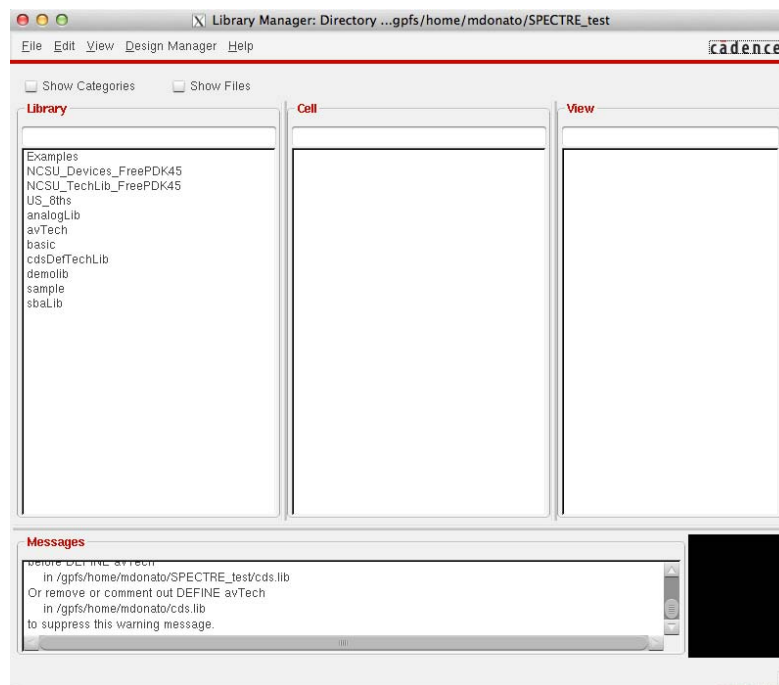


Figure 1 Library Manager

At this point you will be asked to choose a name for our library; for this tutorial use the name Tutorial_lib (Figure 2). When asked which Technology File you would like to use for the library Tutorial_lib, check 'Attach to an existing technology library' (Figure 3), confirm and select NCSU_CellLib_FreePDK45 from the Technology Library list. At this point you should be able to see an instance of the new library in the Library Manager.

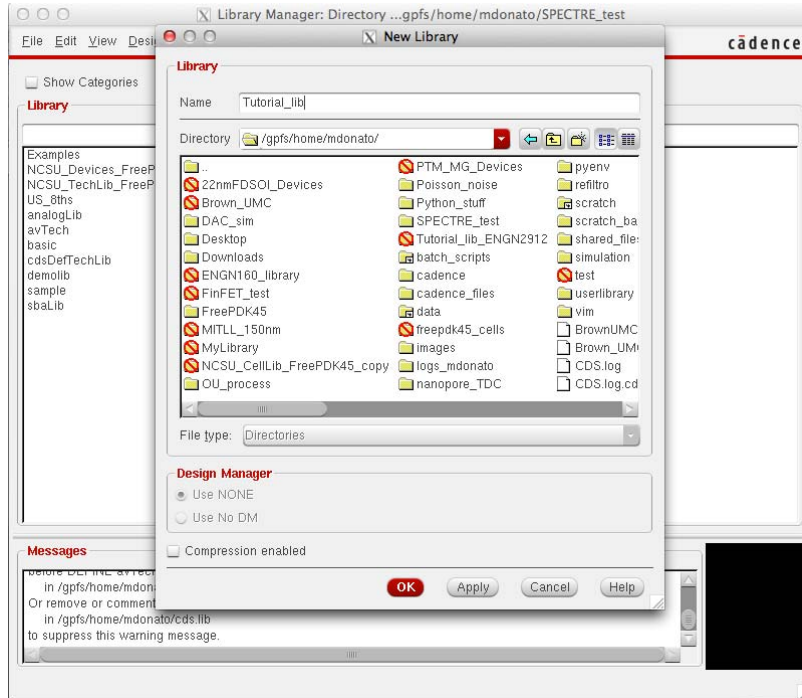


Figure 2 Create New Library

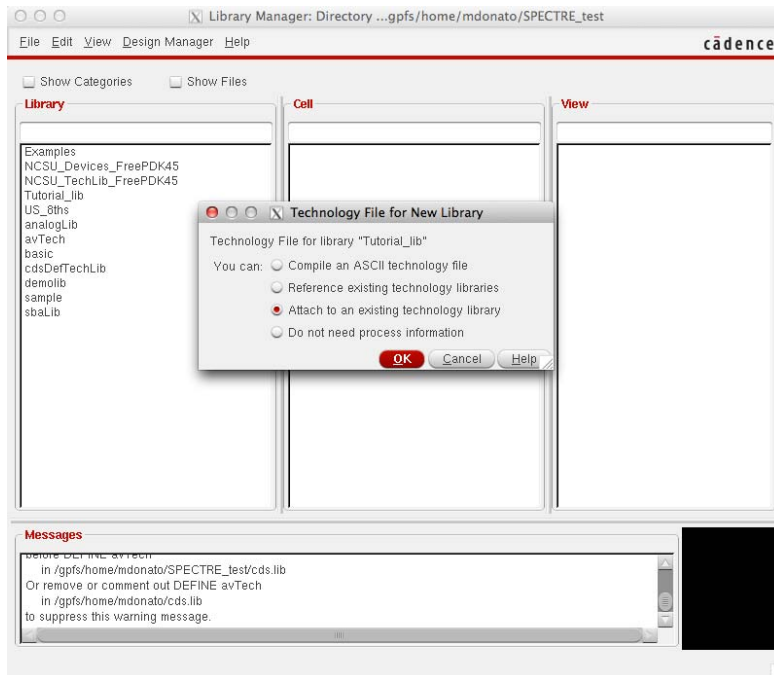


Figure 3 Select Technology file for the new library

We will use this library for saving all the cells used in this tutorial. At this point we are ready to design our first schematic.

Creating an inverter using transistors from the PDK library

Throughout the course, you will be asked to create your own standard cell library. Using a standard cell library allows us to easily create digital circuits starting from a wide variety of common logic gates (inverters, NAND, NOR, latches). The first step in building a standard cell library is designing the schematic of the logic gates at the transistor level. The NCSU_Devices_FreePDK library provides four different technology flavors for both PMOS and NMOS devices: high-performance (VTL), low operating power (VTG), low standby power (VTH) and thick-oxide devices (THKOX) (Figure 13).

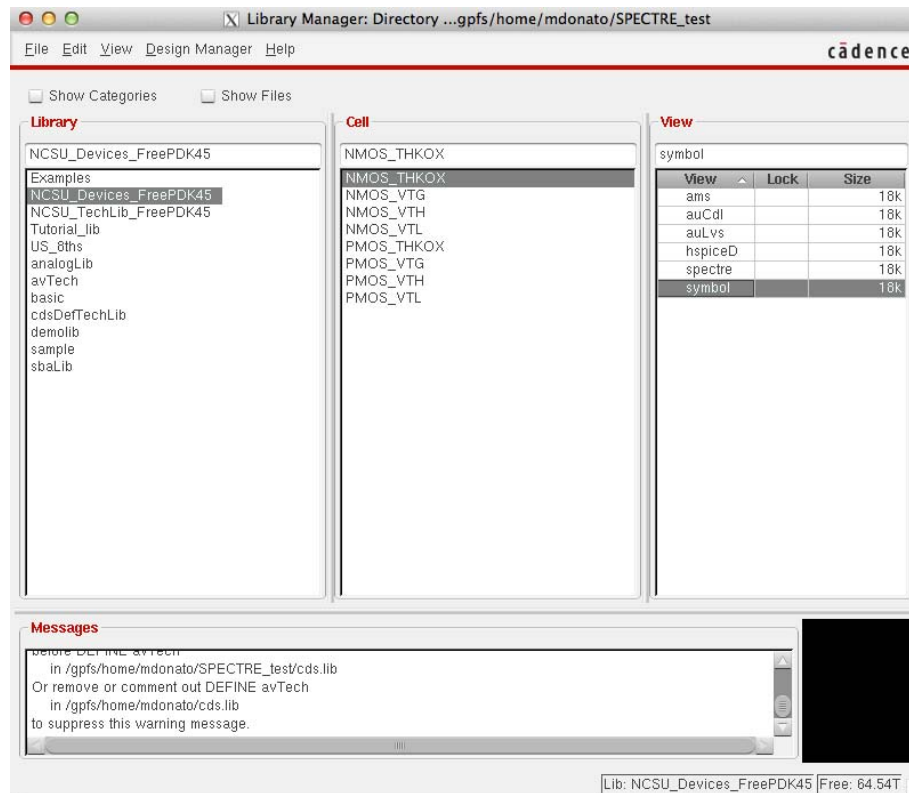


Figure 4 Technology flavors for the FreePDK45 library

This tutorial will describe how to design a standard CMOS inverter using low-standby power transistors (VTH). First, we need to create a new cell view in our Tutorial_lib library. From the Library manager, go to **File > New > Cell View ...** From the **New File** window (Figure 5) you can select the destination Library (Tutorial_lib), the Cell name (let's call this cell low_power_inv) and the View (schematic).

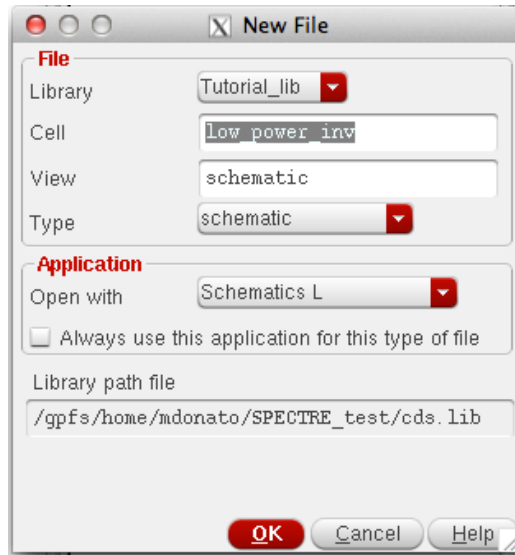



Figure 5 Create new schematic view

After confirming the creation of the new cell, a Schematic Editor window will appear (if a license warning message appears, click Yes). We now need to place and connect the components required for our inverter. Click on the Create Instance button  or alternatively press 'i' to open the add instance window (Figure 6).

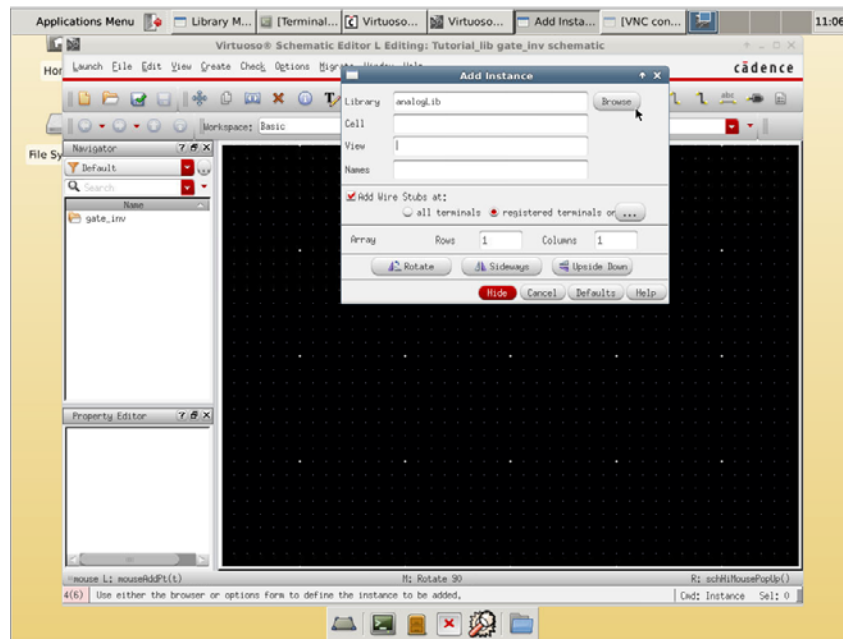






Figure 6 Add new instance

The first step in creating the new schematic consists of adding the instances for a NMOS and a PMOS transistor. In the **Add Instance** window, type NCSU_Devices_FreePDK45 in the library field and then select NMOS_VTH symbol view. Before placing the instance in the schematic editor check that the width and length for the transistor are set properly (length = 50nm and width = 150nm as shown in Figure 7). Repeat the same procedure for the PMOS transistor but this time setting the width to 300nm. After having placed the transistor, add the vdd and gnd symbols from the analogLib library and connect the terminals together using the wire tool (). As we want to be able to interface this circuit with other circuits, we need to add input and output ports. The input/output pins are created by clicking on the Create Pin button  or by pressing 'p'. When the Add Pin window appears, we need to select the Pin Name and the Pin direction (input or output). For example, the settings for the input of the inverter will look as in Figure 8. The complete schematic should look similar to the one in Figure 9. After checking and saving the schematic () go to **Create > Cellview > From Cellview ...** Confirm the settings in the two following windows (Cellview from Cellview and Symbol Generation Options). This will open a Symbol Editor as in Figure 10. By clicking on Check and Save, a symbol view of our custom inverter will be added to the library. With a little patience you can redraw the symbol (Figure 11) to look like an inverter using the drawing tools 

This symbol can be used to add inverters in a different circuit schematic, as we will see in the next section.

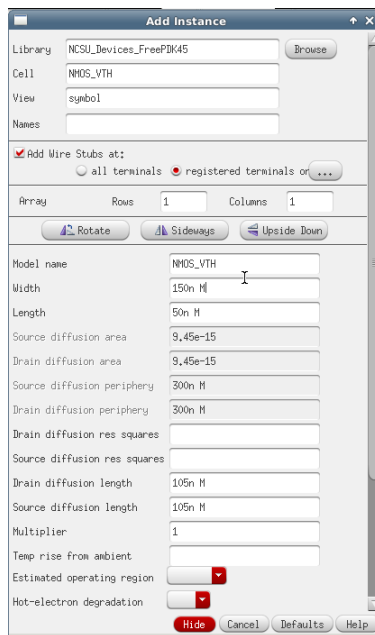


Figure 7 Edit width for a NMOS transistor

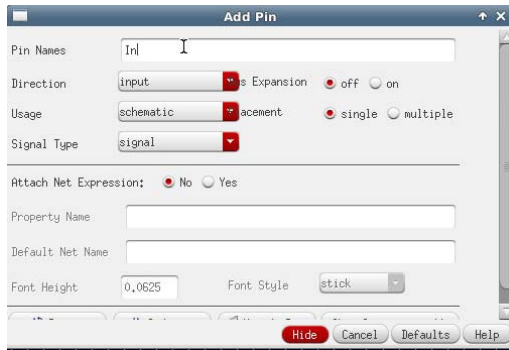


Figure 8 Add pins to the schematic

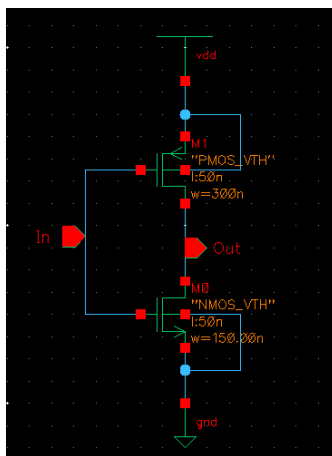


Figure 9 Transistor schematic for a CMOS inverter

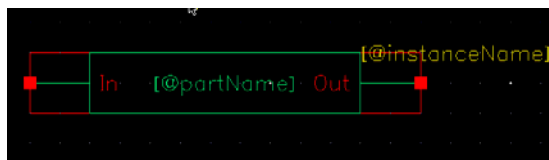


Figure 10 Original symbol view

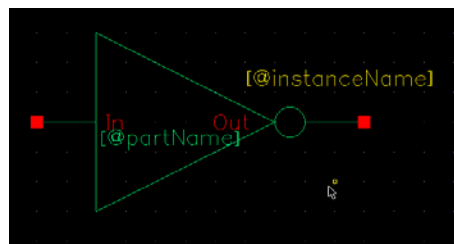



Figure 11 Better symbol view

Create a test circuit schematic using our standard library cell

Once we have set up our standard library we can combine cells together and create bigger circuits or add the components we need for a circuit simulation.

First, select the Tutorial_lib library and go to **File > New > Cell View ...** Create a new cell named gate_inv. Once the Schematic Editor has started, select Create Instance. From here select Browse and using the Library Manager select your newly created library Tutorial_lib and from there go to low_power_inv cell and select the symbol view by double clicking. After confirming the selection, click on an empty spot in the schematic editor for placing the instance of low_power_inv. Press Esc to stop the creation of new instances.

After placing the inverter symbol we need to create the input and output pins and the power supply connections. As before, use the Create Pin button (or press 'p') and connect the pins to the symbol of the inverter by using the wire tool.

Note that even though no power supply connection has been specified yet, the cell low_power_inv includes implicit connections to the vdd! and gnd! nodes. We need then to explicitly connect these nodes to a DC voltage generator. This can be done by separately instantiating the cells for vdd, gnd and vdc from the analogLib library. The first two cells represent the connection to vdd! and gnd! nodes, while vdc is a DC voltage generator. When a vdc instance is created we can specify the DC voltage across its terminal using the corresponding parameter as in Figure 12. The parameter window can always be recalled by selecting the desired instance and pressing 'q'. For this example, set the DC voltage to 1. In order to provide correct power supply to the inverters, these cells should be placed and connected as shown in Figure 13. Once we are done with the design of our schematic we can save the result and check for errors (i.e. dangling wires or misnamed pins/ports) by pressing the Check and Save button 

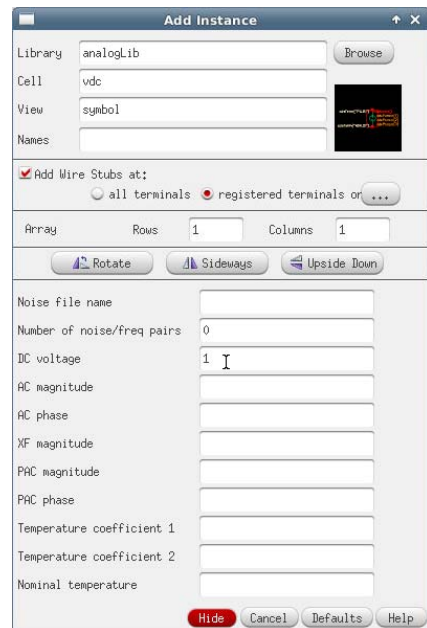


Figure 12 Setting voltage value for VDD generator

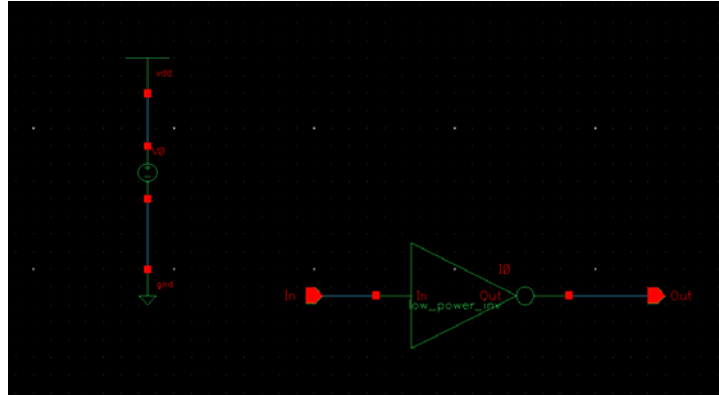


Figure 13 Inverter subcircuit with power supply generator

Circuit simulation with Spectre

Spectre is the circuit simulator in the Cadence tool suite (i.e., the Cadence version of SPICE). Circuit simulation settings are created using the ADE (Analog Design Environment) tool. For this tutorial we will characterize the custom inverter designed in the previous section.

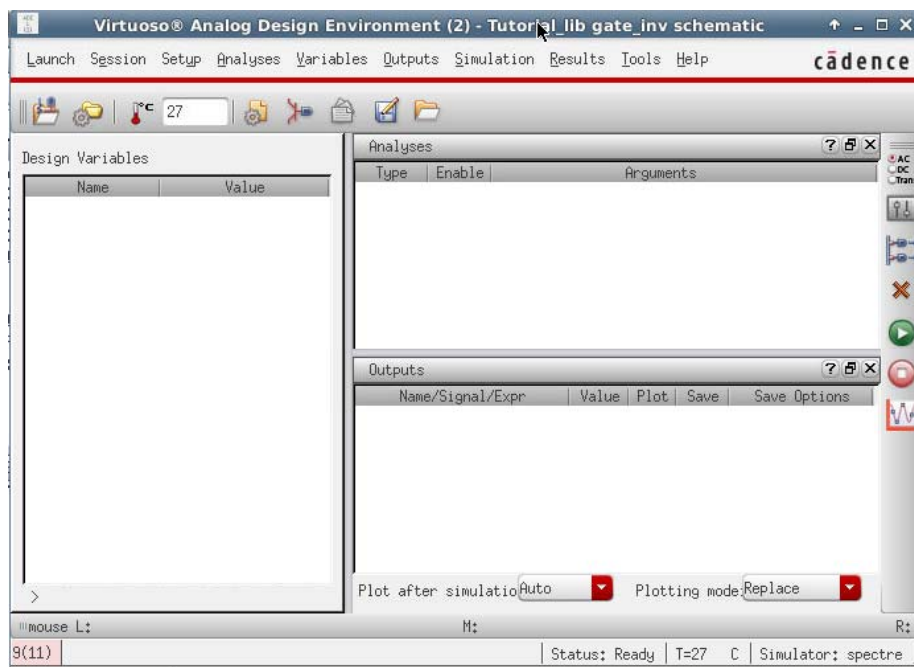



Figure 14 Analog Design Environment window

In order to start ADE for our simulation examples, from the schematic view of the gate_inv cell, go to **Launch > ADE L** (Figure 14).

Since we started ADE L from the gate_inv schematic, it will automatically refer to this design. First, we need to assign the models for the transistors in the schematic; click on the Model libraries button 

then click on the browse button (Figure 15) and select the file at the following path:

```
/gpfs/runtime/opt/cadence/shared/FreePDK45/ncsu_basekit/models/hspice/hspice_nom.include
```

This will select the nominal corner library. The other two available files are used for fast and slow corner libraries (where fast and slow refer to the relative threshold voltage variations).

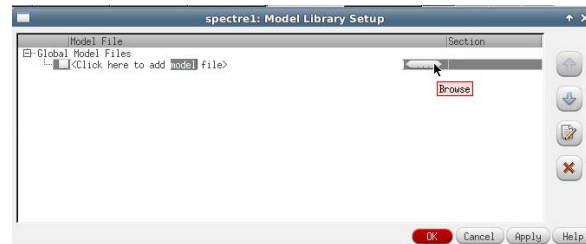




Figure 15 Model file selection window

Apply the changes and return to the main window. Next we need to setup the stimuli for our circuit. We will first perform a DC sweep to extract the voltage transfer characteristic of the inverter. Click on the Stimuli button  and set the analog stimulus as in Figure 16:

1. Select dc from the Function menu;
2. Input Vin in the DC voltage field;
3. Check the Enable flag;
4. Apply changes;

Note that we are assigning to the DC voltage a variable (Vin) rather than specific value so that we will be able to perform a DC sweep. For the simulator to recognize the variable, click on the Edit Variables  button and write Vin in the Name field and assign 1 to Value (Expr) (this value will be overwritten during the DC sweep). Click on Add and return to ADE main window.

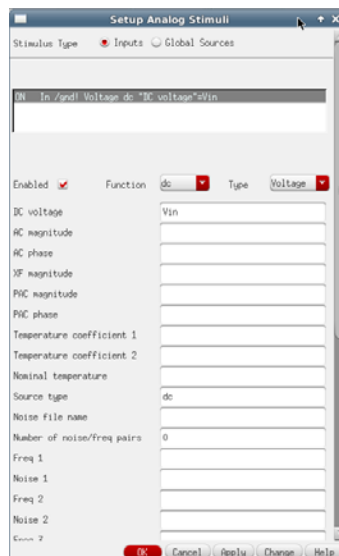


Figure 16 Setup stimulus for DC sweep analysis

We now need to specify which analysis we want to perform. Click on the Choose analysis button and setup the parameters for a DC simulation as in Figure 17.

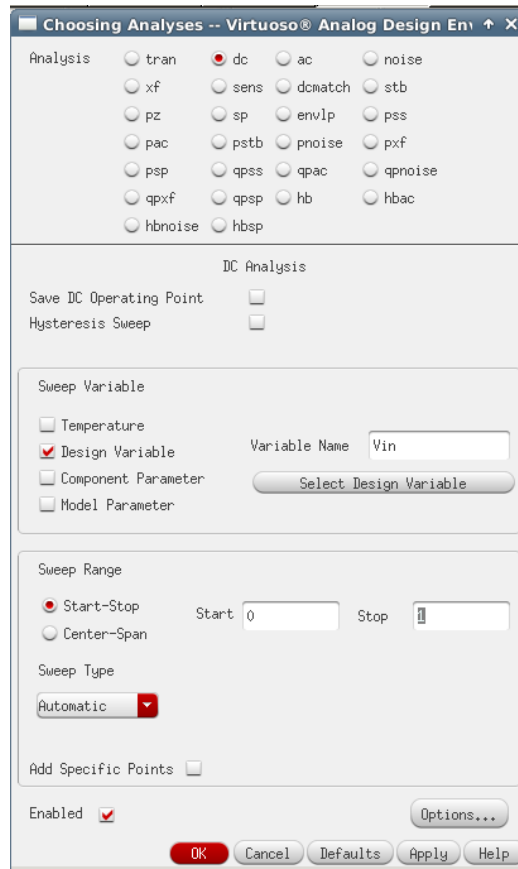



Figure 17 DC sweep analysis settings

As a last step we need to select the signals we want to be plotted in the output. Click on the Setup Outputs button  and click on From Schematic. At this point you should see the schematic view from which you can select the output node by clicking on the output net (wire). After the selection, the net should be highlighted as in Figure 18 (the color may vary).

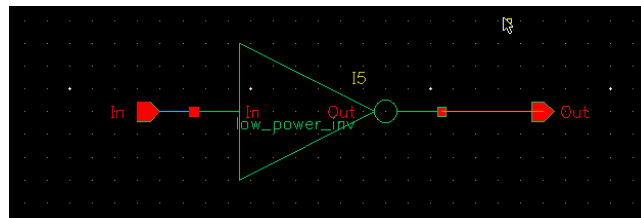




Figure 18 Select output analysis nodes

We can now run the simulation by clicking on the Run button 

If everything runs without any error, at the end of the simulation the results will be plotted as in Figure 19.



Figure 19 Output plot for a DC sweep analysis of a CMOS inverter

DC analysis is useful for characterizing the static behavior of a circuit. The dynamic behavior characterization requires to run a transient analysis. Before setting up a new simulation, we need first to modify the schematic of our circuit and add a load on the output node. Open the schematic view for the gate_inv cell. Using the Create Instance tool  select the analogLib library and then the symbol view for the cap cell (capacitor). The capacitance value can be set to 1fF as in Figure 20. Connect one terminal of the capacitor to the output node and the other terminal to gnd as in Figure 21.

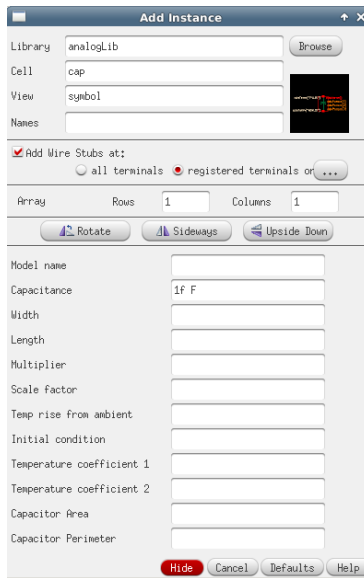


Figure 20 Insert 1fF capacitor

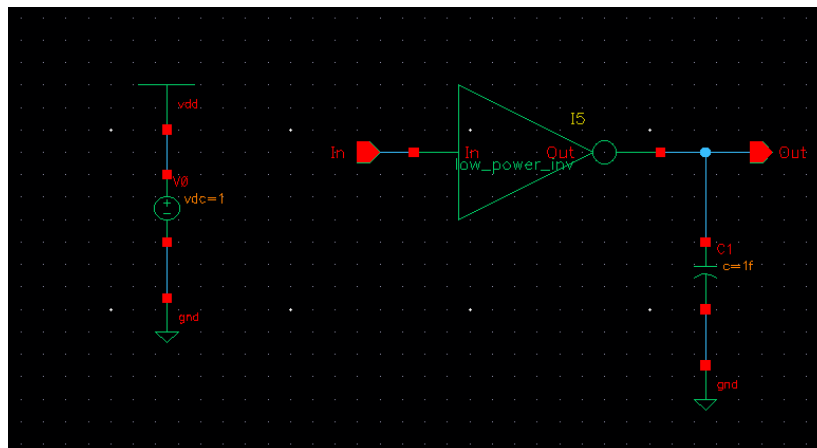





Figure 21 Schematic of inverter with capacitive load

After Check and Save  return to the ADE window. Going from a DC simulation to a TRAN simulation will require adopting a different stimulus and setting the proper analysis. First, setup the analog stimulus using a pulse function as in Figure 22 and apply the changes. Then, change the simulation settings by creating a new analysis setup; click on Choose analysis  and set the parameters as in Figure 23. Since for the transient analysis we want to be able to compare input and output signals, you should add the input node using the Setup Outputs tool  as done before for the output node. Finally, uncheck the dc analysis in the main ADE window. If everything is set up properly, your ADE window should look as in Figure 24.

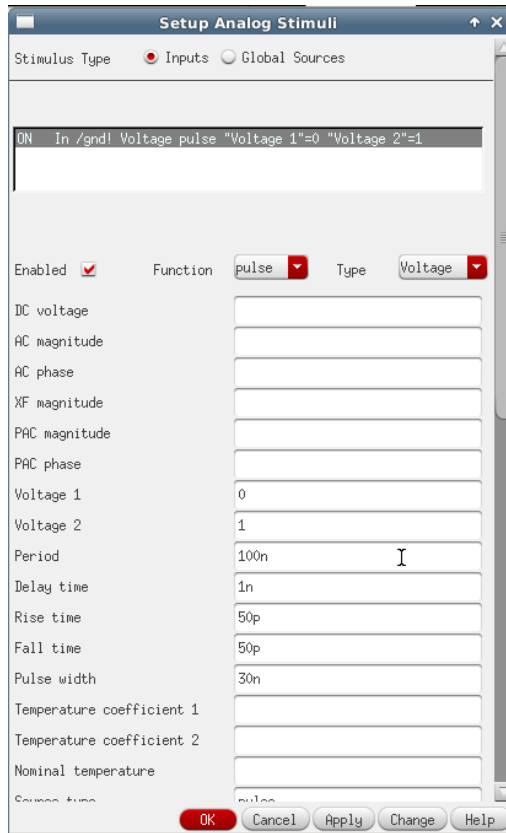


Figure 22 Setup stimulus as pulse function for transient simulation

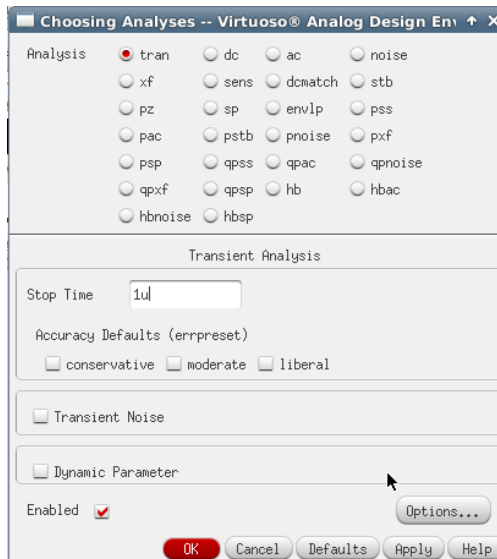


Figure 23 Transient analysis setup

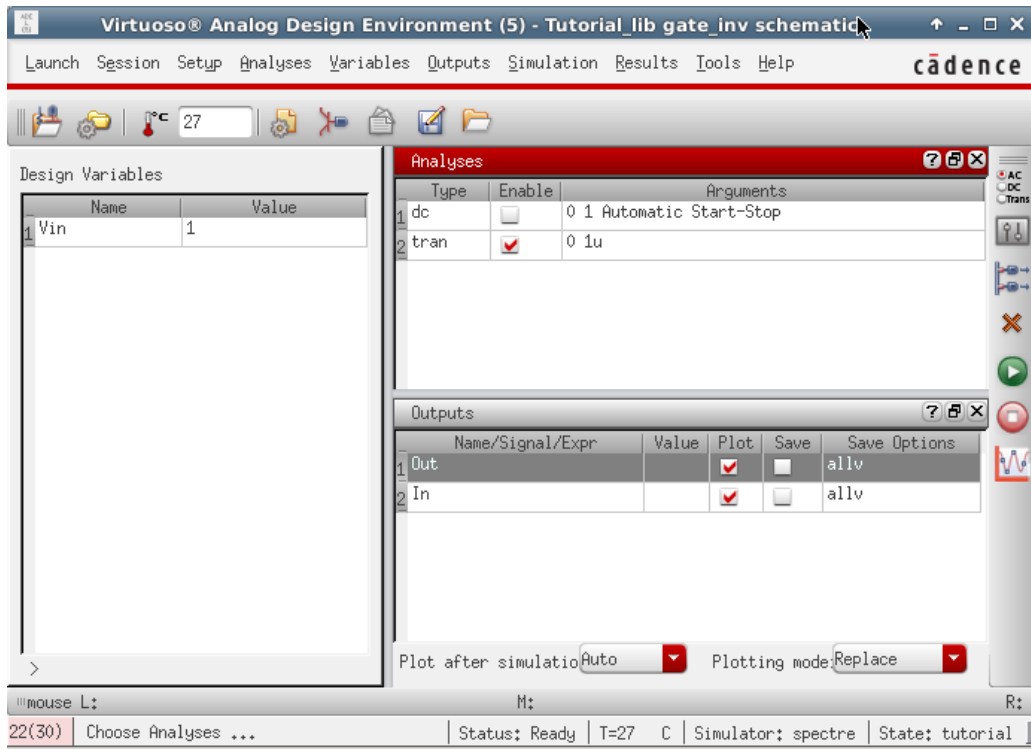


Figure 24 ADE window for transient simulation

After running the simulation, the output plot should show two overlapping signals. To better distinguish between the two traces go to **Graph > Split Current Strip**. This will generate a plot as in Figure 25.

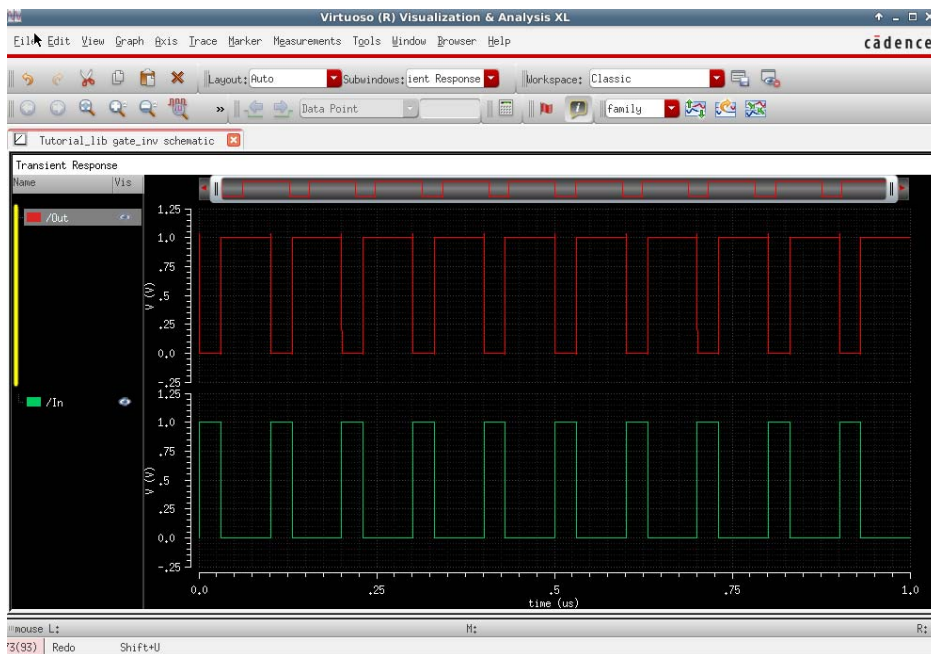




Figure 25 Plot of input and output trace for a single inverter


We now want to get the time characteristic of the inverter as a function of the load capacitance. We can set ADE to run parametric simulations i.e., we can run the transient simulation for different values of a specified parameter. Since we want to evaluate the output for different loads we will have to parameterize the capacitance value. First open the gate_inv schematic, select the load capacitor and press 'q'. Replace the Capacitance parameter value with Cload and confirm the changes. In ADE, go to the Edit Variables  tool and select Copy From. This will add Cload to the list of Design Variables. Set the Value (Expr) to 1f and confirm the changes. We now want to set the simulator so that it will output the propagation delays (both for a rising and falling transition of the input signal). These values are not automatically generated at simulation time; therefore we need to create the expressions for computing the delays. This can be done by using two of the functions in the ADE calculator: getValue() and cross(). Go to the Setup Outputs  tool and create the following setup:

1. Set Name (opt.) to tp1;
2. Input the following line in the Expression field: `(cross(getData("/Out" ?result "tran") 0.5 1 "falling") - cross(getData("/In" ?result "tran") 0.5 1 "rising"))`
3. Click Add;

Repeat the procedure for a new output variable tp2 using the following expression:

```
(cross(getData("/Out" ?result "tran") 0.5 1 "rising") - cross(getData("/In" ?result "tran") 0.5 1 "falling"))
```

For the first example the first cross function will evaluate at the time the Out signal (getData("/Out" ?result "tran")) will cross the threshold (set at 0.5 V) for the first falling edge.

Once we have set the output variables we need to setup the parametric analysis. Go to **Tools > Parametric Analysis ...** and set the parameters as in Figure 26 for running 5 different iterations with values from 1f to 1p. While in this case the values are automatically generated, you can specify which values you want to use in the Inclusion List. Start the simulation on the selected values by clicking on 

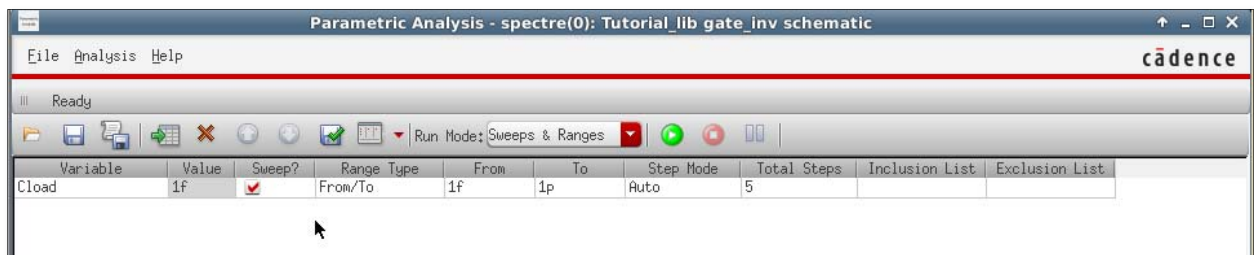


Figure 26 Parametric Analysis settings

The output should give you two plots: the first plot contains the two delays tp1 and tp2 as a function of Cload and the second contains input and output signals for different values of Cload as a function of time.

Note: Two tips for improving the visualization of the results:

1. For the transient simulation, you can zoom in on a particular transition by shrinking the time window to the desired time step as in Figure 27
2. The delay functions are plotted as straight lines connecting the actual computed points. You can visualize the single points by clicking on the plotted lines twice and select points/Solid/Fine in the Type/Style field. The result should look as in Figure 28

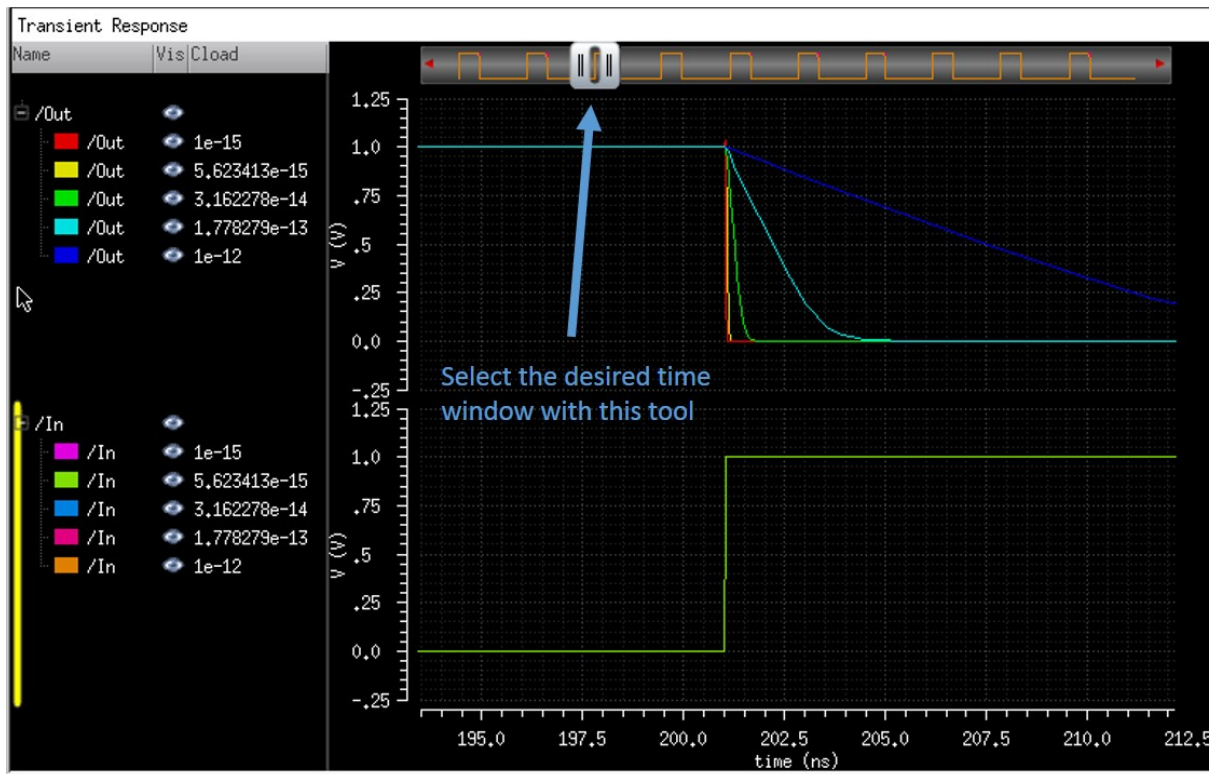


Figure 27 Select desired time window

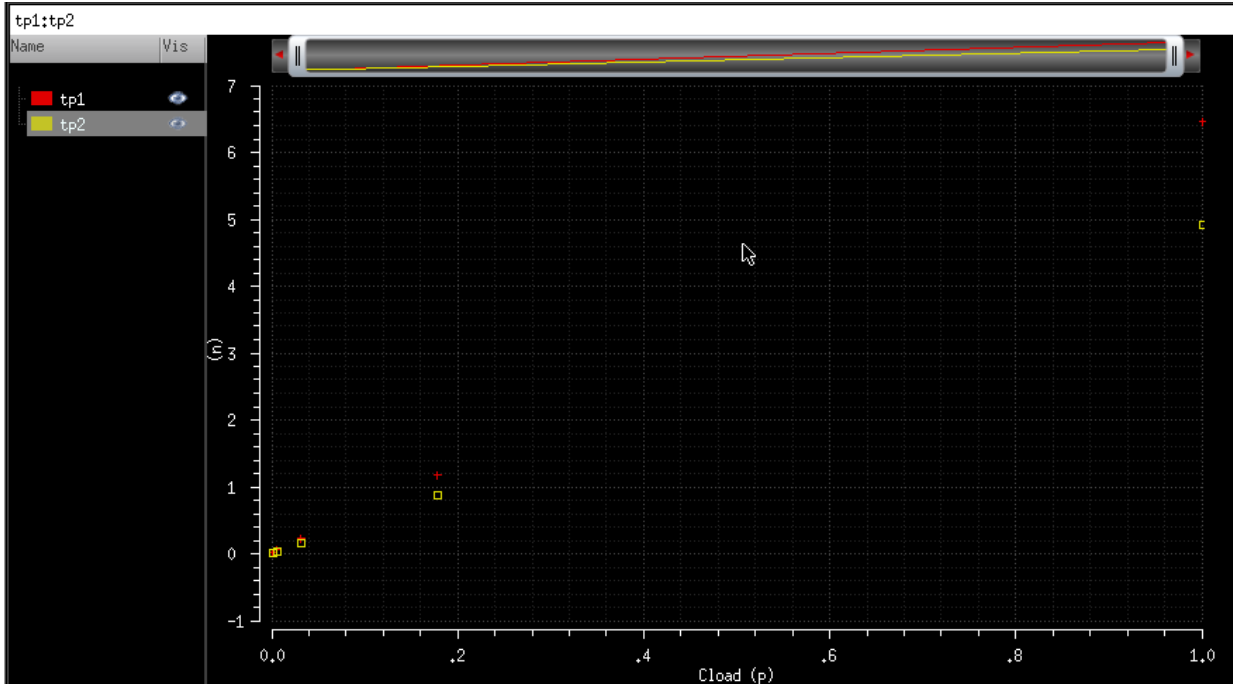


Figure 28 Plot single points from parametric simulation results

Cadence Virtuoso Layout Editor and Mentor Graphics Calibre Verification

The design schematic allows us to explore and simulate the circuit design and have a first idea of the performance of the circuit. The next step in the design flow is the layout. By drawing the layout of our circuit we can verify that the area requirements are met but also improve the circuit simulation by adding parasitic components to the netlist (these components are in fact, dependent on the physical realization of the circuit and cannot be determined only by its schematic description). Therefore, the layout design is both a necessary step towards the physical fabrication of the circuit and a means for improving the accuracy of the circuit simulations.

In this tutorial we will create the layout for the inverter gate in the Tutorial_lib. From the Library manager, select the low_power_inv cell and create a new Cell View with layout type (Figure 29).

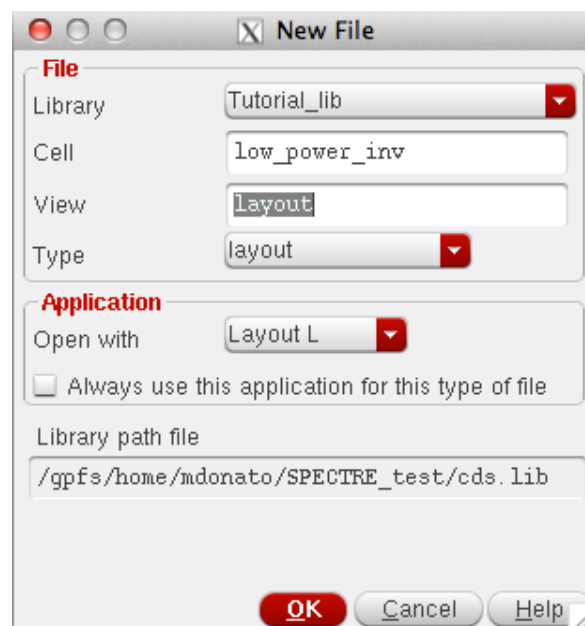


Figure 29 Create new layout cell view

This will open the Layout L editor window (Figure 30). On the left side you should be able to see a list of layers associated with the 45nm PDK. We will now design the layout for the PMOS and NMOS transistors and connect them together to form a CMOS inverter.

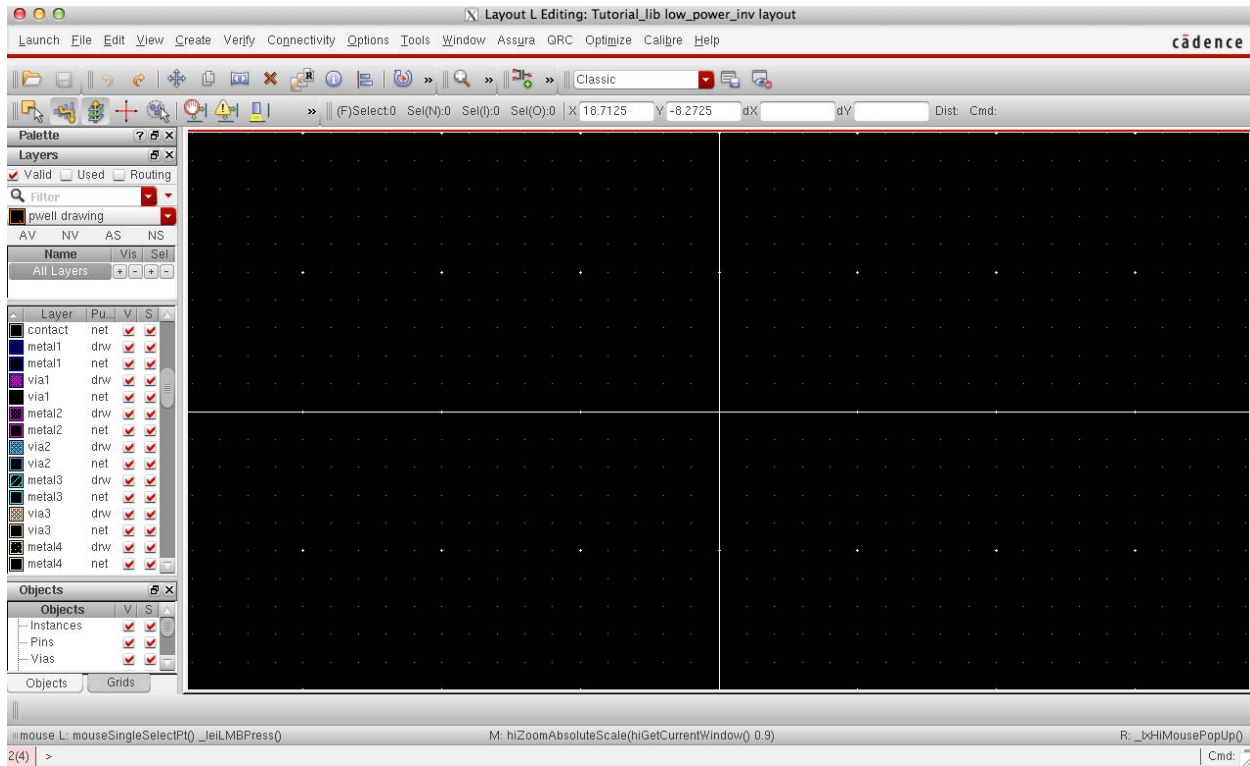


Figure 30 Layout L editor window

PMOS layout design

The PMOS sizes as defined in the schematic are $W=300\text{nm}$ and $L=50\text{nm}$. When drawing the layout, you are not required to follow a specific order in drawing the layers. However, you should check the design rules for minimum size and spacing. The width of the transistor is given by the size of the **active** layer edge parallel to the **poly** layer that defines the gate. Since usually the transistors are drawn orienting the **poly** gate vertically, we will start by drawing an **active** area with height equal to the W of the transistor:

1. Select **active** from the list of layers on the left, making sure to select the one for drawing (drw);
2. Press 'r' and click once in the black drawing window. Move the cursor until the 'dY' box in the top shows '0.300' (the sign depends on which way you move the cursor relative to the first point) and 'dX' is roughly twice the size of 'dY'.
3. Once your rectangle has the desired size, click again to complete the shape.

The PMOS is fabricated using a **pimplant** on a **nwell**. The **pimplant** overlaps with the **active** layer. Select **pimplant**, and following the three steps above, create a layer exactly on top of the **active** layer. While you are drawing, you can use the scroll wheel in your mouse to zoom in and out to improve the accuracy. At this point you should have something that looks like Figure 31.

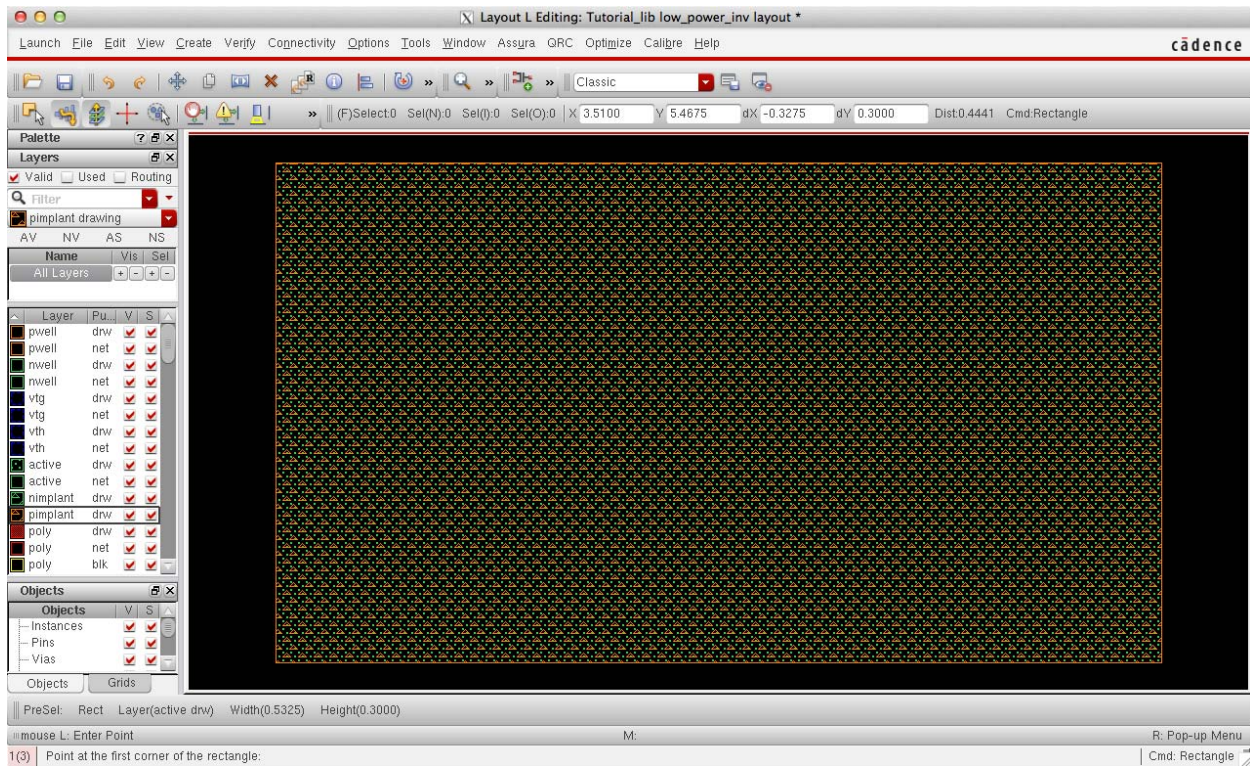


Figure 31 Active and pimplant layers for a PMOS transistor

In order to create the transistor channel we need to place the **poly** layer. As before, create a rectangular shape but this time the width ('dX') should be exactly 0.050 and the height ('dY') should be bigger than the transistor width. In order to get the right size for the height we will need to use the design rules. Place the layer roughly in the middle of the **active/pimplant** layers. If you need to adjust the position of your layer, press 'm' and click on the layer you want to move. You will notice that you can adjust only in one direction. If you need to adjust the both vertical and horizontal position, you will need to repeat the operation twice. The layout should now look as in Figure 32.

While laying out a circuit, it is very important to follow the design rules in order to both properly design the devices but also minimize the area required for a circuit. As an example, consider the design rule POLY.3 which states that the *Minimum poly extension beyond active* should be 55nm (a list of the design rules can be found at <http://www.eda.ncsu.edu/wiki/FreePDK45:Contents>). Let's go back to the layout and zoom on the top portion of the drawing, where the **poly** extension is. A good way to check for design dimension is by using the ruler (press 'k'). The ruler can be drawn as any other layer: click once at the edge of the **active/pimplant** layer, move the cursor to the top corner of the **poly** layer and click again when the length is equal to 55nm (if you are too close to the **poly** edge it will lock to its corner). The result should look as in Figure 33. In my case, the **poly** extension is too short and would fail the design rule check (DRC). To fix the problem, we can stretch the layer (press 's' and click on the edge that you want to move). You can repeat the procedure for the bottom side of the layout.

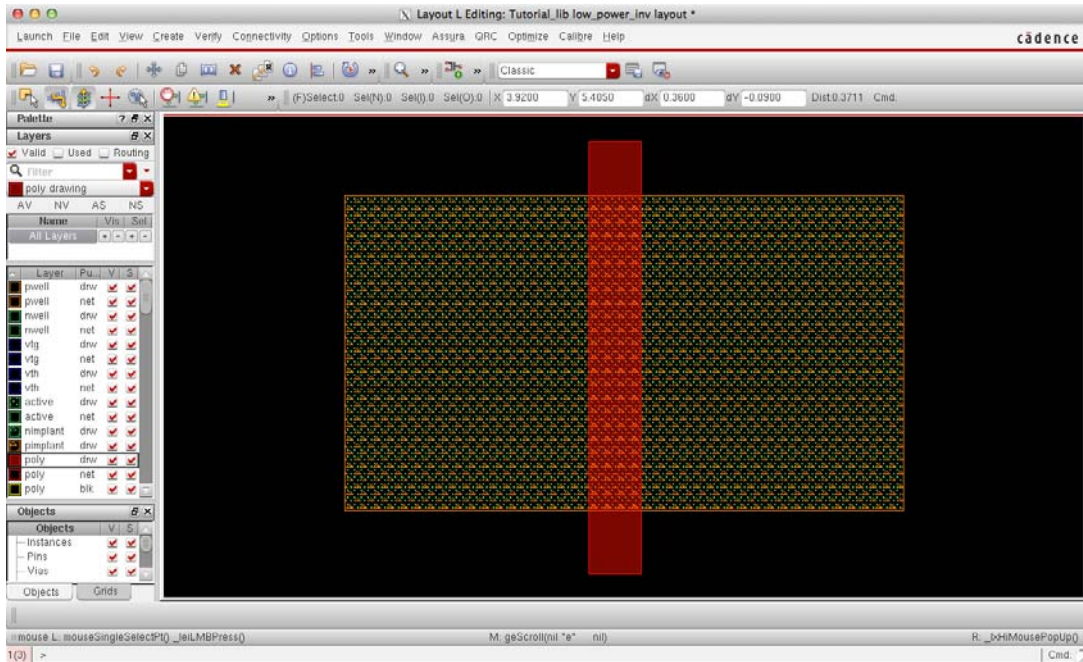


Figure 32 Poly layer

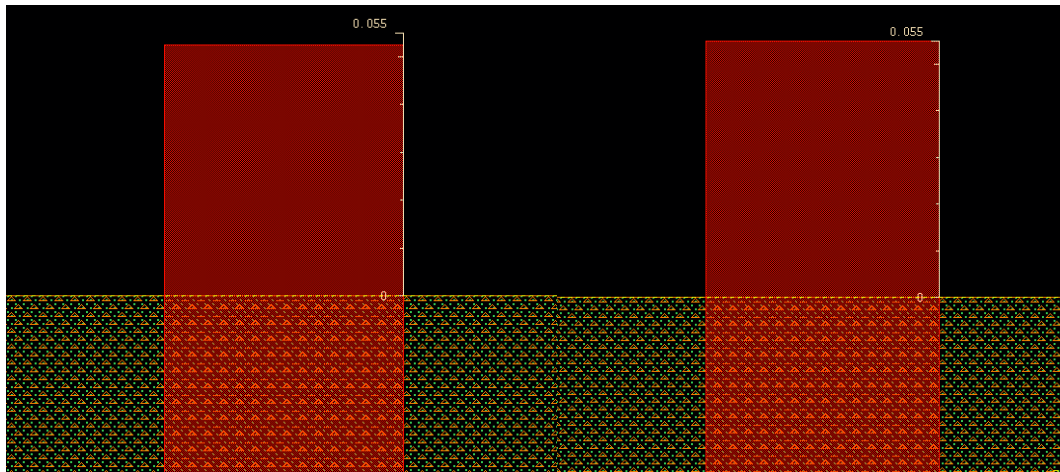


Figure 33 Fixing Poly layer extension using ruler

We will now place the contacts for drain and sources. By following the design rules you can adjust the dimension of the **active/pimplant** layers for minimum area (see Figure 34). A good way to follow design rules is to place the rulers as guidelines before drawing the layers. Figures 34-36 show the additional steps for adding the **nwell** layer and the **metal1** contacts.

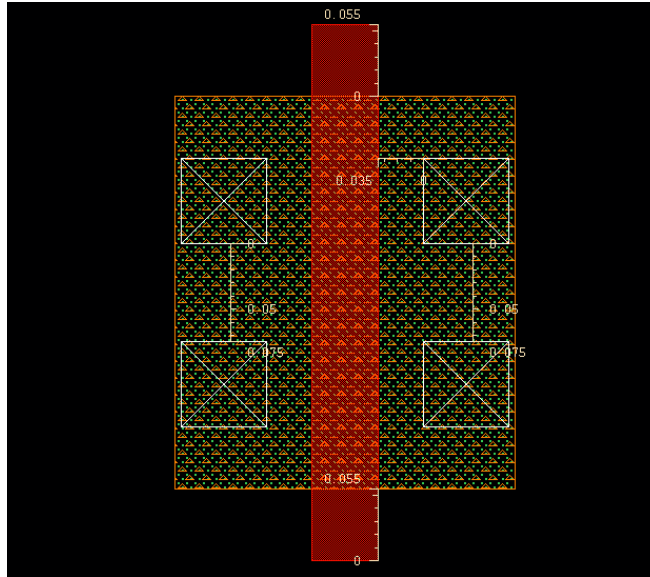


Figure 34 Add contacts for source and drain

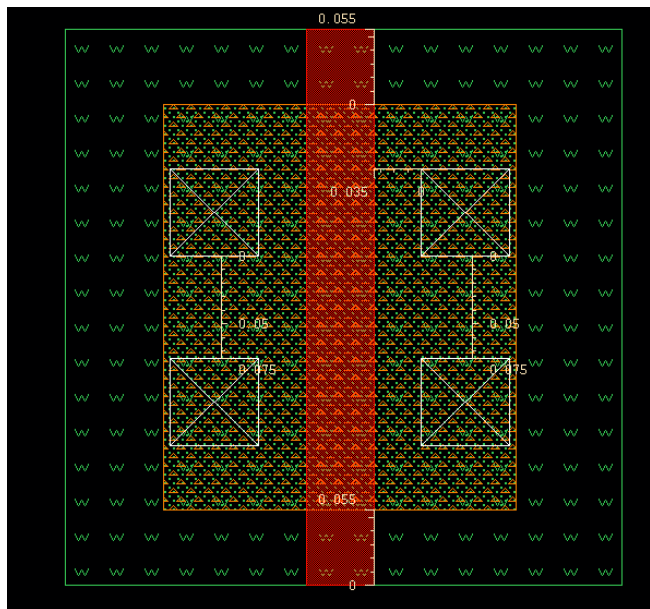


Figure 35 Adding well

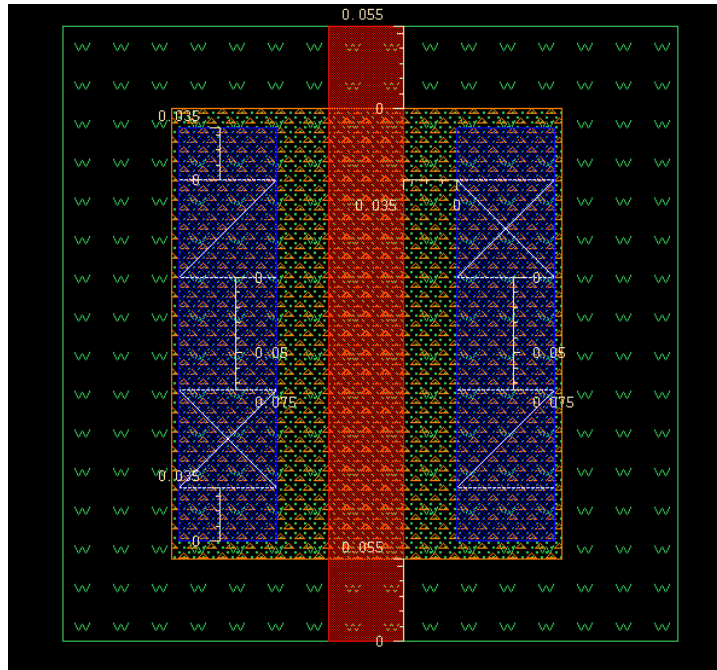


Figure 36 Adding metal1 contacts

The layout shown in Figure 36 is already a functional PMOS transistor. However, it does not match yet the transistor used in the schematic. In fact, if we left the layout as is, we would have a PMOS_VTL device. In order to get a PMOS_VTH device as in the schematic, we need to add a High threshold implant layer (**vth**) (overlap with the **nwell** layer) as in Figure 37.

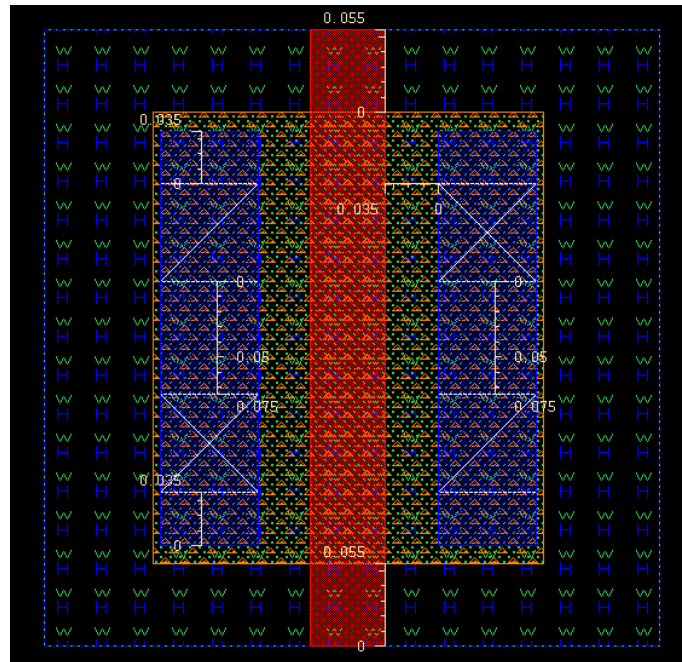


Figure 37 Complete layout for a PMOS_VTH transistor

You can repeat the same steps to design the NMOS transistor (remember that this time the width should be 150nm). The two transistors together should look like in Figure 38.

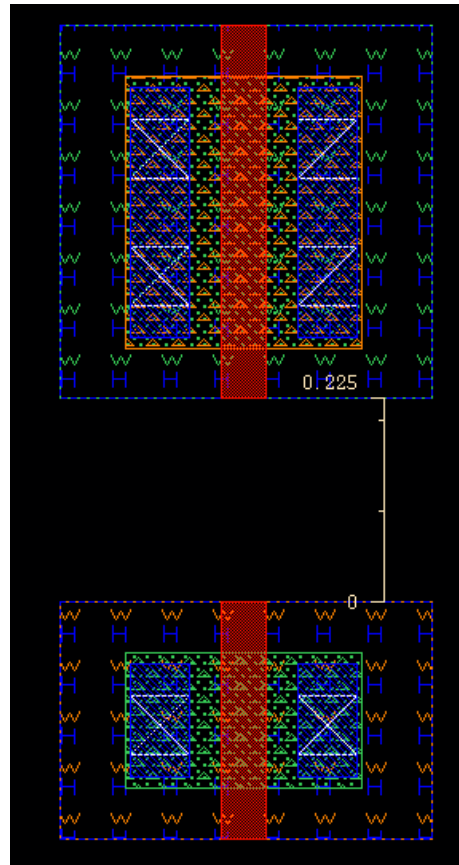


Figure 38 Layout of NMOS_VTH and PMOS_VTH transistors for an inverter

Combining these two transistors in a CMOS inverter requires making the input/output connections and adding Vdd and Gnd rails. The output connection can be easily made running a metal line from the drain of the PMOS to the drain of the NMOS. For the gate, we need to connect the two **poly** lines and make a **poly contact**. Contacts made on **poly** use the same layer as contacts on the **active** area. The PMOS and NMOS source will have to be connected to the Vdd and Gnd lines. The vertical spacing between the two power-supply metal lines is also known as cell pitch. The cell pitch defines an important feature of the standard cell library. When creating a standard cell library it is good practice to keep the cell pitch and the size of the power-supply lines constant. This makes it easier to connect the layout of different cells together. The last step consists in adding the body contacts. These contacts are made creating a **pimplant** in the **pwell** and a **nimplant** in the **nwell**. The **active/implant** layers are then connected to the power rails using a **contact** on **metal1**. The final layout is shown in Figure 39. For this specific example, I choose a cell pitch of 1 μ m (however, depending on the size of the gates in your standard library, you might have to choose a bigger spacing).

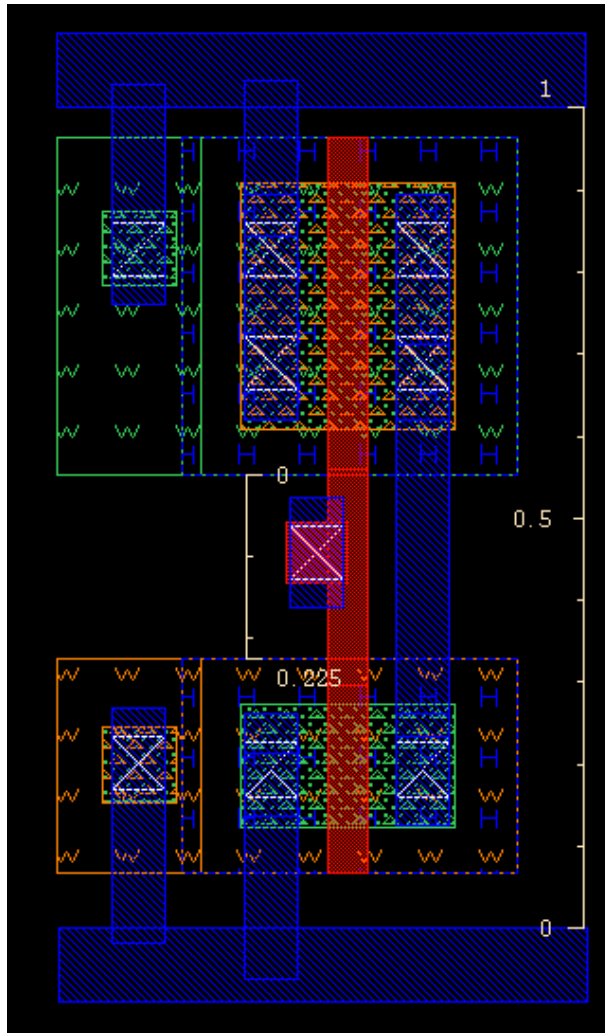



Figure 39 Inverter final layout

Layout design verification with Calibre

Calibre is a Mentor Graphics tool that works within the Cadence tool suite providing design verification. It provides Design Rules Check (DRC) and Layout vs Schematic comparison (LVS). All these functionalities can be recalled from Layout L through the Calibre menu.

Design Rules Check (DRC)

From Layout L, select **Calibre > Run nmDRC**. This will launch Calibre Interactive directly from Virtuoso. After loading automatically the ruleset file, you can start the check pressing **Run DRC** on the left of the Calibre Interactive window. The results are shown in a new window where a green check mark means no error, while a red x mark means that there is a violation of the design rule. If your design is error free, you can intentionally introduce an error to see how the DRC works. For instance, let's remove the **poly** from the **metal1-poly contact** as in Figure 40. After running the DRC again, the results should look as in Figure 41 (it is helpful to show only the rules unresolved rules by clicking on the filter button ).

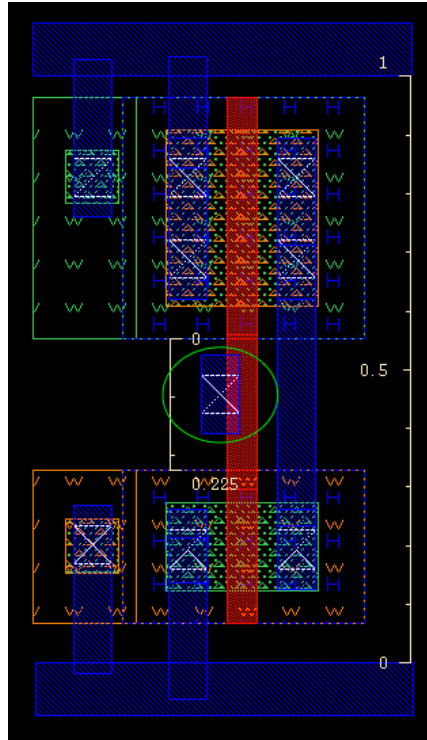


Figure 40 Inverter layout missing poly on metal-poly contact

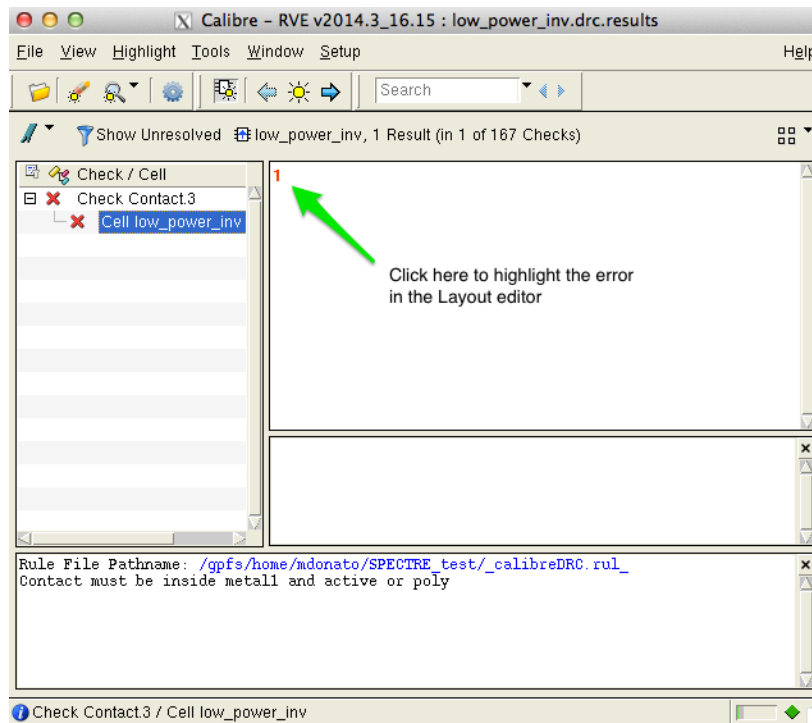


Figure 41 DRC results window

The right portion of the Calibre window in Figure 41 will show a list of the errors in the layout (in this case there is only one). By clicking on the error index (1) the Layout Editor will automatically zoom in on the portion of the layout where the error is.

Layout vs Schematic (LVS)

While the DRC allows us to check for errors in how the layout has been drawn, LVS checks whether the layout matches the schematic. This control guarantees that the fabricated circuit will functionally match the designed schematic.

In order to compare layout and schematic we need to add labels to the layout. Assuming we want to add labels to the metal lines, select **metal1** from the layer list. Then select **Create > Label** and make sure that the settings match those in Figure 42. Confirm the selection and place the labels by clicking on the corresponding **metal1** layer. The result should look as in Figure 43.

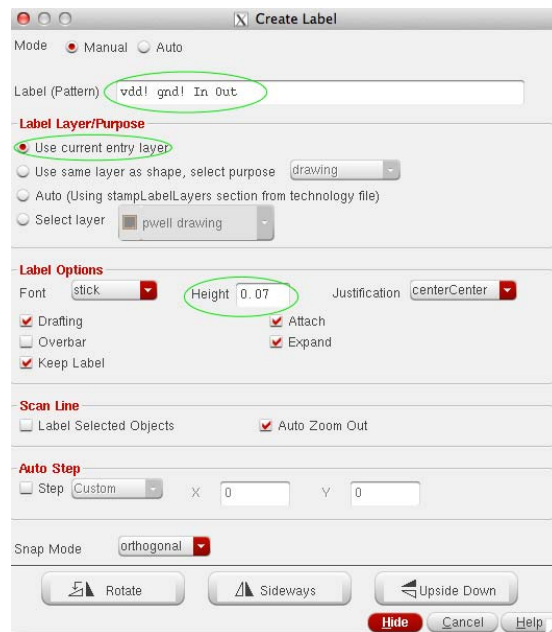


Figure 42 Creating labels for the inverter

At this point we can launch LVS by selecting **Calibre > Run nmLVS ...** Before running LVS go to **Input** and check **Export from schematic viewer** (Figure 44), then click on **Run LVS** on the left side of the window.

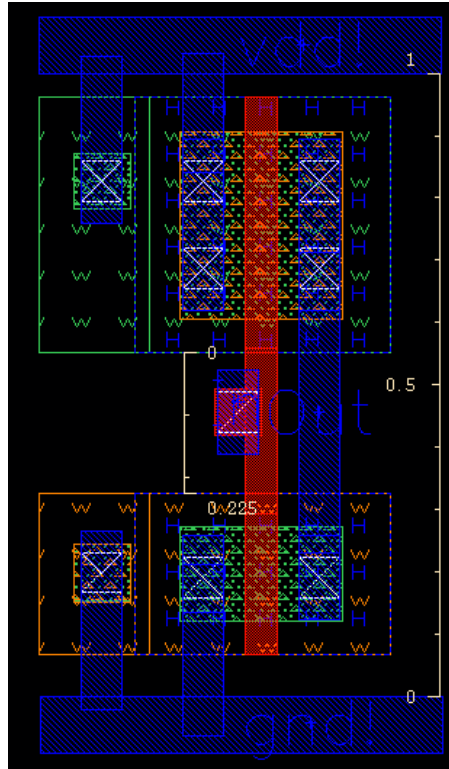


Figure 43 Adding labels to the inverter layout

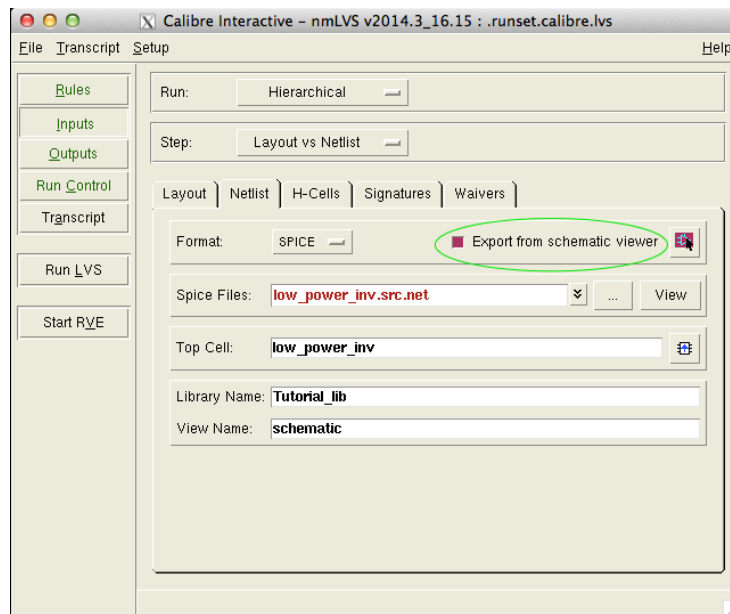


Figure 44 Setup LVS netlist export

If the layout matches the schematic you will see a happy smiley will pop up in the results window ☺ (Figure 45)

```

#####
##          CALIBRE SYSTEM          ##
##          LVS REPORT              ##
#####

REPORT FILE NAME:  low_power_inv.lvs.report
LAYOUT NAME:      /gpf/home/mdonato/SPECTRE_test/low_power_inv.sp ('low_power_inv')
SOURCE NAME:      /gpf/home/mdonato/SPECTRE_test/low_power_inv.src.net ('low_power_inv')
RULE FILE:        /gpf/home/mdonato/SPECTRE_test/_calibreLVS.rul_
RULE FILE TITLE:  LVS Rule File for FreePDK45
CREATION TIME:    Wed Sep 10 23:05:49 2014
CURRENT DIRECTORY: /gpf/home/mdonato/SPECTRE_test
USER NAME:        mdonato
CALIBRE VERSION:  v2014.3_16.15   Thu Aug 14 10:55:35 PDT 2014

OVERALL COMPARISON RESULTS

#####
#          CORRECT          #
#####

```

No errors!
Calibre is happy!



Figure 45 LVS summary with no errors found

How can we detect errors? Let's change the layout so that it will not match the schematic anymore. For instance, let's remove the metal connection between the body contact of the PMOS transistor and Vdd (Figure 46).

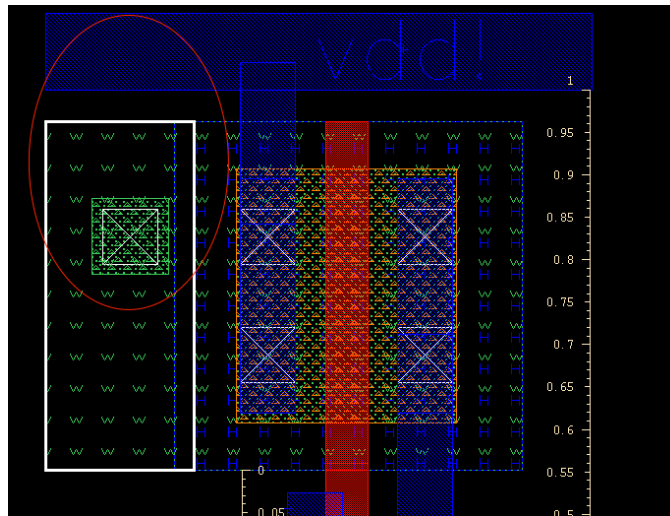


Figure 46 Remove body connection to Vdd

When we run nmlVS again the report will show more detailed results. In particular, if we look at the discrepancy report for incorrect instances, we can see that Calibre identifies the problem as a misplaced body connection (Figure 47)

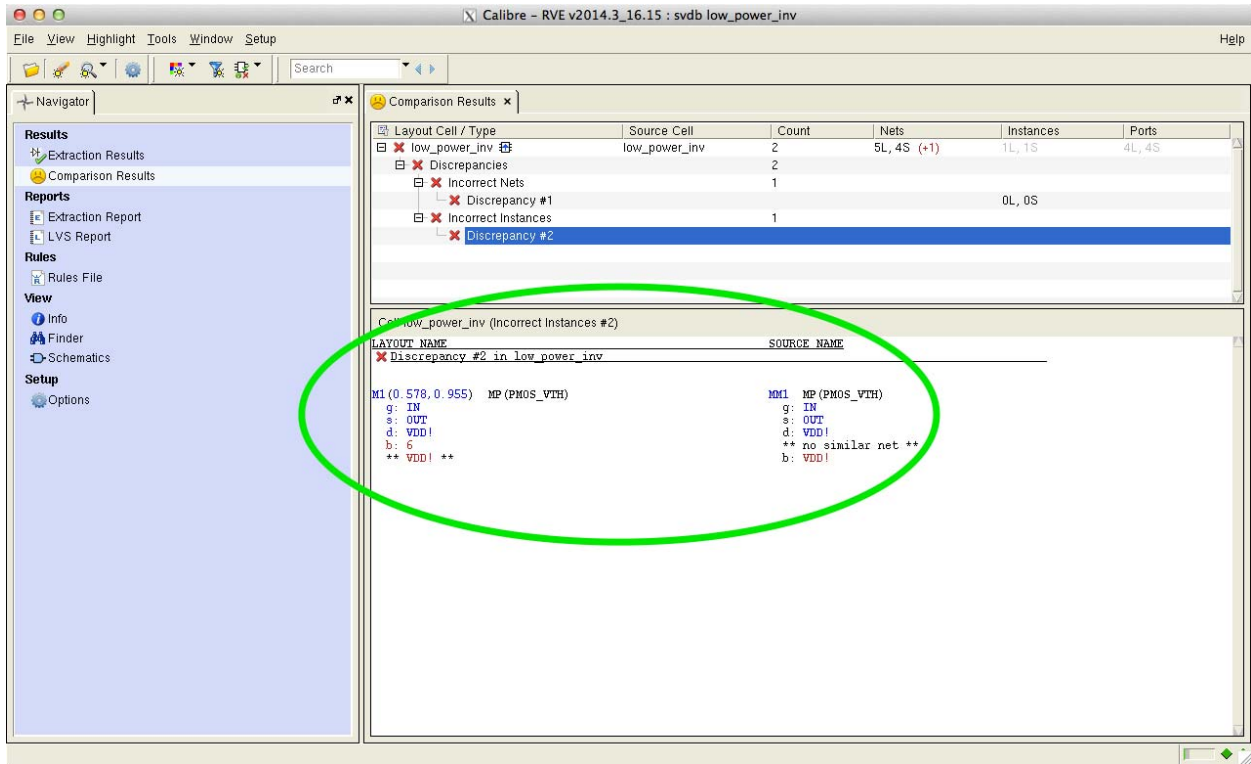


Figure 47 Report for missing body connection