ECEN 454 – Lab4: Design & Simulation of 1-bit adder

1. Introduction

In this lab you will be designing the 1-bit adder circuit, which will be then used to form the 4-bit adder in the next lab. In the following sections, you will be given instructions on how to perform **post-layout simulations** which you will need to do on each of the standard cells that have been designed and then use these standard cells to build the layout of the 1-bit adder. After this, you will verify its overall performance by performing post-layout simulations on the entire circuit.

2. General idea of Post-Layout Simulations

The electrical performance of a full-custom design can be best analyzed by performing a post-layout simulation on the extracted circuit net-list. At this point, the designer should have a complete mask layout of the intended circuit/system, and should have passed DRC&LVS steps with no violations. The detailed simulation performed using the extracted net-list will provide a clear assessment of the circuit speed, the influence of circuit parasitic and any glitches that may occur due to signal delay mismatches. Note that a satisfactory result in post-layout simulation is still no guarantee for a complete successful product.

The layouts that have been designed by you so far have included power lines (VDD, GND), input and output signals. But you have not checked whether the circuit will perform as you would want it to. Hence you will apply appropriate voltages and loads at the corresponding pins of the layout and then measure the circuit's performance. This would be a more realistic measure of its performance since it would include the circuit parasitics as well. This process is something similar to what you did in Lab1 where you designed a schematic and then simulated its performance (NOTE: In lab1 you only verified the logic performance of the circuit in terms of 1's and 0's and you did not measure its timing performance either) by applying a verilog test bench, but in this case you will be applying Pulse, DC and other supply voltages and measuring timing performance of the circuit using spectre.
To perform such simulations you will need to create a test-schematic which will contain the "symbol" of the CUT (Circuit-Under-Test) and you will then need to apply the appropriate inputs to measure its performance.

Before simulation, create a symbol of the basic gates, by choosing "Design>Create Cellview>From Cellview". Please edit the symbol to make it meaningful. An example of a symbol of Inverter is given below in Figure 1, and yours will look similar to this. Appendix at the end of this manual gives general instructions to draw custom symbol, and you can pick helpful parts from it to edit yours.

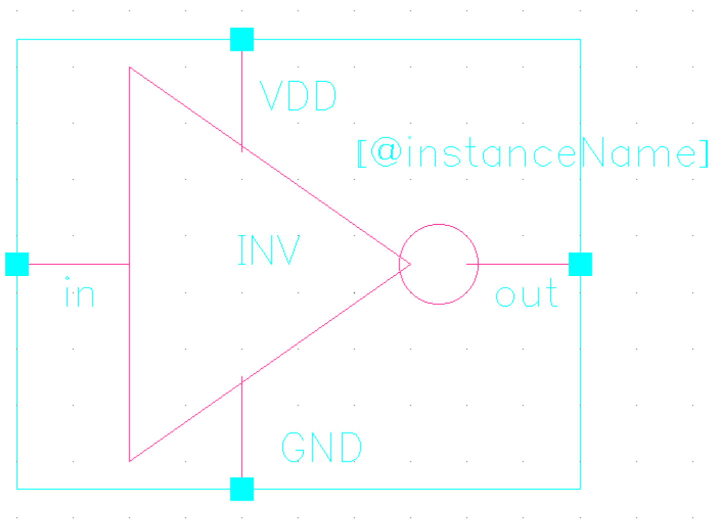


Figure 1. An example of a symbol of Inverter

3. Simulation

This section will explain how to perform the required simulations on a circuit. But, you must remember that for any circuit, you first need to make sure that it is DRC clean and that the layout and the schematic match by doing a LVS check. It is only after this that you must perform the post layout simulations. Here the procedure to perform post-layout simulations has been explained for an inverter circuit for which the DRC and LVS checks had been performed in Lab1.

- Open a new schematic in the same library where your Inverter circuit is by choosing "File >New > Cellview" in the Library Manager and selecting "Schematic" from the Tool selection widget. Let the Cellview name be something like "inverter_simulate".

- Next you select and place the components (You will wire them up in the next step). Choose your library and place the symbol of Inverter you created for the inverter design. (If you forget how to choose and place a component, refer back to Lab2). Change the library in the Component Browser to "**NCSU_Analog_Parts**" and place "**vdd**" and "**gnd**" by choosing "**Supply Nets**" from the component browser. Also place a DC voltage source "**vdc**" and a pulse waveform generator "**vpulse**" from the "**Voltage Sources**". Also place a capacitor in front of the output of Inverter by selecting "**cap**" from R_L_C section of the Component Browser. For now, please just place the components in your schematic, and let us set the parameters of those later in this section.

- Next you wire up these components. You use the dc voltage source to supply the VDD voltage to the circuit. For this you simply connect the "**vdd**" supply net with the **positive terminal** of the DC voltage source "**vdc**" and the "**gnd**" with the **negative terminal** of the voltage source. Also connect the VDD and GND of the Inverter to the positive and negative terminals of the DC supply voltage "**vdc**", or appropriate supply nets mentioned here.

- Now connect the **"+" terminal** of the pulse generator "**vpulse**" to the input pin of the Inverter symbol. Also connect the "-" terminal of "**vpulse**" to "**gnd**" net. The pulse generator generates a pulse of desired duration, pulse width, and voltage to provide the input stimuli to the CUT.

- Connect the capacitor to the output pin of the Inverter symbol. This completes the wiring of the circuit. Your schematic will be very similar to the design below:

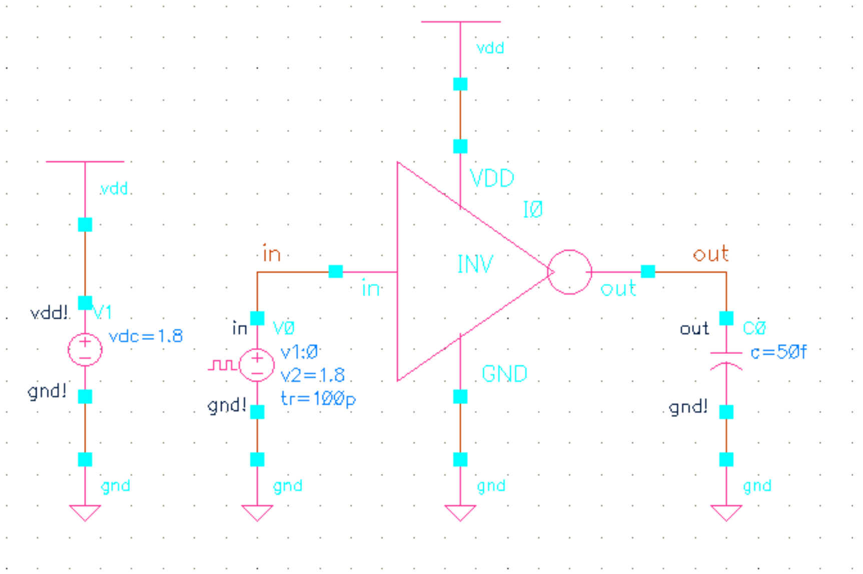
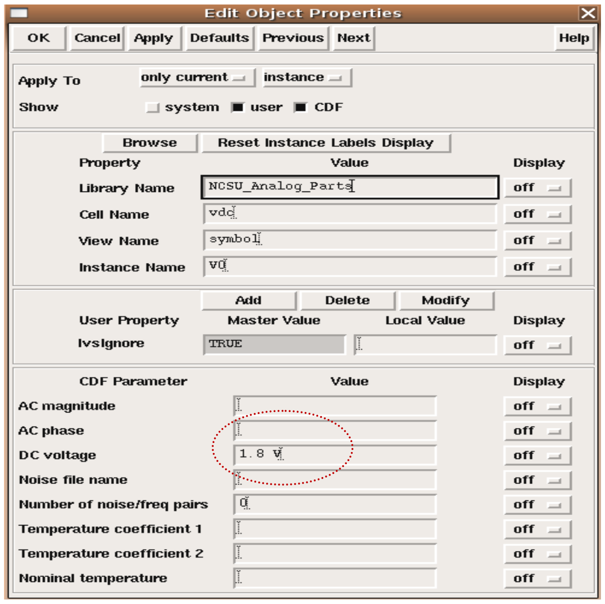


Figure 2. A sample schematic to simulate Inverter

- You may wonder that you are making a symbol from the schematic and trying to simulate it, then how can it be called a Post-Layout Simulation. But remember you had performed a LVS check on this circuit and if the layout and the schematic netlists match then the circuit would be the same. **All you will have to make sure is that the simulation is performed with the transistor and parasitics parameters extracted from the layout instead of the schematic**. This can be easily taken care by inserting an option as you will see later.

- You now edit the values of the parameters of the voltage sources and the load. For the DC voltage, you edit its voltage by first clicking it and then hitting "q" on the keyboard (or choosing "Edit > Properties >Object"). The Edit properties window pops up. Edit the "DC Voltage field" and fill in "1.8" as shown in the figure 3(a).
- Now you edit the properties of the pulse generator. You may try experimenting with the pulse width and the period of the clock as you like, but as an example you may fill up the properties as shown in figure 3(b). The delay time indicates the time after 0 after which the waveform is generated. Voltage 1 and Voltage 2 may correspond to the respective voltages



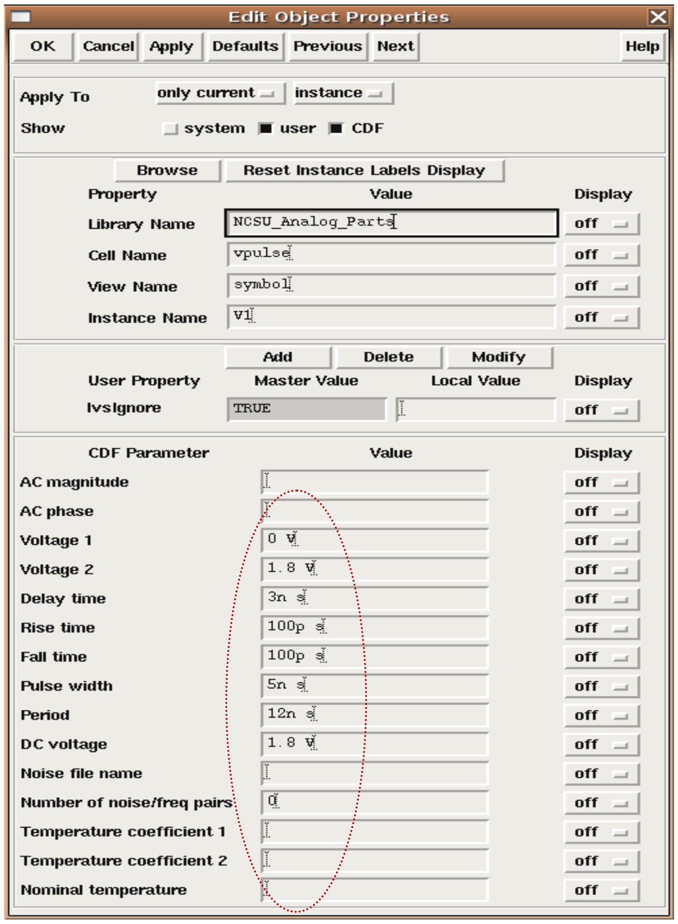
Apply To: only current, instance
Show: system, user ■, CDF ■

Property	Value	Display
Library Name	NCSU_Analog_Parts	off
Cell Name	vdc	off
View Name	symbol	off
Instance Name	V0	off

User Property	Master Value	Local Value	Display
lvsIgnore	TRUE		off

CDF Parameter	Value	Display
AC magnitude		off
AC phase		off
DC voltage	1.8 V	off
Noise file name		off
Number of noise/freq pairs	0	off
Temperature coefficient 1		off
Temperature coefficient 2		off
Nominal temperature		off

(a) DC voltage (vdc)



Apply To: only current, instance
Show: system, user ■, CDF ■

Property	Value	Display
Library Name	NCSU_Analog_Parts	off
Cell Name	vpulse	off
View Name	symbol	off
Instance Name	V1	off

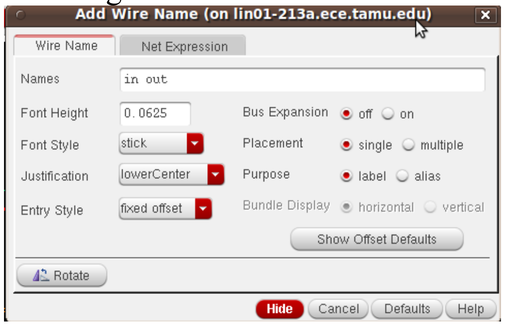
User Property	Master Value	Local Value	Display
lvsIgnore	TRUE		off

CDF Parameter	Value	Display
AC magnitude		off
AC phase		off
Voltage 1	0 V	off
Voltage 2	1.8 V	off
Delay time	3n s	off
Rise time	100p s	off
Fall time	100p s	off
Pulse width	5n s	off
Period	12n s	off
DC voltage	1.8 V	off
Noise file name		off
Number of noise/freq pairs	0	off
Temperature coefficient 1		off
Temperature coefficient 2		off
Nominal temperature		off

(b) pulse voltage source(vpulse)

Figure 3. Setting the parameters of voltage sources

- Edit the properties of the output load (capacitor). A default value of 1pF will be filled in. But this can be lowered down to a lower value of around 50fF.
- Next you add label names to the nets you want to observe after simulation. Choose "Add >Wire Name" and then type in a desired wire name in the pop-up window and then click on the corresponding net you want to name. For the case of the inverter you will observe the input and the output signals and name them as "in" and "out" as shown in the schematic design above in Figure 2.



Names: in out
Font Height: 0.0625
Font Style: stick
Justification: lowerCenter
Entry Style: fixed offset
Bus Expansion: [✓ off] on
Placement: [✓ single] multiple
Purpose: [✓ label] alias
Bundle Display: [✓ horizontal] vertical

Figure 4. Naming a wire

- Now you will start the simulation. Cadence will internally generate a netlist of the circuit being simulated and use Spectre for all the timing simulations and then generate a waveform for the signals that need to be probed. Since Spectre will be looking for model parameters for nmos and pmos transistors you have to include the model cards in a certain location and ask Spectre to look for the model cards by indicating the path to the simulation tool.
- Create a directory named "models" under your "cadence" directory. And create a directory named "spectre" in the "models" directory.

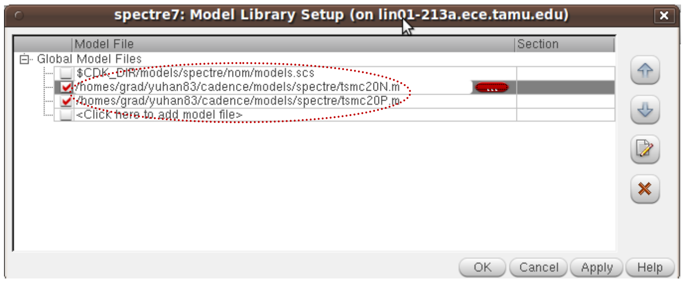
```
cd    ~/cadence
mkdir      models
cd    models
mkdir    spectre
```

- Download the model cards (tsmc20N.m, tsmc20P.m) that have been put up on the website and place them in "~/cadence/models/spectre/" after creating the directory. Make sure these are the only 2 files in that particular directory so as to avoid confusion and also retain their names.

- To be able to run simulations, please source the following first (make sure cadence is closed):
  source /opt/coe/cadence/SPECTRE181/setup.SPECTRE181.linux.bash
  Now run Cadence as per procedure.

  Now select launch> **ADE XL**, then go to:
  Setup > Simulator/Directory/Host". Change the Simulator from Hspice to Spectre. Click OK.

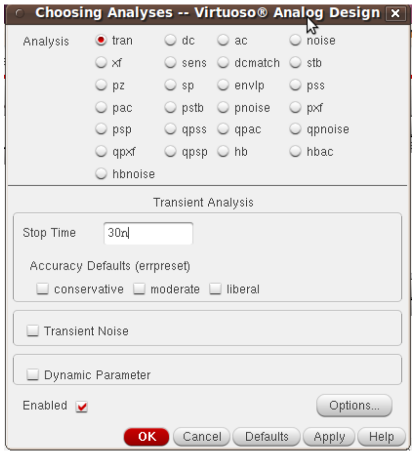
- Now you have to make sure that the simulator has the right path to pick up the model card definitions. Choose "Setup > Model Library" and the following window will pop up. Make sure you have to add tsmc20N.m and tsmc20P.m into model library and delete the old ones. Then click on "OK".



Model File	Section
Global Model Files	
☐ $CDK_DIR/models/spectre/nom/models.scs	
✓ /homes/grad/yuhan83/cadence/models/spectre/tsmc20N.m	
✓ /homes/grad/yuhan83/cadence/models/spectre/tsmc20P.m	
☐ <Click here to add model file>	

Figure 5. Setting model Path Window

- Now you will choose "Analyses > Choose". By default the "tran" (tran stands for transient) option has been selected. Since you choose to do transient analysis on our circuit you will continue with the option. But you still have to fill in the Stop Time which indicates the time until which the simulation will be performed. Let this around 30ns. Click OK.


Choosing Analyses -- Virtuoso® Analog Design

Analysis: [✓ tran] xf pz pac psp qpxf hbnoise dc sens sp pstb qpss qpsp ac dcmatch envlp pnoise qpac hb noise stb pss pxf qpnoise hbac

Transient Analysis

Stop Time: 30n

Accuracy Defaults (errpreset): ☐ conservative ☐ moderate ☐ liberal

☐ Transient Noise

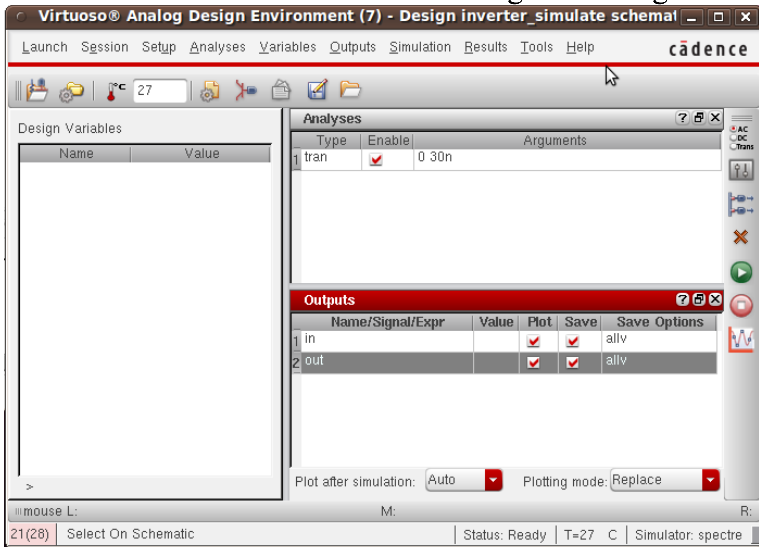
☐ Dynamic Parameter

Enabled ✓

Options... OK Cancel Defaults Apply Help

Figure 6. Choosing Analyses window

- Next you select the output signals that need to be plotted. Choose “Outputs > To be Plotted > Select on Schematic”. Then click on the nets in the schematic that you had named “in” and “out” to be plotted and then click on ESC. Verify if the “Outputs” section of the Analog environment main window is populated with the correct signal names.
- The main window will look similar to following at this stage.


Virtuoso® Analog Design Environment (7) - Design inverter_simulate schemat

Launch Session Setup Analyses Variables Outputs Simulation Results Tools Help — cādence

Temperature: 27

Design Variables

Name	Value

Analyses

	Type	Enable	Arguments
1	tran	✓	0 30n

Outputs

	Name/Signal/Expr	Value	Plot	Save	Save Options
1	in		✓	✓	allv
2	out		✓	✓	allv

Plot after simulation: Auto   Plotting mode: Replace

mouse L: M: R:

21(28) Select On Schematic | Status: Ready | T=27 C | Simulator: spectre

Figure 7. Analog Environment Window with proper simulation setup

- Now start the simulation by selecting “Simulation > Run”. The waveforms for the selected nodes should appear in a new window as follows. You may change the color, type or style by right click related signal you are observing.

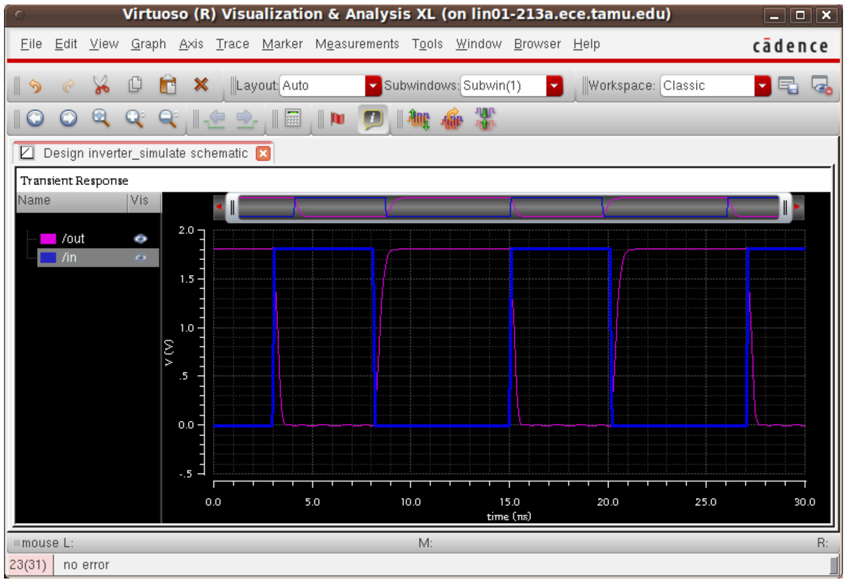

Virtuoso (R) Visualization & Analysis XL (on lin01-213a.ece.tamu.edu)
File Edit View Graph Axis Trace Marker Measurements Tools Window Browser Help
cādence
Layout: Auto
Subwindows: Subwin(1)
Workspace: Classic
Design inverter_simulate schematic
Transient Response
Name
Vis
/out
/in
V (V)
time (ns)
mouse L:
M:
R:
23(31)
no error

Figure 8. Waveform Window with results from transient simulation

- You will see the waveforms plotted together on the same axis. To separate them you can choose the option "Graph > Split Current strip". This will display the signals on different axes.

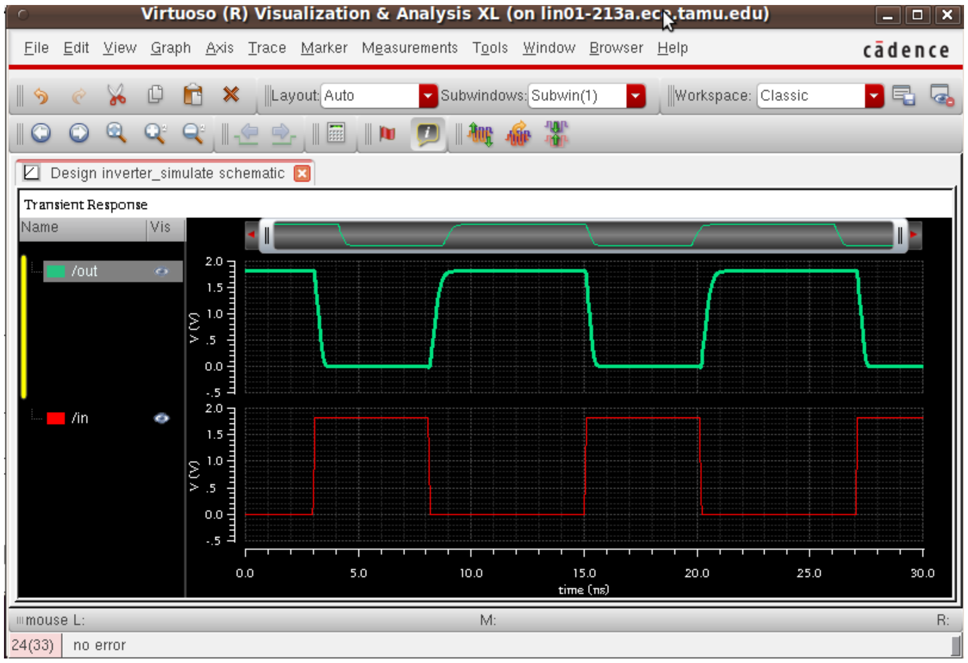
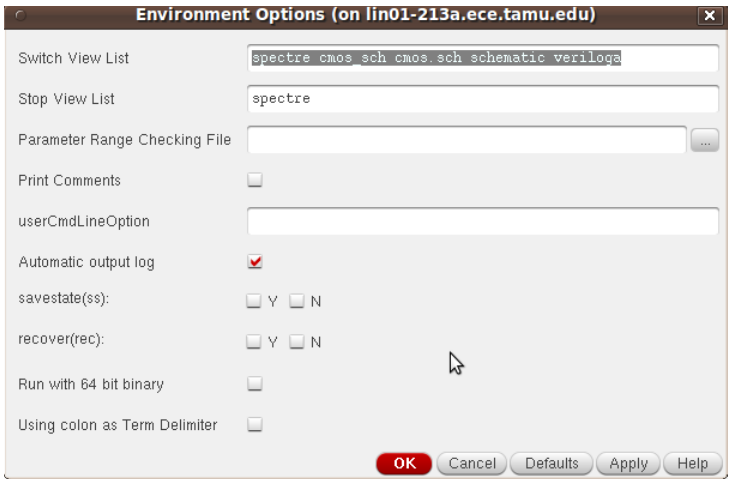

Virtuoso (R) Visualization & Analysis XL (on lin01-213a.ece.tamu.edu)
File Edit View Graph Axis Trace Marker Measurements Tools Window Browser Help
cādence
Layout: Auto
Subwindows: Subwin(1)
Workspace: Classic
Design inverter_simulate schematic
Transient Response
Name
Vis
/out
/in
V (V)
time (ns)
mouse L:
M:
R:
24(33)
no error

Figure 9. Waveform Window with results from transient simulation(split form)

- **So far is the procedure to generate the waveforms on the schematic design that you created for the inverter, not the layout with extracted parasitic**. Although the schematic and the layout designs are equivalent, the transistors in these designs have different sizes. The LVS check that was performed only checks if the devices, nets and terminals in both the schematic and the layout (By layout, you mean the "extracted" view of the layout) are equivalent. It however has not been programmed to check if the device sizes are equivalent. The device sizes will actually determine the timing of the circuit. Also the extracted view of the layout may include a lot of parasitics which can also affect the timing.

- Now, let us make the simulation to takes into account the extracted parameters of the layout. You need to make a small modification in the "Analog Environment" window. Choose "Setup > Environment". You will see the following window.



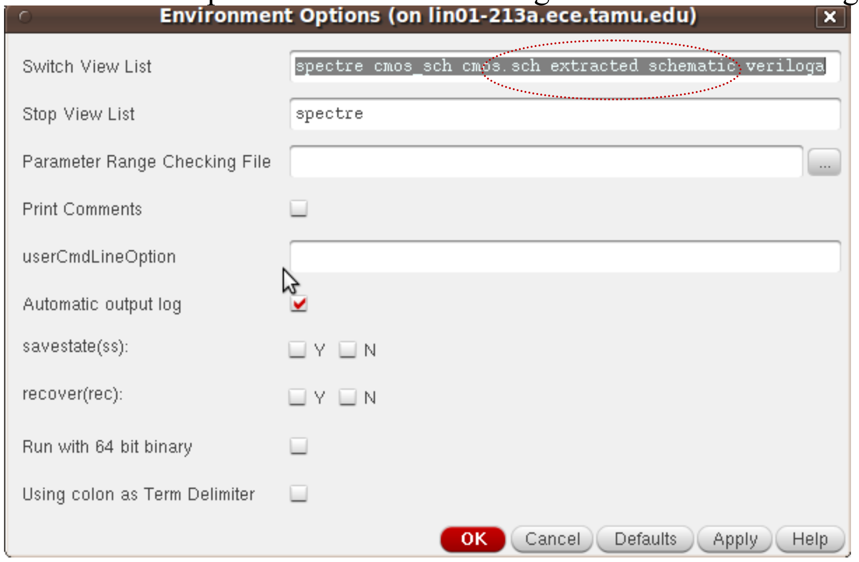
Environment Options (on lin01-213a.ece.tamu.edu)

Switch View List	spectre cmos_sch cmos.sch schematic veriloga
Stop View List	spectre
Parameter Range Checking File	
Print Comments	☐
userCmdLineOption	
Automatic output log	✓
savestate(ss):	☐ Y ☐ N
recover(rec):	☐ Y ☐ N
Run with 64 bit binary	☐
Using colon as Term Delimiter	☐

Figure 10. Environment Option Window

- Now you will make a change in the line "Switch View List" by including "extracted" exactly in front of the word "schematic". This makes sure that the simulator picks up the extracted view parameters (if one exists for the same cell).

The Environment Options will look something like this after the change.



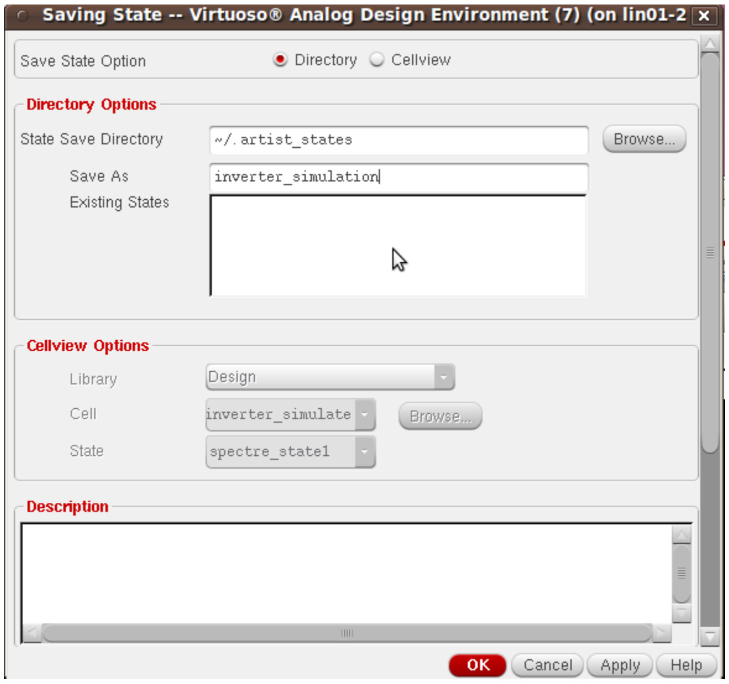
Environment Options (on lin01-213a.ece.tamu.edu)

Switch View List	spectre cmos_sch cmos.sch extracted schematic veriloga
Stop View List	spectre
Parameter Range Checking File	
Print Comments	☐
userCmdLineOption	
Automatic output log	✓
savestate(ss):	☐ Y ☐ N
recover(rec):	☐ Y ☐ N
Run with 64 bit binary	☐
Using colon as Term Delimiter	☐

Figure 11. Environment Option Window With Modification

- Click "OK" and then rerun the simulations. The waveforms will now be according to the parameters that have been generated from the extracted layout.

- You may save the simulation setups to reuse it later so as not to resetting the environment again, by choosing “Session > Save State” and putting a setup name on “Save As” field.



Save State Option [✓ Directory] Cellview

State Save Directory ~/.artist_states

Save As inverter_simulation

Existing States

Library Design

Cell inverter_simulate

State spectre_state1

Figure 12. Save simulation setup

4. Power, Rising and Falling Delays and Cell Height.

- **Power Measurement:** To measure the power delivered by the voltage source during the simulation (post-layout), choose Browser->Results->Open Results, double click “psf” file, browser window pops up in the left part of waveform window. Now Click on V0 under “tran to tran” directory(or whatever is the node name of the VDD voltage source), then click “/VO/PLUS” on the left side of waveform window and click tools->calculator to activate “**calculator**” for this signal. Then the “Calculator” pops up as shown below. Now choose **average** from the special functions tab in the calculator. Next select Tools->Table in calculator window. Then the average current delivered from the power source is calculated.

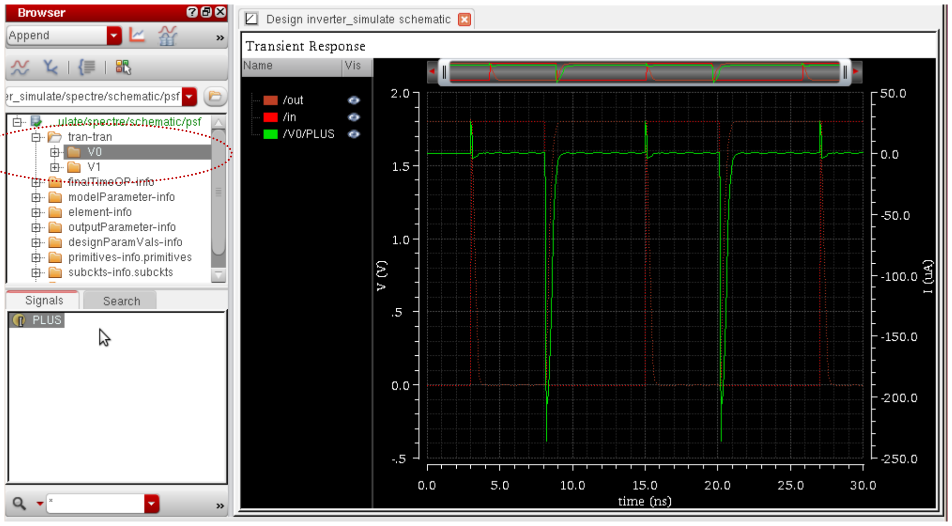
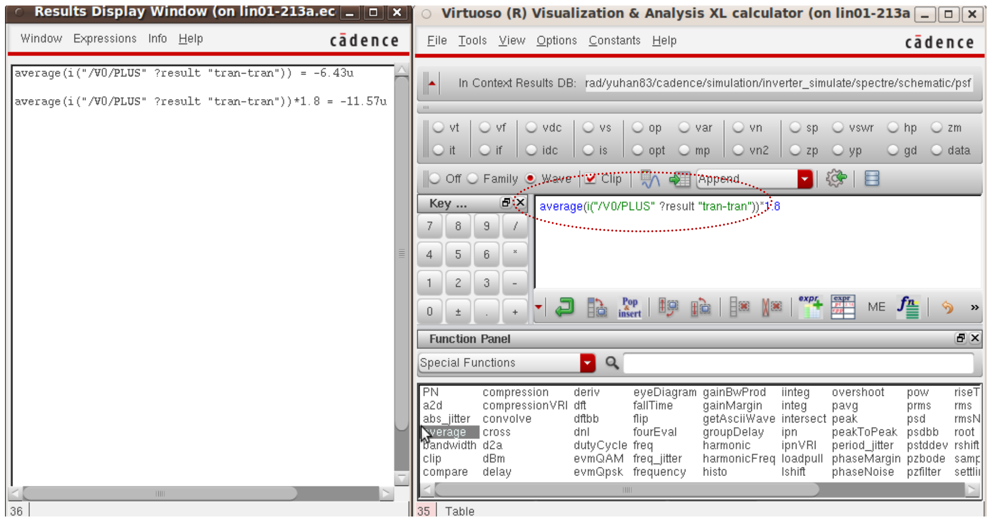


Figure 13. Simulation Results Browser


Results Display Window (on lin01-213a.ec
Window Expressions Info Help
Virtuoso (R) Visualization & Analysis XL calculator (on lin01-213a
File Tools View Options Constants Help
Function Panel
Special Functions

```
average(i("/V0/PLUS" ?result "tran-tran")) = -6.43u

average(i("/V0/PLUS" ?result "tran-tran"))*1.8 = -11.57u
```

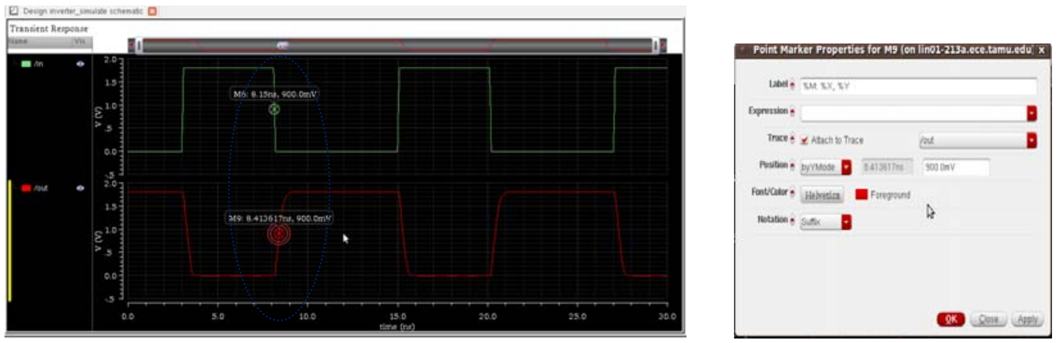
In Context Results DB: rad/yuhan83/cadence/simulation/inverter_simulate/spectre/schematic/psf

```
average(i("/V0/PLUS" ?result "tran-tran"))*1.8
```

Figure 14. Calculator for results display

Multiplying this with the voltage supplied from the source gives you the power delivered. You could go on to type "*1.8" in the calculator window, then click Tools->table, this will give you the average power consumed from the voltage source. Perform the same on the other voltage sources. If the voltage source is a pulse find the average voltage (choose "in"->"average") during the period of transient analysis and then multiply with the average current calculated.

- From the waveforms generated, you can measure the rising and falling delays of the output signal. There are two ways to get delays, you could choose either way to implement it.
  Method1: use marker to get related 50% of output delay time and 50% of input delay time, then subtract each other.
  Click Marker->Click Marker, then you would see a circle with crossing line in it. Drag it to the position you want to measure.



Point Marker Properties for M9 (on lin01-213a.ece.tamu.edu)

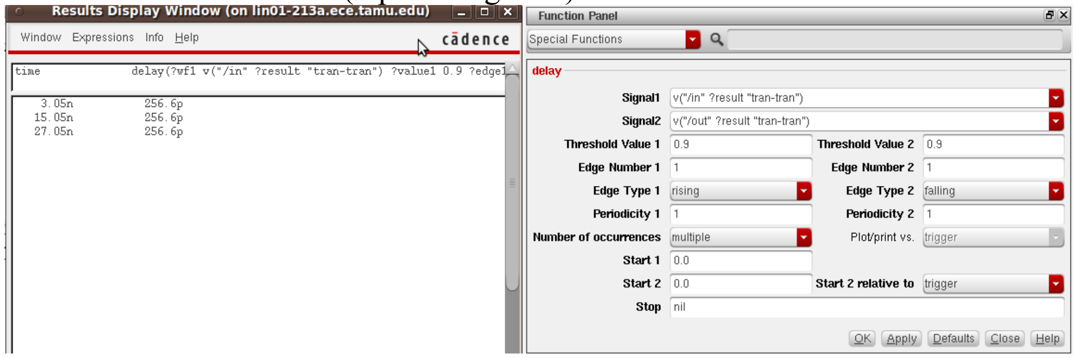
Label: %M: %X, %Y
Expression:
Trace: ✓ Attach to Trace /out
Position: byYMode 8.413617ns 900.0mV
Font/Color: Helvetica 🟥 Foreground
Notation: Suffix

Figure 15. Use marker to measure delay

In such a case, the rising delay would be (8.4136-8.15)ns. During this process, in order to make the marker to stay exactly on 0.9 v (half of 1.8v), you could change marker property as what I showed above.

Method2: Use delay function of calculator to measure it directly

You choose either "in" or "out" signal to activate "calculator", then choose "delay" on "function panel" on it, then make changes on "delay" option table as below. Make sure signal1 and signal2 refers to "in" and "out", their threshold values should be 0.9. In this case we test falling delay, output signal should be "falling" edge type while input signal should be "rising" edge type, Number of occurrences should be "multiple". Then the results display window would show falling delay 256.6ps as well as the occurrence time "3.05ns 15.05ns 27.05ns".(input rising time)



```
time                delay(?wf1 v("/in" ?result "tran-tran") ?value1 0.9 ?edge1
    3.05n           256.6p
   15.05n           256.6p
   27.05n           256.6p
```

delay

Signal1	v("/in" ?result "tran-tran")		
Signal2	v("/out" ?result "tran-tran")		
Threshold Value 1	0.9	Threshold Value 2	0.9
Edge Number 1	1	Edge Number 2	1
Edge Type 1	rising	Edge Type 2	falling
Periodicity 1	1	Periodicity 2	1
Number of occurrences	multiple	Plot/print vs.	trigger
Start 1	0.0		
Start 2	0.0	Start 2 relative to	trigger
Stop	nil		

Figure 16. Use function table to measure delay

- Ideally we would want to match the rising and falling delays of the standard cells to achieve the best trade-off between area and frequency and to make sure that the worst case and best case delays of the circuit do not differ a lot. But this might require modifying your earlier layouts.

- Now, repeat this exercise for the other standard cells that you have designed i.e. XOR2 and the NAND2 gate.

- You need to make sure all the standard cell layouts have the same height. This is done because you will eventually be using all these gates to build the adder circuit and you would need to place these gates adjacently. When they are of the same height the power lines VDD and GND can be connected to each other and also the Nwell regions of all the cells can be merged.

- Hence it has to be made sure that the Nwell regions of the cells have the same size and also the power lines have to be at the same height so that it is easy to connect to cells with common power lines.

5. **Single bit Adder.**

- Next you will use the standard cells (XOR, Nand and Inverter) to build the layout of a single bit full adder circuit.

- You have a free-hand to place the cells and route the signals accordingly. Only make sure that you use a row-based (use multiple cells in each row and the design may contain 2-3 rows) design to place the standard cells adjacently. You may connect the power lines of adjacent cells. (Since all are of the same height).

- The single bit adder should not take more than a couple of rows. You have to be careful during routing. Make sure you connect the appropriate pins without violating design rules or creating shorts between crisscrossing layers of the same type. You are free to use metal layers 1-4. (Do not use higher metal layers).

- Also you will need to be careful with pin names. No 2 nets can have the same pin name. DRC will flag an error. So you will have to give different names to pins that need to be created for the design.

- After you complete the routing, run a DRC check on the circuit. You will have to debug any errors in the layout. After the layout is DRC clean, perform extraction on the circuit with the “extract_parasitic_capacitances” switch selected.

- Next you will need to perform the LVS check, for which you will first need to create a schematic of the same design. This is similar to the schematic of the full-adder that you created in Lab1, the difference being that this time you will use gate symbols from your library instead of the inbuilt Cadence library parameters.

- Wire the schematic and make sure it has no warnings. Now open the extracted view of the layout that was generated and perform a LVS check on the circuit. Debug the errors if the LVS netlists don't match.

- Now perform post-layout simulations on this circuit (This will be helpful to you since this block will be repeated 4 times to create the complete 4-bit adder in the next lab). Refer to the previous sections to perform the post-layout simulations.

- How to get maxim frequency: There are three inputs for 1bit adder. Make sure that two of them have long enough pulse and cycle. We just focus on the other input. Then decrease the pulse as well as cycle for this input little by little. Then the output including carry and sum would become more and more abnormal little by little, until the rising can not completely arrive to VDD or the falling can not completely fall to zero, we measure this threshold as minimum period, you could get maximum frequency through $1/T_{min}$. For example, in the waveform below, we set A 40ns to be its period, while we set B 20ns as its cycle. Then you set C 10ns first. We may find its output is normal, then decrease it step by step, maybe when T=3ns it is still could arrive VDD or Zero, but when you go on to decrease it to T=2.9ns it is abnormal, then your threshold is T=3ns, your maximum frequency is 1/3ns.

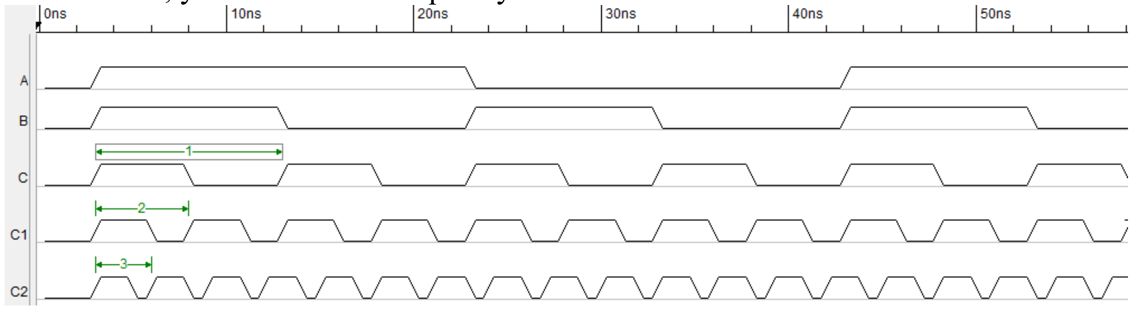


Figure 17. Maximum Frequency measure

Report Requirements:

1. The report should contain the schematic, layout, DRC and LVS reports of the 1-bit adder.

2. Report rising delay as well as falling delay for all the gates you designed including inverter, nand2, xor2 and 1bit adder. Make sure the error with respect to each other should be less than 10%.
3. Please also report the schematic, layout, and LVS result of each gate only if you update anyone of those or you have failed LVS of any gates so far in previous labs (if you resize the nmos or pmos, you should also included new version).

4. Turn in the waveforms of the outputs of all gates and 1-bit adder with all possible combinations of all inputs. Please make all your input signals (the pulses) represent all the possible logic values. For example, pulses into two inputs to NAND2 will show 4 logical values: 00, 01, 10, and 11. This can be done easily with putting multiples of pulse voltage sources into your simulation schematic.

5. Report power dissipations from all voltage sources of all the gates and adder. When acquiring power dissipations from pulses, you need to multiply the average current with the actual average value of the pulses, not just 0.9V.

6. Report the maximum frequency which your 1-bit adder can achieve. Provide waveforms to support. (**note that your outputs should completely rise to VDD and fall to GND**)

   For this, you may want to resize your transistors to reduce delay. This means you also need to modify the layout of the standard cells. In case you choose to modify your layout, make sure you perform i)DRC check ii)LVS check before going ahead with the post-layout simulations to measure the new delays. This is important since while changing the layout you may have accidentally changed some wiring which you did not intend to do.

7. Please try to reduce delay. Additional scores will be given according to your effort.

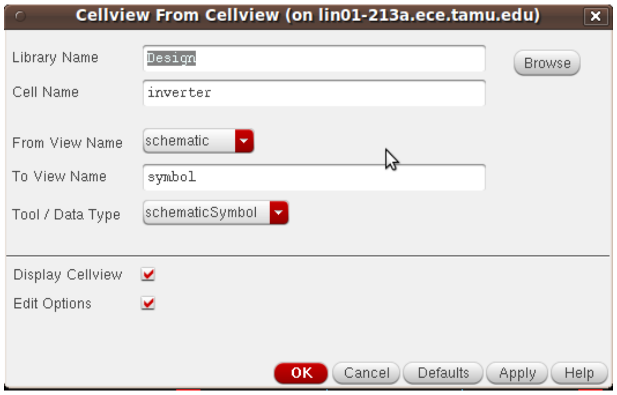
8. APPENDIX : Creating Custom Symbols

- This section is providing instructions to draw custom symbols for gates created as schematic. You may first extract a symbol from your schematic, and use this section to update and edit the symbol because the symbol drawn automatically from your schematic will have only rectangle and pins deployed randomly.

- As you saw in Lab1 you can create a symbol of any circuit that you design by using the inbuilt option in the Design Window to create a symbol for the current cellview. But you can also create custom symbols for the logic gates rather than the rectangular ones that are created by default. For example you would need to create a triangular symbol followed by a bubble for an inverter.

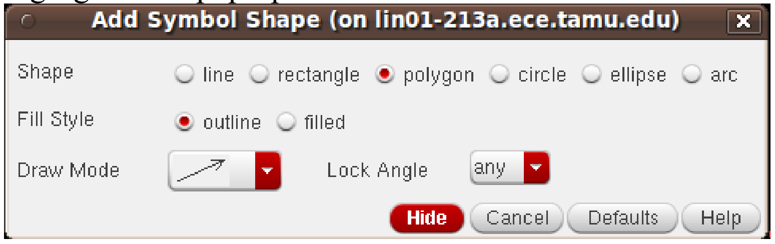
- To create a custom symbol for any design you can create a new cellview "symbol" in the same cell. The following instructions explain how you can create a symbol for an inverter.

- In the schematic window, choose Create > Cellview > From Cellview. Type in the cell name of the inverter you have designed as shown below. Click "OK" and a new window will pop-up.


Cellview From Cellview (on lin01-213a.ece.tamu.edu)

Library Name: Design
Cell Name: inverter
From View Name: schematic
To View Name: symbol
Tool / Data Type: schematicSymbol
Display Cellview: ✓
Edit Options: ✓

- Since you want to draw a triangle, now choose Create> Shape > Polygon and the following figure will pop-up.


Add Symbol Shape (on lin01-213a.ece.tamu.edu)

Shape: line rectangle [✓ polygon] circle ellipse arc
Fill Style: [✓ outline] filled
Draw Mode:
Lock Angle: any

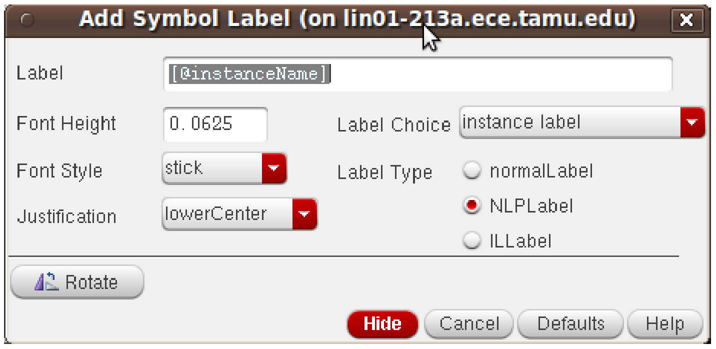
- Choose Hide and begin at one of the desired vertices of the triangle that you want to draw. Choose all the vertices that you want in the polygon and then click back on the first vertex to complete the figure.

- You can then draw a bubble in front of the triangle by choosing to draw a small circle in from of it. You can click on a point just in front of the triangle and then extend the mouse to indicate the radius of the circle that you want to create.

- You should continuously perform "Check and Save" on the symbol by pressing "F8" or choosing "Design > Check > Save". The symbol will automatically be compared to the schematic in the same cell and at this point will give out some warnings. This is because the tool sees that you have not yet included the names of the pins defined in the schematic in the symbol.

- You can add a pin by choosing "Add > Pin" and choose the appropriate direction (input, output or input/output) for it. Add pins for input A, output A', power lines (input/output) VDD and GND. You will see that the pin in this case is slightly different from the one that appears in a schematic. It is a small box with a line attached to it and the pin name on it towards the end. The box is the actual pin and so it must be pointing in the away from the rest of the symbol. Note after you add in all the appropriate pins with proper directions the warnings go away.

- The next step is to create a label for the symbol. You do this by choosing "Add > Label" after which the following dialog box will appear.


Add Symbol Label (on lin01-213a.ece.tamu.edu)

Label: [@instanceName]

Font Height: 0.0625

Label Choice: instance label

Font Style: stick

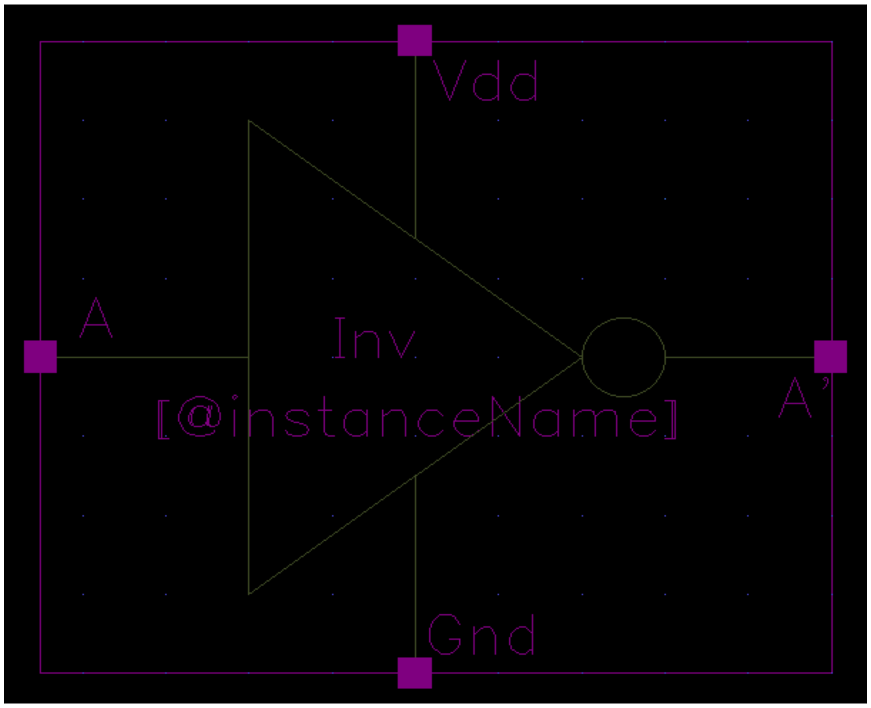
Label Type: normalLabel [✓ NLPLabel] ILLabel

Justification: lowerCenter

The first label you add to the symbol is the [@InstanceName] which will be replaced by instance names such I0, I1 etc when the symbol will be used in other designs. This is to differentiate between different instances of the same cell.

The second label you add to the symbol will be the cell name. You do this by choosing "Add > Label". Then choose Label Type as "NormalLabel" and type in the cell name like "inv" in the Label field.

- The final step is to create a selection box for the symbol which defines the area in which the symbol will be selected if you were to click in it. This is done by choosing "Add >Selection Box". Click on "Automatic" and a selection box will surround the symbol with pins on its boundary. The inverter symbol will look something like this.



- Try to edit the symbol shape such that all its pin-names are within the selection box to avoid confusion later on when using the symbol for a higher level of design.