EE431 Lab 3 – CAD of VLSI Devices Lab

Week 3: Timing analysis and PCELLs

(Details of what is due at end of handout)

Please do not start this lab until you've finished the week 2 lab!!!!

IMPORTANT POINTS:

- Simulate the extracted view you created last week
- Simplify layout by using PCELLs

INTRODUCTION: (This is the third lab in a five lab series on custom design entry)

Last week you drew a layout entirely by hand. There were a ton of steps and I'm guessing it took at least close to the entire lab to finish if not more. That was for a single inverter – two transistors. If your boss comes to you and asks you to build a custom 64-bit multiplier (that has 10s of thousands of transistors) needless to say, you'll probably have something to do until it's time for you to retire. To make it easier to build custom circuits in a reasonable amount of time, a thing called PCELLs (parameterized cells) were developed. PCELLs are pre-drawn transistors but you can set various parameters on them. Width, length, fingers as well as some other parameters can all be set through by just typing them into a window. If you have 10,000 transistors to draw, that will save you some time but it's still not a solution that will make you totally happy. Tricks to speed things up include finding a block that repeats in your circuit, draw one of those blocks really well, make a symbol for it and then instantiate as many as needed. Or you can place your transistors and let the software draw in all the interconnect (though every time you trust the software to do some of your design you are possibly losing some of the advantage of your custom work). If you don't care about performance (power, speed, size, testability, etc), then you can write a line of Verilog code, press a button and you'll have a layout in no time at all. That gets your chip to market quickly but it probably will run slower than another company that spent time customizing it. Next week we'll look at letting the software do all the work but this week we'll introduce how to use PCELLs.

The other thing that we'll do is to see how to simulate the extracted view that you generated last week. Remember that the extraction process evaluates the parasitic that the wires and transistors see and adds them to a schematic-ish/layout-ish type view. Once you have these parasitic you can more accurately simulate the true behavior of your circuit. You could calculate these capacitances by going to the tables near the beginning of last week's tutorial and looking at the shape overlaps and then manually add those capacitances to your schematic but, again, you'd like to finish your chip within your lifetime. For an inverter it isn't so bad but let's say you wanted to know how fast the 10,000 transistor multiplier could run…. 'nuf said.

We'll start with simulation of last week's extracted view of the inverter and then go to the PCELL introduction.

LAB:

1. **PREPARATION:**

To do this tutorial, you need the extracted view of the inverter you drew last week (week 2). There should be no extraction errors.

2. **TIMING ANALYSIS:**

This section takes you through simulation of your extracted view.

a. Open the extracted view.

b. Due to a bug that hasn't been resolved at the time that I was writing this tutorial, go to: Verify → LVS… and then click on "Build Analog". In your Library Manager window, a

new view should appear called “analog_extracted”. Close the “extracted” view and open the “analog_extracted” view.

c. From the “analog_extracted” layout window, open the simulator window: Launch → ADE L (Analog Design Environment – ADE L).

d. In the ADE:

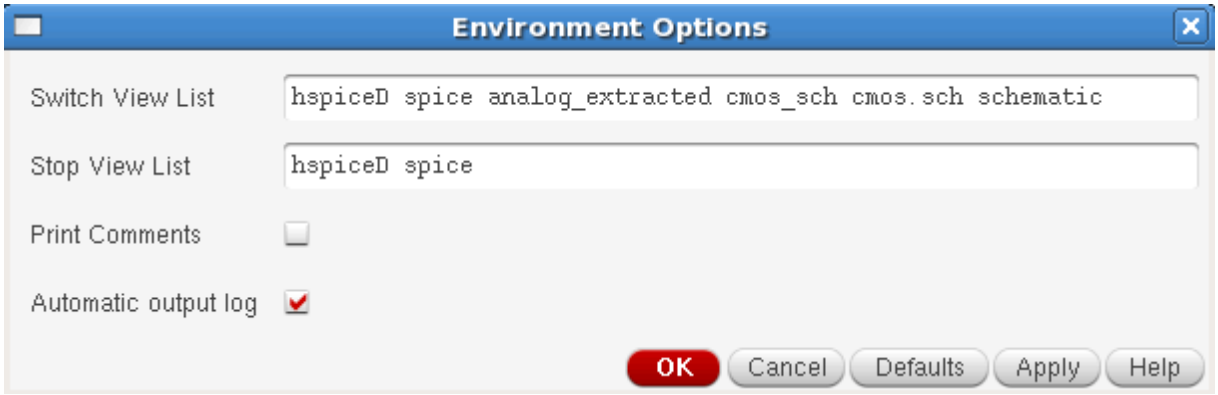
i. Setup → Simulator: Spectre

ii. Setup → Libraries… Select
/usr/cad_tools/cadence/NCSU/ncsu-cdk-1.6.0.beta/models/spectre/nom/ami06N.m
and
/usr/cad_tools/cadence/NCSU/ncsu-cdk-1.6.0.beta/models/spectre/nom/ami06P.m

iii. Analyses → Choose…. Do transient for 1µs.

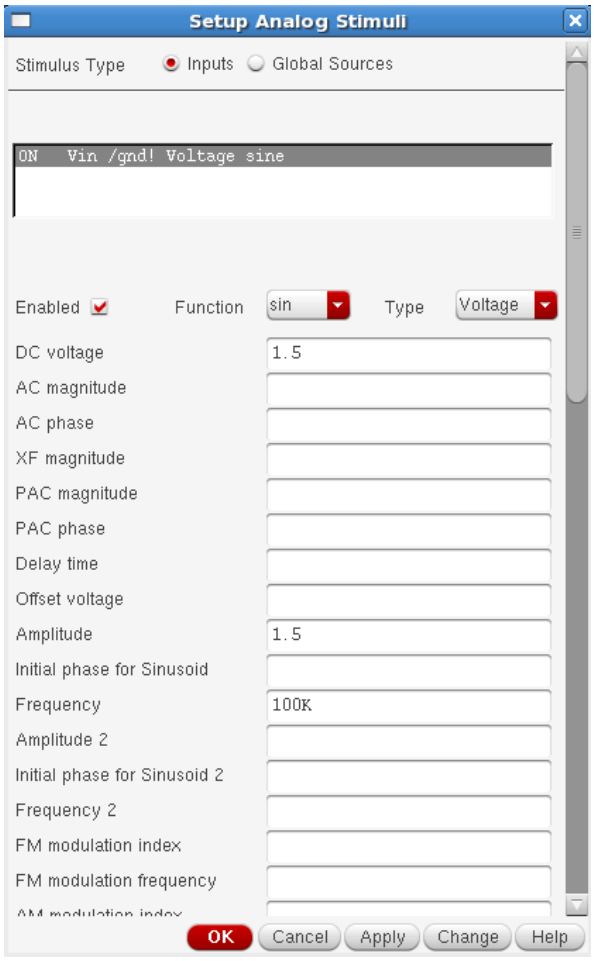
iv. Outputs → To Be Plotted… Select (by clicking on extracted view) VDD!, GND!, Vin and Vout.

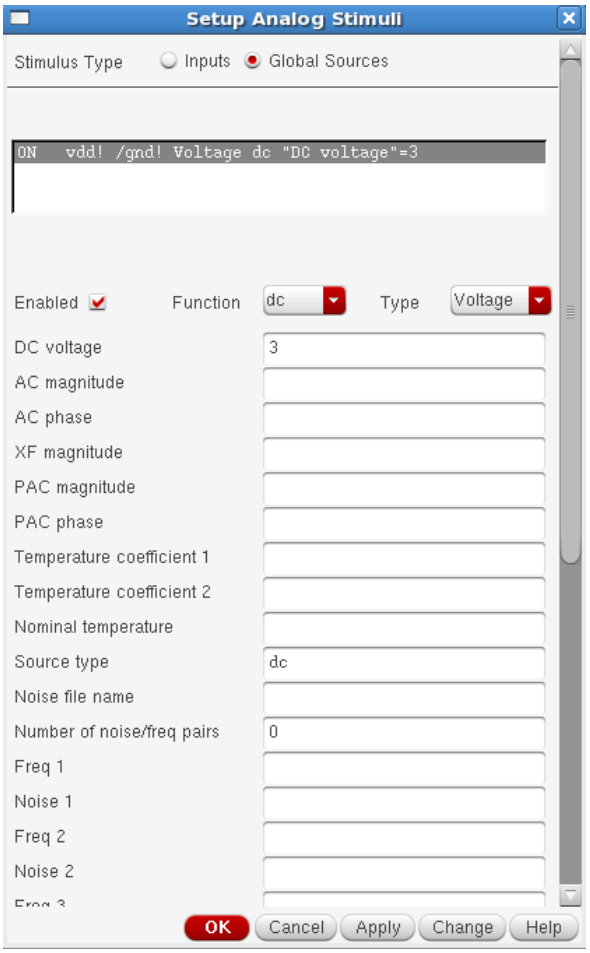
v. This step is the only step different than the way you simulated your schematic. Setup → Environment… In the “Switch View List”, add ‘analog_extracted’ anywhere before ‘schematic’ in the list:



This list is an ordered list of cell views. The simulator (actually the netlister) will search until it finds one of these views. By adding analog_extracted BEFORE schematic, the simulator will see the analog_extracted view first and simulate that instead of the schematic view.

vi. Set stimuli: Setup → Stimuli… . For the stimulus type “Input”, Vin should be enabled and set to a 1.5V sin wave with a DC offset of 1.5V. The frequency should be 100KHz. For the stimulus type “Global Sources”, vdd! should be enabled and set to a DC value of 3V. (Diagrams on next page). Click on “OK”.
I have a simple fix for the stimuli window that should work. If I haven’t shared it with you tell me to now!





e. There is another way to set stimuli. **NO NEED TO DO IT NOW** but if you ever want to write out the simulation information in a text form you would:
   i. Open up some kind of text editor.
   ii. Enter:

```
simulator lang=spectre
Vdd (vdd! 0) vsource dc=3
v1 (Vin 0) vsource type=sine sinedc=1.5 ampl=1.5 freq=100K
c0 (0 Vout) capacitor c=3f
```

   iii. (More information on writing stimulus files in Spectre in “Stolen Guide To Writing Stimuli Files” and “Spectre Users Manual”. Both posted on PolyLearn)
   iv. (The capacitor on Vout is a common thing to do when testing circuits since most circuit will be driving some capacitive load and, ideally, you want to test your circuit in conditions as close as possible to the conditions it will be running in the real world).
   v. Once you’ve saved your stimulus file, you would go back to the simulation window and then to: Setup → Simulation Files… and under “Stimulus Files” you’d add the file you just wrote. Make sure that the checkbox in front of the file is checked. Cadence does this so you can include many stimulus files and just select between them by checking the box of the one you want.

f. After everything is set, click on the green arrow or go to Simulation → Netlist and Run. The waveform should appear.

g. Hopefully your simulation ran successfully. You can save some time next time you want to simulate this circuit by saving your setup. In the simulator window, go to: Session → Save State … Give this simulation a name (enter the name in the "Save As" field). Press OK. When you want these settings back, go to Session → Load… . I've found that not all the information I thought I saved is actually saved with NCSU on IC615 so, if you are using an AMI process, always check all settings before you simulate.

h. To view the individual signal easier, on the waveform window, try Graph → Split Current Strip. You can drag each strip to move it to a different position in the stack. The icons top right split and merge the strips: 

i. To put a permanent marker on your graph, click on the point where you want a marker and then press 'm'. That will give you a "point" marker. If you want get all the curve values at a particular X value, click on any point that has the X value you want and then press 'v'. If you want get all the curve values at a particular Y value, click on any point that has the Y value you want and then press 'h'. There are a number of things you can set through Marker → Create Marker… If you want to learn about the options I'm just going to point you to "Help".

j. In the first tutorial I mentioned the calculator. You can always combine curves mathematically to make a new curve.

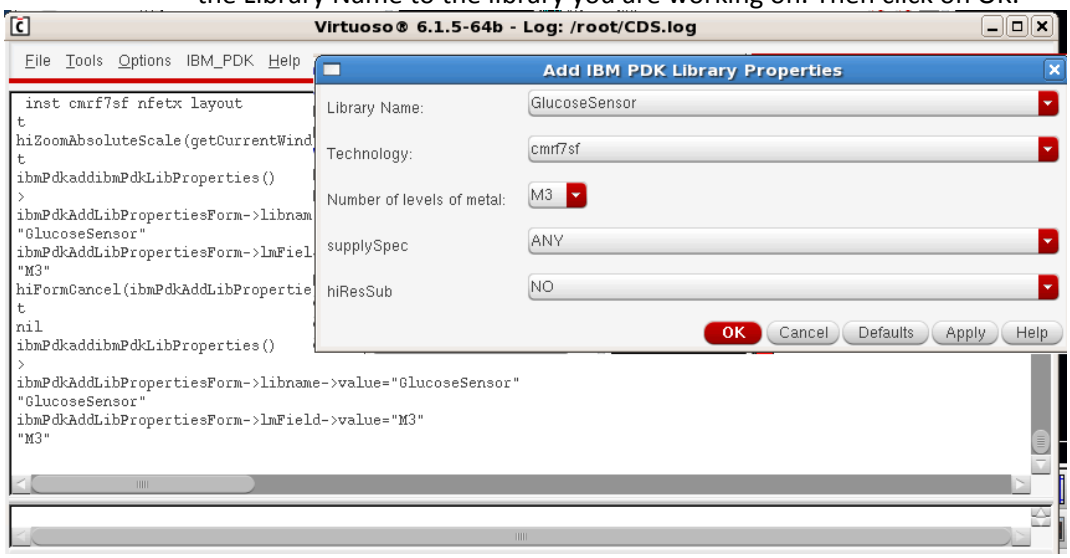
You may be surprised but that's the end of the timing analysis section of this tutorial.

3. PCELLs:

This section takes you through laying out your inverter using PCELLs instead of drawing each layer by hand. We'll be using transistor PCELLs which, again, are premade cells that can be changed into the size and shape of transistor you want by setting some parameters.

a. Setup:

i. To make sure we can do the LVS and DRC checks, we need to add the library properties to the design. Go to the CIW window (main window) and go to: IBM_PDK → Library → Add IBM_PDK Lib Properties". Set metal layers to 3 and the Library Name to the library you are working on. Then click on OK.

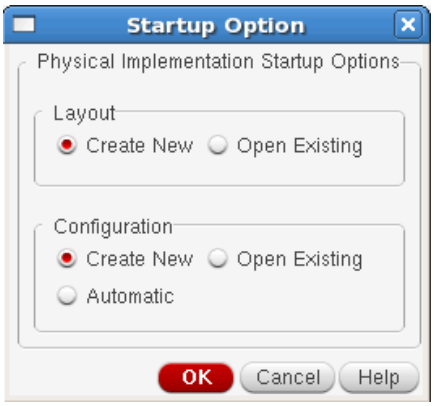


ii. First, go to the library manager and open the schematic view of your inverter. When the schematic editor comes up, copy it to a Cell Name of Inverter2 or

something. To do that: File → Save a Copy… and put in the new name. I will use the name Inverter2 in the rest of this description. You should, of course, use the name that you gave the new schematic in place of "Inverter2". The reason to make a copy of the schematic cellview is so that the work you do on an automatic layout doesn't overwrite the work you did on your hand done layout.

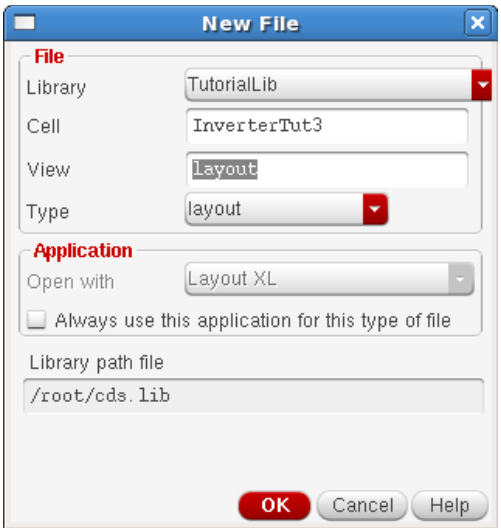
iii. Close the original inverter schematic and open your new schematic (Inverter2?).

iv. On schematic, select: Launch → Layout XL. Layout XL allows you to connect the schematic and the layout whereas Layout L doesn't – it just lets you draw a layout without using any information from the schematic.

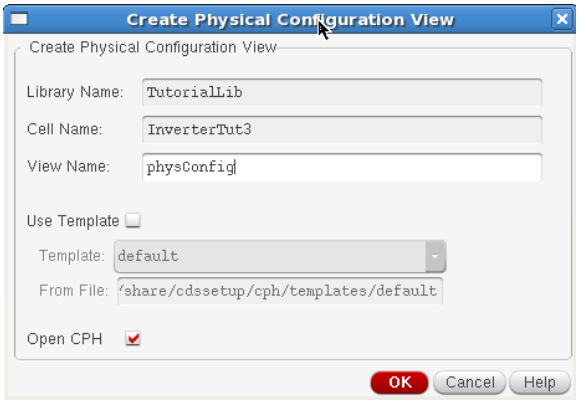


v. In the "Startup Option" window select "Create New" for both the Layout and Configuration and then click "OK":

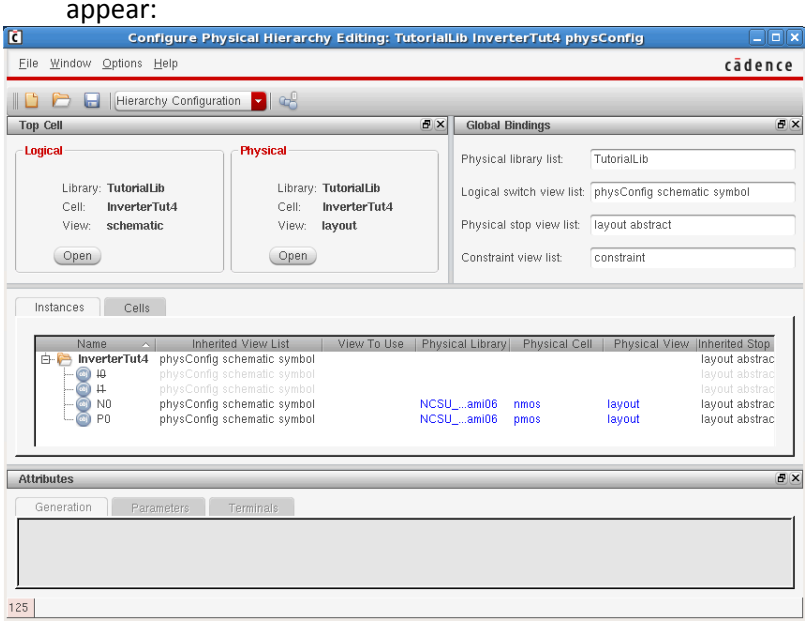
vi. The "New File" window should come up. Set as follows and click "OK". Don't worry if you can't set the "Open with" field to "Layout XL". That's what will come up (after a window asks if it's OK for the suite to use "XL" instead of "L"):

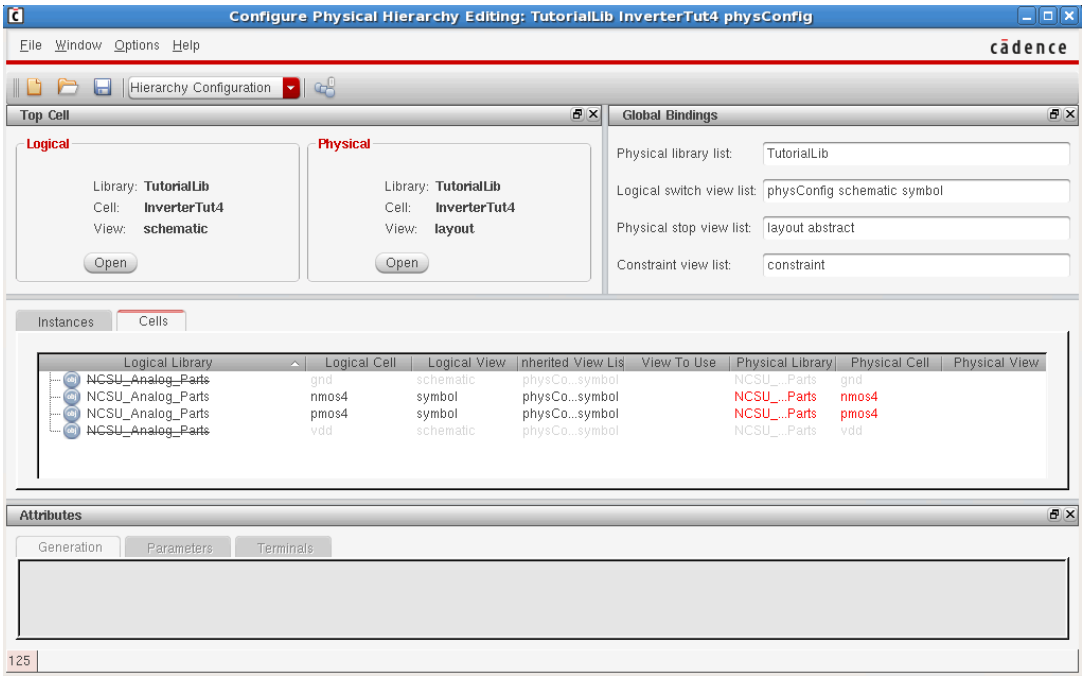


vii. In the “Create Physical Configuration View” window set the following values (they should already be set by default) and click “OK”. This creates a “physConfig” cellview that contains some important information for the layout:



viii. Hopefully “Open CPH” was checked and if it was the following window should appear:





Make sure your window has the Physical library, Physical cell and Physical view set for your transistors, resistors and/or capacitors in your design. If not then there is a possibility that the software and/or libraries aren't set up correctly. You often can get around missing values by clicking in the column + row of the missing value and setting it but there is a good chance you will have trouble later if the software wasn't able to find it when displaying this window. With AMI, there seemed to be problems initially with using nmos4 and pmos4 and we set them to nmos and pmos instead (by clicking on the pmos4/nmos4 text and resetting it to nmos/pmos).

ix. Click on the floppy (save the physConfig view). Then File → Exit…

b. Setting display options:
   i. Check the title bar and make sure that the Layout XL tool came up. If the Layout L tool cam up you can change it to the Layout XL tool by: Launch → Layout XL.
   ii. Open the Options → Display window. Values are on next page.

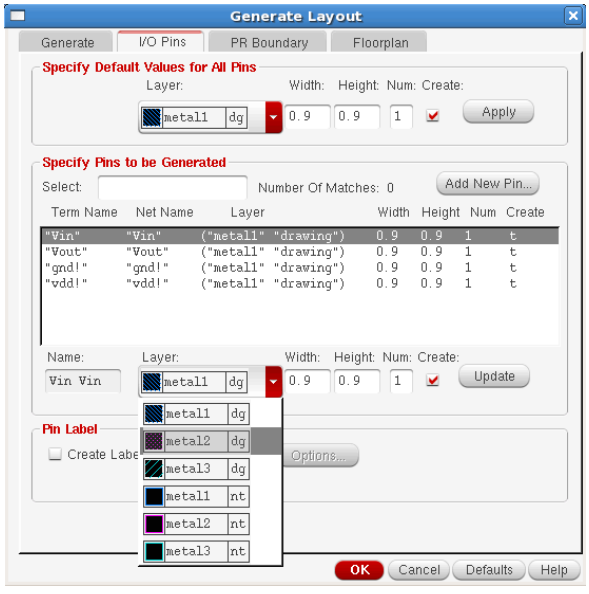
c. Convert schematic to (unplaced) layout:
   i. In the Layout XL tool: Connectivity → Generate → All From Source… A window with four tabs will come up. Screen captures of the tabs (with example settings) are on next page(s). Your settings will be a little different (a little simpler) and will be, for everything, “Layer”=“M1” and “Create”=’t’ on the IO Pins tab.

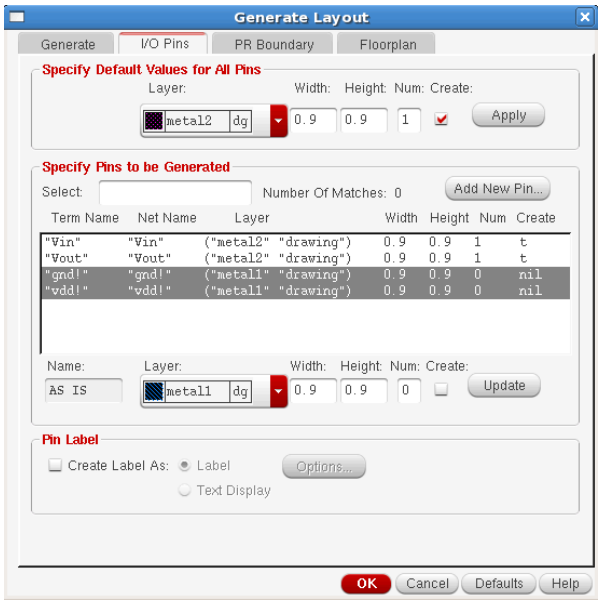
ii. Generate tab settings: (Set as in screen capture below).

iii. I/O Pins tab settings:

- GENERAL - wires: Since an inverter is a simple circuit, all wires can be on Metal1. Use M1 – dg (Metal1 drawing). There won't be any (wire) collisions. This won't work with even a slightly more complicated circuit – even a NAND. In that case you want the default to be Metal2, power on Metal1 and possibly the other signals on Metal2 or Metal3. There are other opinions on this so playing with the layers is fine if you use this method to layout your design for the project.
- GENERAL: If you uncheck Create then it means that you will place the part by hand and it shouldn't be automatically placed.
- To set/unset "Create" to "t"/"nil", select the row of the wire or contact and then check/uncheck "Create". Then click "Update".
- To set change metal layer of a wire, select the row of the wire or contact and then select metal layer from the pulldown menu. Then click "Update".

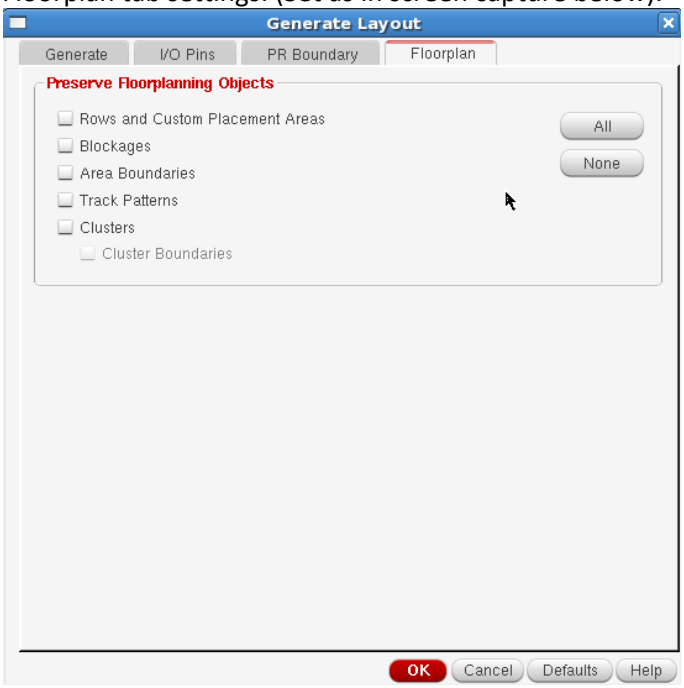
How to change metal layers and "Create" settings of Vin:





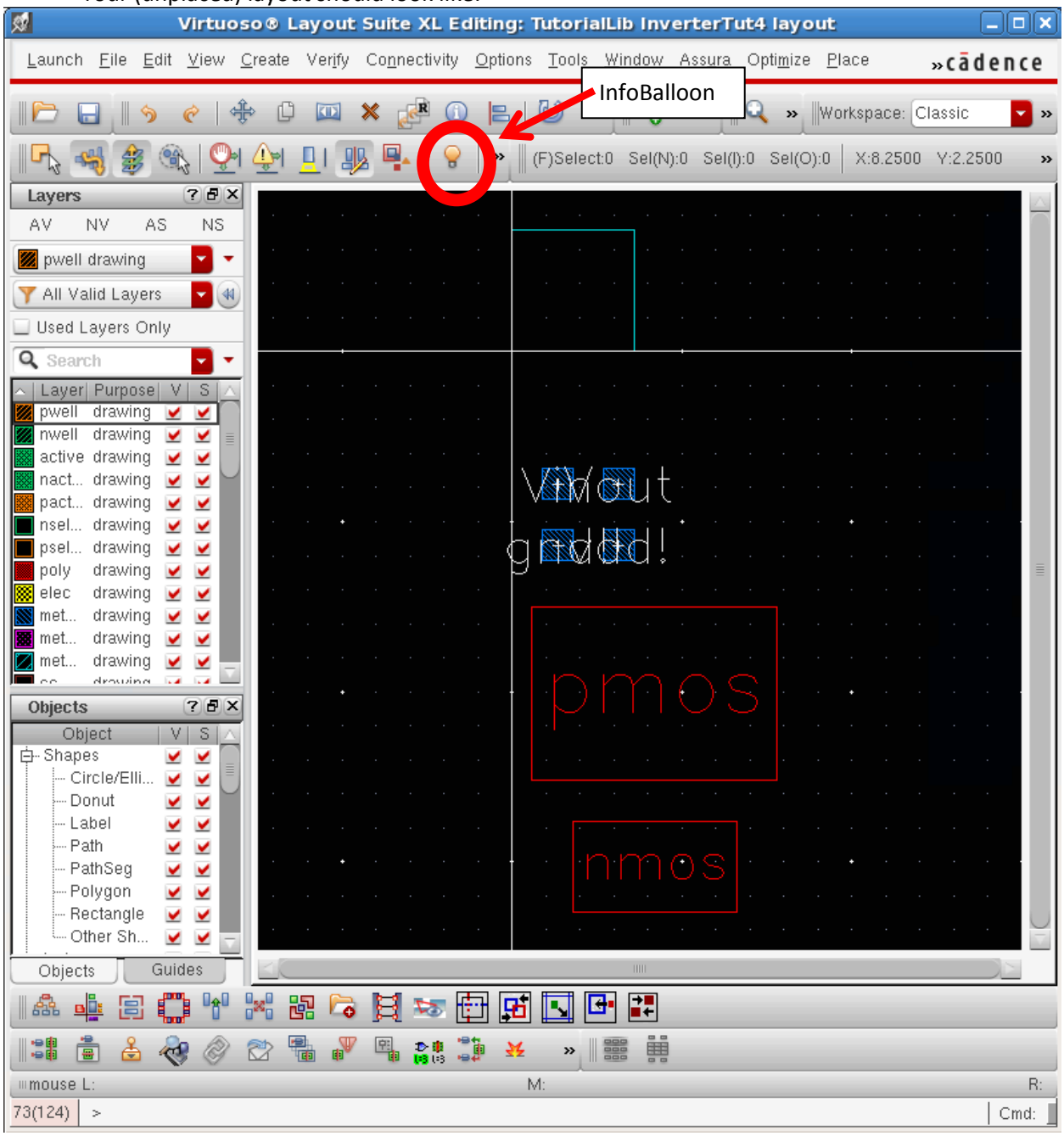
iv. PR Boundary tab settings: (Set as in screen capture below).

v. Floorplan tab settings: (Set as in screen capture below).



Then click “OK”.

Your (unplaced) layout should look like:



d. Playing with automatically placed layout

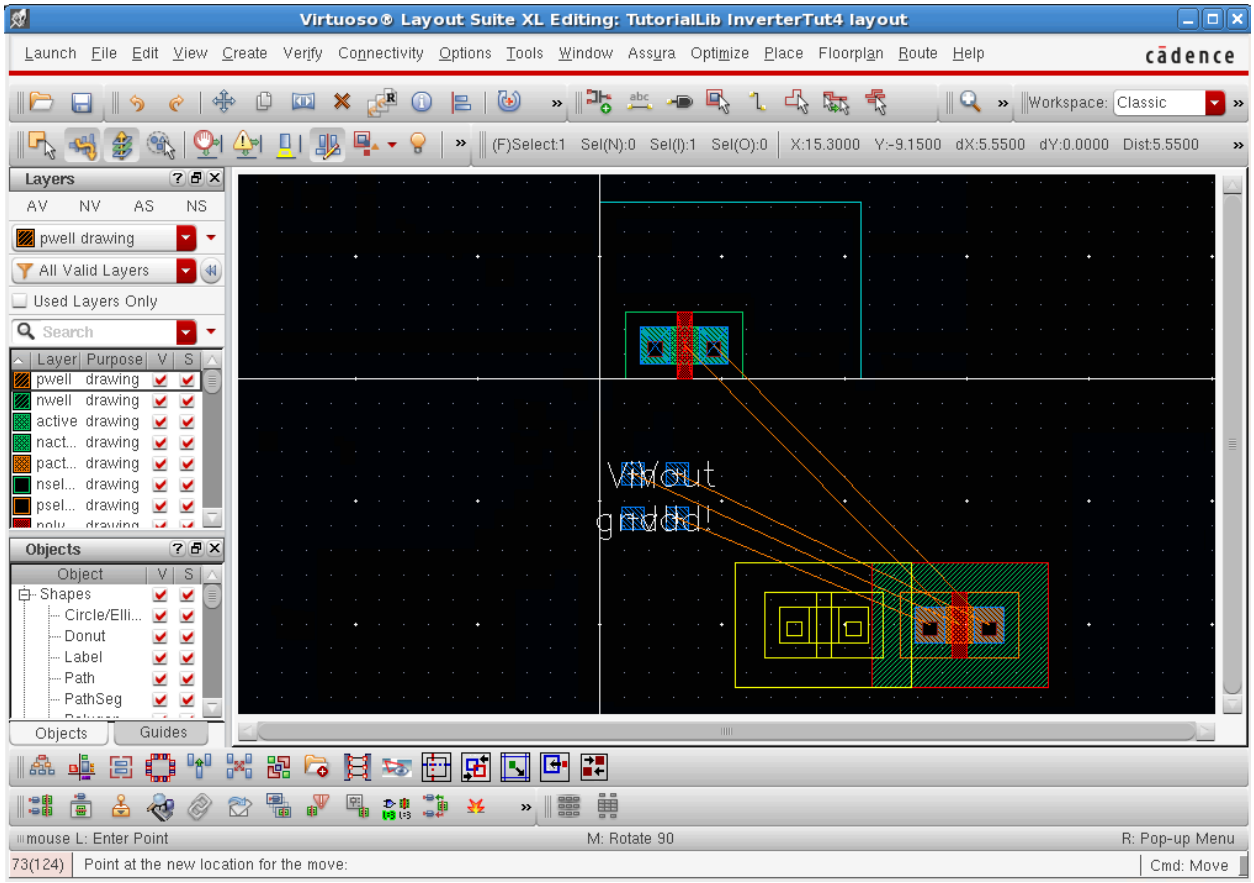
   i. Let's play a little first. Click on (select) an instance in the layout. You'll see it selected in the left window of the SCHEMATIC window. Selecting and deselecting can be done in the schematic window not just for the schematic but for the layout also.

   ii. Click on the "Move" icon  . Click on the NMOS and you'll see flight-lines. Two to the PMOS, one to Vout, one to Vin and one to gnd!. These are the wires

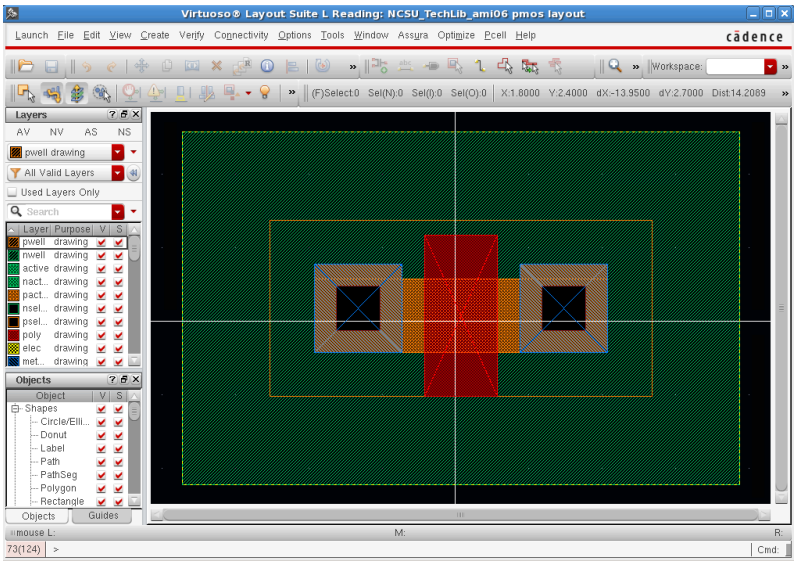
that connect all the components in your layout. They aren't "routed" yet so they are there just to show you what is connected to what.

iii. Go to the infoBalloon and turn it on. Bring the mouse pointer over your design. It should tell you pin names, sizes among other things.

iv. To get a view of what's inside each of those squares (show all levels), press SHIFT+F. (The key bindings are different for AMI and the other libraries. I am making sure that the common bindings file doesn't cause trouble and hope to have the same bindings for all libraries by Fall 2013. This won't work with the present bindings in libraries other than AMI). Now you can see orientation and where contacts are for each transistor. Turn the InfoBalloon on (if you turned it off) and note that you can now see what the layers are too now. Click on MOVE again and you can see that the flight-lines make it clear which contact is the source and drain of the transistors. To leave the detailed view press CTRL+F (return to level 0). Here's a view with SHIFT+F and a move. You can see the flight-lines.



v. A way to look inside a single transistor is to: Select the transistor of interest (left click) and then right click on it. And select "Descend Read". Now you should be able to see all the stuff you drew when you drew the transistor by hand. To return, right click on the background (grid) and select "Return".



vi. The name “PCELL” comes from “parameterized cell”. What that means is that there are parameters that you can set and the transistor will change without you having to redraw layers and all that other stuff you’d have to do if you if you had drawn the transistor by hand. Select a transistor and press ‘Q’. Click on “Parameter”. Set the width to 15µ and then press “Apply”. You should see a change in width of the transistor. (I’ll be asking you to include a screen capture for postlab so grab one right now. To do that with the least pain, bring the focus – which window is selected – to anything on the local computer’s desktop. Then do CTRL+PrntScrn. Go to Paint and do a CTRL+V then edit in Paint. Please write if that doesn’t make sense). Note the added contacts. That’s to reduce the resistance of the signals to the source and drain. You may ask why you have a lot of separate squares rather than one continuous band of metal if your goal is to reduce resistance. It turns out that bands of metal tend to get cracks in them. Just using a lot of contacts are a more robust way to reduce resistance. Once you’ve got your screen capture, return the width to 1.5µ or else LVS won’t pass.

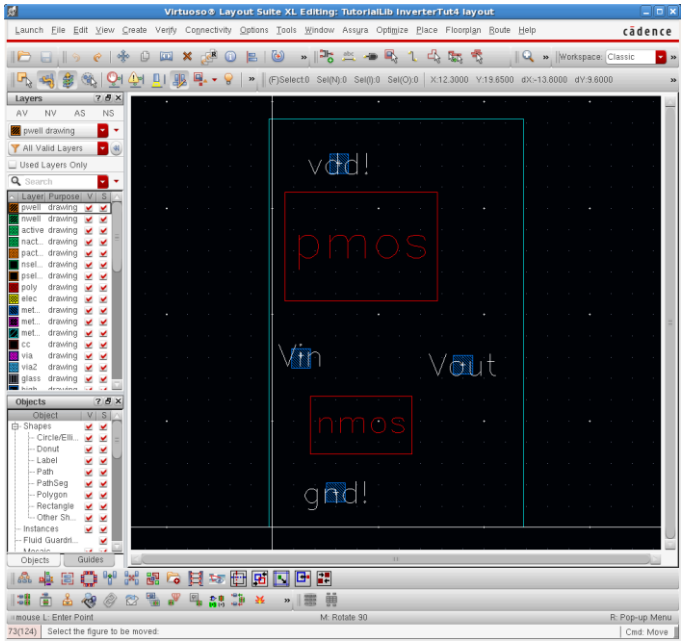
vii. While we’re there, see what happens when you change the number of fingers to 4. (Screen capture for postlab).

viii. SOMETHING TO NOTE: The blue empty box (“bounding box”) is the box that Cadence thinks all the parts can fit into. Haha. Not quite…

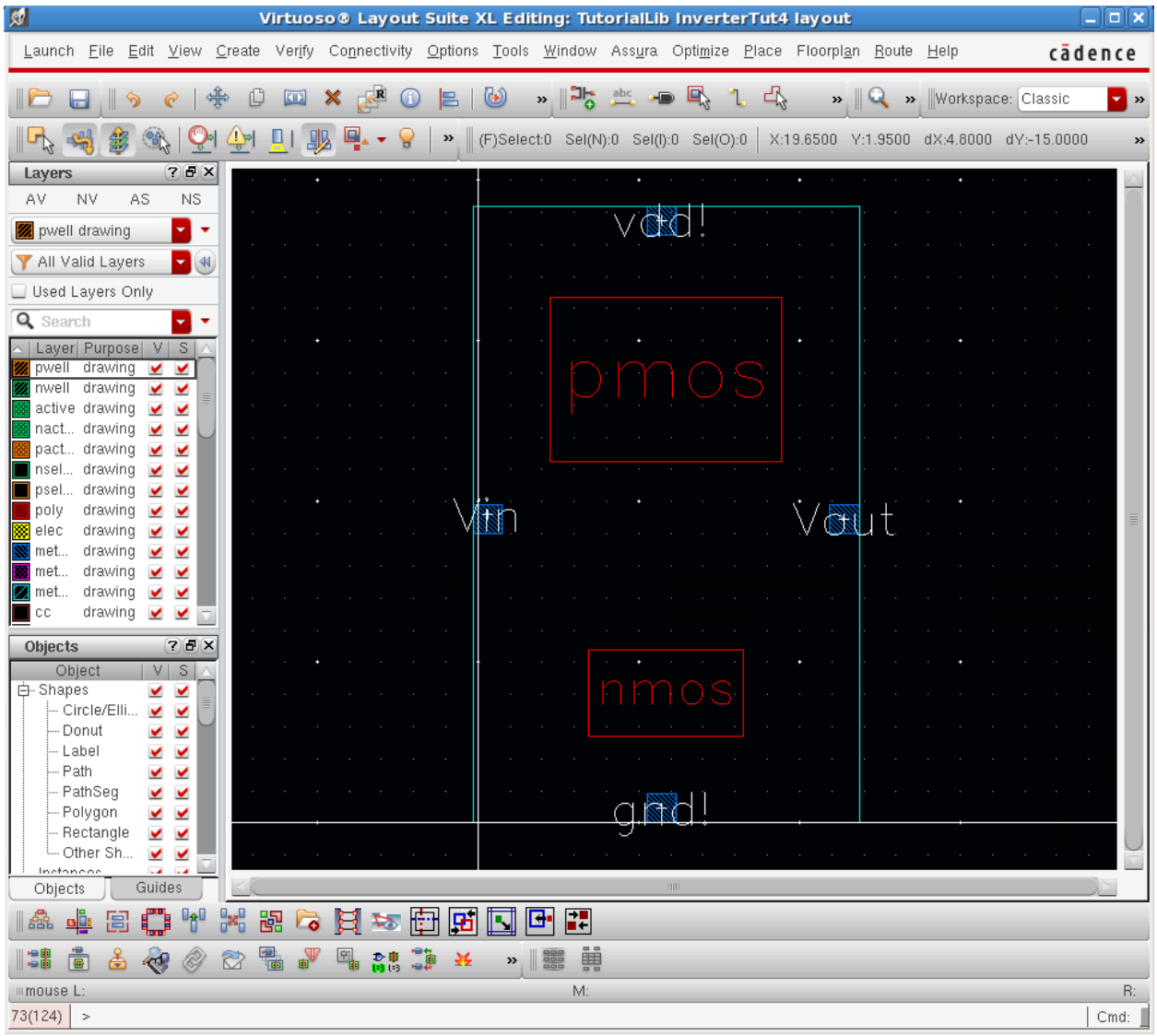
e. Placing parts in the bounding box

i. Ignoring the bounding box place the NMOS, PMOS, GND! contact and VDD! contact so that the gate goes up and down (not across) and GND! and VDD! are where they were in the layout. No need to be too careful. We’ll let the software place stuff better in a minute. Try to get the flight-lines so they don’t cross. Use SHIFT+F and CTRL+F as needed to make your job easier. (Again, this doesn’t work in libraries other than AMI).

ii. Changing the size of the bounding box (BB). In theory, you should be able to select the BB and then right clicking then select “Stretch” from the menu. If you have trouble using the menus (I did), use the shortcut ‘S’. (**Bug? NCSU CDK issue with IC615?**) Then grab one side of the BB and drag it so that the parts (NMOS, PMOS, VDD! contact and GND! contact) are contained within the BB. It might be useful to place an instance of your hand layed out inverter for comparison. To do that, go through the same steps as when you placed a component (like a transistor) but place the layout view you created last week. Last week’s bounding box was about W=10 and L= 21.3. I had to make today’s box about W=11.7 and L=21.3 to get a successful automatic placement.



iii. Let the software do the placement: Connectivity → Generate → Place As In Schematic. That will move the parts to fill your BB. “Placement” is a technical term and means exactly what it means: To put the parts of your chip in a position. I won’t say “final position” because, of course, may move the parts as you’d like post-automatic-placement. But if you just want a legal layout (that is probably not optimal in one or more of the metrics speed, power or size) automatic placement is handy. (If you want to see another version of a placement, got to “Place→Analog → Quick Placement” or “Place→Analog → Automatic Placement…”).



f. Routing of your circuit: You can always follow the same steps you did in the previous tutorial to connect (draw wires between) up your devices or follow the steps in the next section. Here's a little review of what we did in the last tutorial:
    i. If you are connecting two materials that are the same, just overlapping the squares tells the software that they are attached.
    ii. If you are connecting silicon to metal 1, you draw a square of "cc" where you want the two layers (metal 1 and silicon) to connect. Overlapping a metal 1 square over where you've drawn a "cc" square tells the software that they are attached.
    iii. If you want to connect metal 1 to metal 2, you need to draw a square of "via".
    iv. If you want to connect metal 2 to metal 3, you need to draw a square of "via2".
    v. If you want to connect metal 1 to metal 3 you need to draw a "via", then a square of metal 2, then a "via2". Don't forget the metal 2 or the vias won't be connected.

Today we'll get a little bit of help from the software but we'll still have to do some things by hand:

a. Draw a 3µm wide band of metal 1 above the PMOS and below the NMOS. They should overlap the vdd! and gnd! pins respectively. Use the infoBalloon to check and make sure that the software knows that these squares should be vdd! and gnd!. (i.e. Check the net name and make sure it's either vdd! or gnd! depending on which pin the square overlaps).
b. Draw an nwell around the PMOS and up to the top of the vdd! metal 1 rail.
c. Connect the gates using poly. The automatic placement will probably have lined up the two transistors and you should be able to connect them with a single poly rectangle.
d. Do Route → Automatic Routing… and then, with the default values, click "Run". **
e. Note that the gate isn't connected to Vin. (**BUG? **. It connects in the libraries other than the NCSU so I think this may be a CDK issue**). The infoBalloon doesn't show the poly as Vin which I think is a bug. Even if your gate isn't connected to the Vin pin, let's go do a quick DRC check to make sure we're OK up to here. The nice thing about automatic routing is that it shouldn't introduce any new DRC errors. If you are using AMI, go to Verify → DRC…, then click "OK". If you are using another library, look for the library PDK menu (for example, look for IBM_PDK if you are using cmrf7sf or cmrf8sf) and select Checking → Assura → DRC. If you have errors fixe them before moving on.
f. If your gate wasn't automatically connected to the Vin pin, connect them now:
    i. Select **poly** layer from the LSW. Draw a rectangle right next to the middle of the (gate) poly you just drew.
    ii. Select **cc** layer from the LSW. Add a rectangle in the center of the poly square you just drew.
    iii. Select **metal1** layer from the LSW and draw a square on top of the poly square you just drew.
    iv. Draw a rectangle connecting the Vin pin to the square you just drew.
    v. Do a DRC check and fix any errors. Common errors are having the poly too small, the contact too small, metal 1 too close to other metal 1, etc. I can't post DRC rules because there are still people that need to sign the NDA.**
g. To do your LVS: (**Still some issues as of posting this document for libraries other than AMI. To be fixed/updated soon. Here too Assura will be used.**)
    i. Go to Verify → Extract… Click on "Set Switches" and select "Extract_parasitic_caps". Then click on "OK".
    ii. Open the extracted view from the library manager and run Verify → LVS… Make sure the "Rules Library" is set to NCSU_TechLib_ami06. Then click "Run".

This week you've used PCELLS to save you some time in drawing transistors, automatic placement to speed up placement and automatic route to draw in some of your wires. This is a simple circuit and all these software steps like automatic placement, LVS, DRC and automatic routing (hopefully) finished pretty quickly. Complicated circuits can actually take hours or days to run. The trick to use with large chips (say 1000000 transistor chips) is to divide your circuit into blocks and check each block separately. Some checks like timing analysis (basically simulation) can take days to run on the extracted views. To speed this up, there is an abstracted view that produces a black box that only provides input capacitance, output capacitance and delay through the circuit. Then static timing analysis is run and these numbers for capacitance and all are used to find delays from one place to another without really running a simulation. This gives a ballpark idea of how fast the circuit will run.

POSTLAB DUE STUFF:

1. Select a transistor and press ‘Q’. Click on “Parameter”. Set the width to 15µ and then press “Apply”. You should see a change in width of the transistor. (Include a screen capture for postlab).
2. email me your LVS “Output”. (Click on the Output button and send me the log).
3. While we’re there, see what happens when you change the number of fingers to 4. (Screen capture for postlab).
4. On your final extracted view, do a SHIFT+F and grab a screen shot.
5. Make up one or two questions about this lab that you’d like answered.